

COMMON WEBAPP SECURITY THREATS

... and what to do about them

Eoin Woods
@eoinwoodz
Endava

Content

- Introducing Web Security Threats & OWASP
- The OWASP Web Vulnerabilities List
- Useful Tools to Know About
- Reviewing Defences
- Summary

Introducing Web Security Threats

Web Security Threats

- We need systems that are **dependable** in the face of
 - Malice
 - Error
 - Mischance
- People are sometimes **bad, stupid** or just **unlucky**
- System security aims to **mitigate** these situations

Web Security Threats

- System threats are similar to **real-world threats**:
 - Theft
 - Fraud
 - Destruction
 - Disruption
- Anything of **value** may attract unwelcome attention

“I rob banks because that’s where the money is” – Willie Sutton

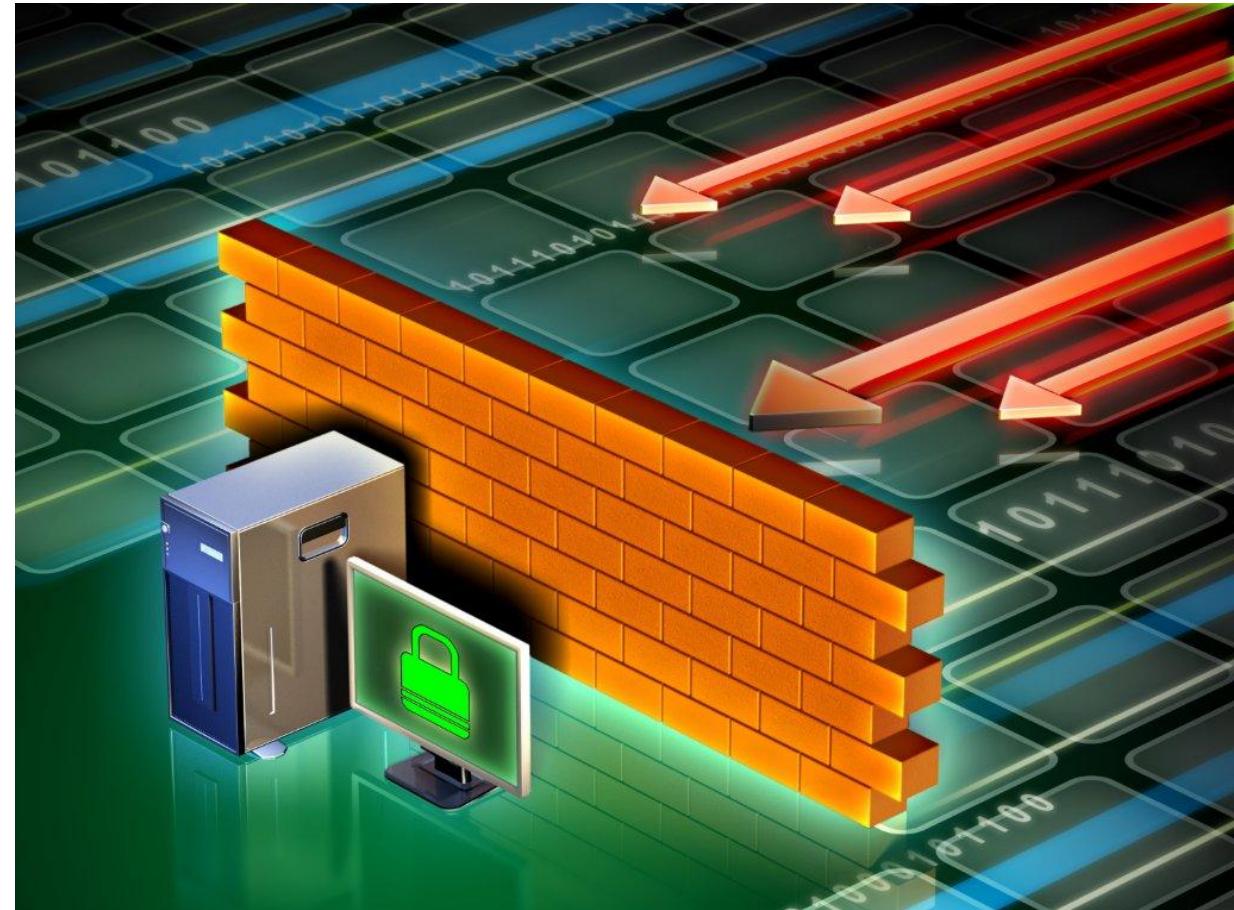
Web Security Threats

- Why do we **care** about these threats?
 - A threat is a **risk of a loss** of some sort
- Common types of **loss** are:
 - Time
 - Money
 - Privacy
 - Reputation
 - Advantage

Web Security Threats

- **Digital channels** need security
- **APIs on the Internet**
- **Introspection** of APIs
- Attacks being “**weaponised**”
- Today’s internal app is tomorrow’s “digital channel”

Security today mitigates tomorrow's threat



Who are OWASP?

- The Open Web Application Security Project
 - Largely volunteer organisation, largely online
- Exists to improve the state of software security
 - Research, tools, guidance, standards
 - Runs local chapters for face to face meetings
- “OWASP Top 10” project lists top application security risks
 - Data-driven list of most significant threats to webapps
 - Referenced widely by MITRE, PCI DSS and similar
 - Updated as threats change (2003, 2004, 2007, 2010, 2013, 2017)



Other Important Security Organisations

- MITRE Corporation
 - Common Vulnerabilities and Exposures (CVE)
 - Common Weaknesses Enumeration (CWE)
- SAFECode
 - Fundamental Practices for Secure Software Development
 - Training



There are a lot of others too (CPNI, CERT, CIS, ISSA, ...)

OWASP Web Vulnerabilities List

How was the 2017 List Produced?

- Project of the OWASP organisation
 - Group of ~75 volunteers create it
- Data set analysis
 - Data from 24 firms including Aspect Security, Checkmarx, MicroFocus, NCCST, Synopsis, TCS, Vantage Point, Veracode, ...
 - Data represents ~114,000 applications
 - <https://github.com/OWASP/Top10/2017/datacall>
- Survey analysis
 - ~500 participants from the OWASP Top 10 mailing list



OWASP Top 10 - 2017

#1 Injection Attacks

#2 Broken Authentication

#3 Sensitive Data Exposure

#4 XML External Entities (XXE)

#5 Broken Access Control

#6 Security Misconfiguration

#7 Cross Site Scripting (XSS)

#8 Insecure Deserialisation

#9 Component Vulnerabilities

#10 Insufficient Logging and Monitoring

Some may look “obvious” but appear on the list year after year, based on real vulnerability data!

What Changed from 2013 to 2017?

OWASP 2013 Top 10	OWASP 2017 Top 10
A1 – Injection	A1 – Injection
A2 – Broken Authentication & Session Management	A2 – Broken Authentication
A3 – Cross-Site Scripting (XSS)	A3 – Sensitive Data Exposure
A4 – Insecure Direct Object References	A4 – XML External Entities (XXE) -- NEW
A5 – Security Misconfiguration	A5 – Broken Access Control
A6 – Sensitive Data Exposure	A6 – Security Misconfiguration
A7 – Missing Function Level Access Control	A7 – Cross Site Scripting (XSS)
A8 – Cross Site Request Forgery (CSRF)	A8 – Insecure Deserialisation -- NEW
A9 – Components with Known Vulnerabilities	A9 – Components with Known Vulnerabilities
A10 – Unvalidated Redirects & Forwards	A10 – Insufficient Logging and Monitoring -- NEW

#1 Injection Attacks

Exploitability	3
Prevalence	2
Detectability	3
Tech Impact	3

- Unvalidated input passed to any interpreter
 - Operating system and SQL are most common
 - Configuration injection often overlooked

```
SELECT * from table1 where name = '%1'
```

Set '%1' to '**OR 1=1 --** ... this results in this query:

```
SELECT * FROM table1 WHERE name = '' OR 1=1 --
```

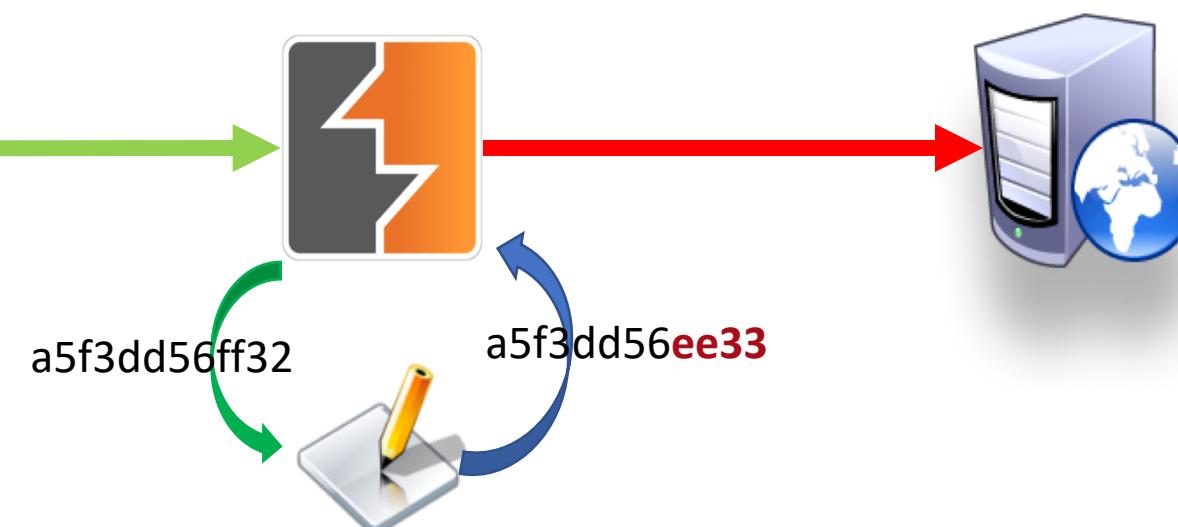
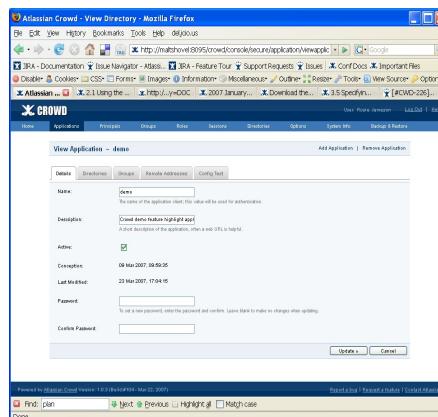
- Defences include “escaping” inputs, bind variables, using white lists, ...

(See also #4 – XML External Entities ...)

#2 Broken Authentication

Exploitability	3
Prevalence	2
Detectability	2
Tech Impact	3

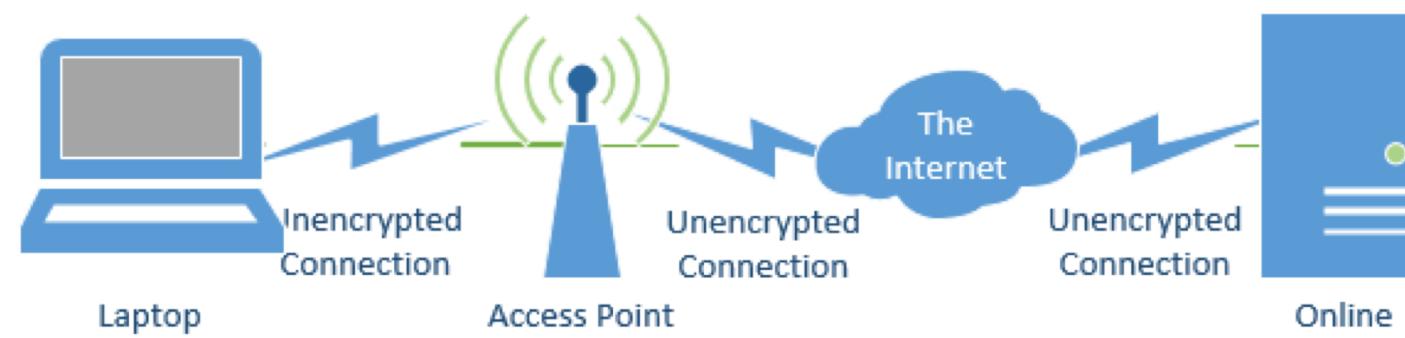
- Credential Stuffing - millions of usernames and passwords available
- Well known credentials often present
- Unprotected session IDs
- Session IDs not rotated after login or invalidated after use
- Mitigations include strong authentication and session management controls



#3 Sensitive Data Exposure

Exploitability	2
Prevalence	3
Detectability	2
Tech Impact	3

- Is sensitive data secured in transit?
 - TLS, message encryption
- Is sensitive data secured at rest?
 - Encryption, tokenisation, separation
- Impact can include loss of data or spoofing attacks
- Mitigation via threat analysis, encryption, limiting scope, crypto standardisation

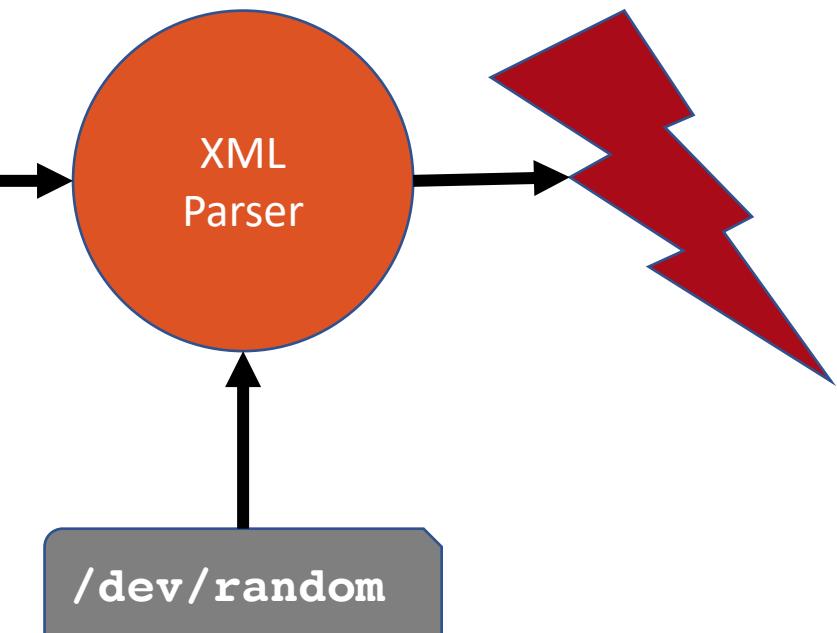


#4 XML External Entities (XXE)

Exploitability	2
Prevalence	2
Detectability	3
Tech Impact	3

- XML “external” entities cause XML parsers to retrieve external data
- Many XML parsers enable this by default (including Java’s standard library)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE danger [
  <!ELEMENT other ANY >
  <!ENTITY dos SYSTEM "file:///dev/random" >]>
<foo>&xxe;</foo>
```



- Can expose sensitive data or provide DoS attack vector
- Mitigate by disabling external entities or removing XML

#5 Broken Access Control

Exploitability	2
Prevalence	2
Detectability	2
Tech Impact	3

- Directly referencing IDs in requests (filenames, accounts, ...)
 - Not authenticating access to each on the server
 - Client can modify request and gain access to other objects
- Relying on UI or other client side code for access control
 - e.g. UI removing "update" option & not validating action on the server

`http://www.example.com/gettxn?txnid=4567`

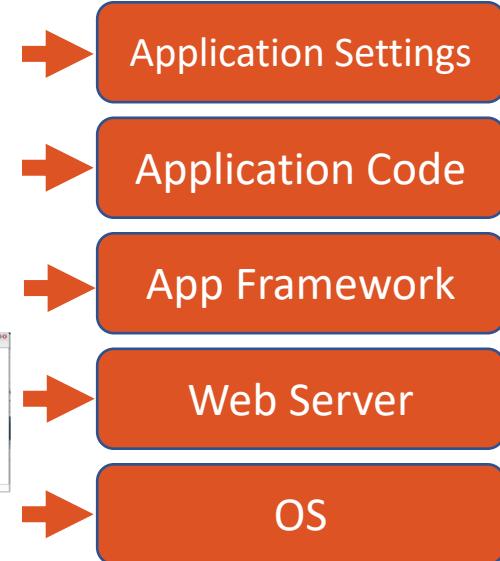
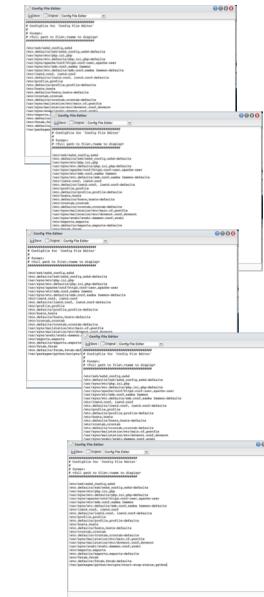
→ `http://www.example.com/updttxn?tid=4567&value=100.00`

- Not checking for tampering or replaying security meta data (e.g. JWT tokens)
- Mitigation through entity level access control, deny by default, strong and standardised authorisation technology and patterns, hide metadata

#6 Security Misconfiguration

Exploitability	3
Prevalence	3
Detectability	3
Tech Impact	2

- Security configuration is often complicated
 - Many different places to put it, complex & varying semantics
 - Layers from OS to application all need to be consistent
- It is easy to accidentally miss an important part
 - OS file permissions?
 - .htaccess files?
 - Shared credentials in test and production?
- Allows accidental access to or modification of resources
- Mitigation via scanning, standardisation, simplicity and automation



#7 Cross Site Scripting

Exploitability	3
Prevalence	3
Detectability	3
Tech Impact	2

- Occurs when script is injected into a user's web page
 - Reflected XSS attack – crafted link in email ...
 - Stored XSS attack - database records, site postings, activity listings
 - DOM XSS attack - data inserted into the browser dom
- Allows redirection, session data stealing, page corruption, ...



- Mitigations include validation and escaping data on the server-side

#8 Insecure Deserialisation

Exploitability	1
Prevalence	2
Detectability	2
Tech Impact	3

- Subverting de-serialisation mechanism
 - e.g. Java “gadgets” vulnerable to abuse with tampered objects
- De-serialising hostile code
 - e.g. serialised code that causes de-serialisation method to loop
- Mitigations include
 - only de-serialising from trusted sources
 - avoiding binary serialisation formats
 - signed serialisation data
 - whitelists of classes
 - platform security managers

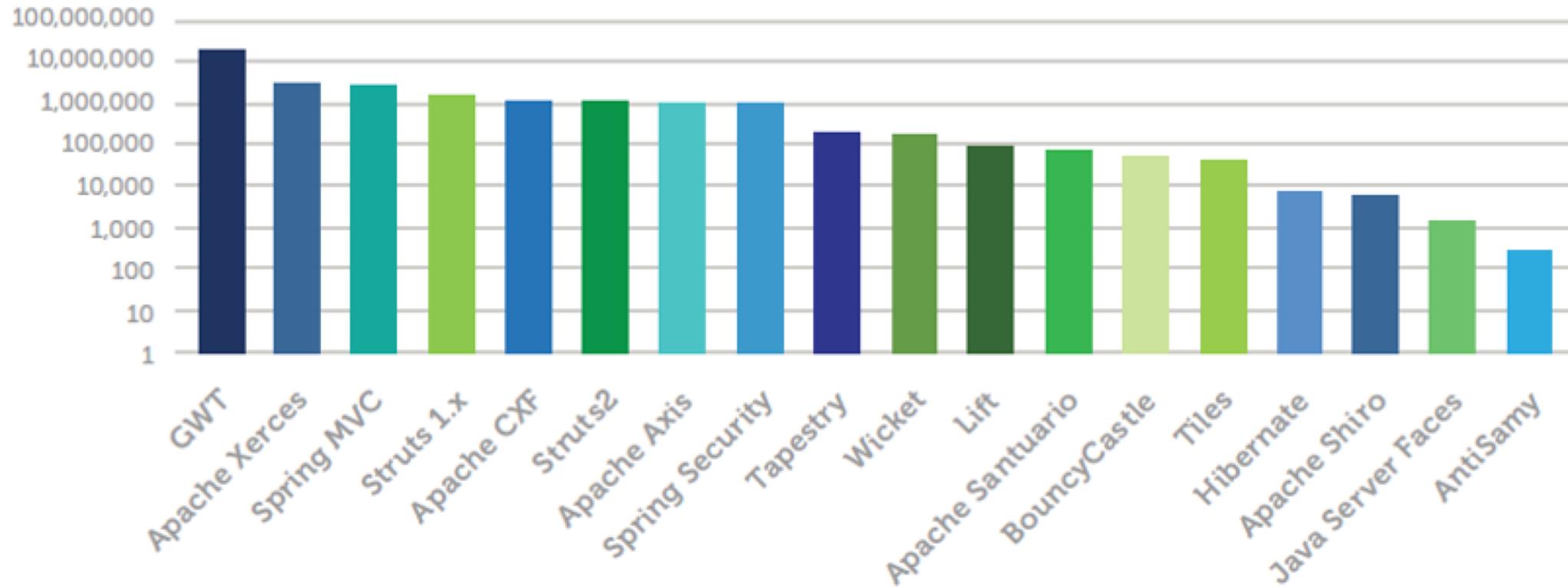
```
public void loadSession(byte[] data) {  
    ObjectInputStream ois = ... ;  
    ...  
    sess.adminUser = ois.readBoolean();  
    return sess ;  
}
```

```
public void resetUser(Session s,  
                      String user, String pwd) {  
    ...  
    if (s.adminUser) {  
        user.setPassword(pwd) ;  
        user.locked = false ;  
    }  
}
```

#9 Known Vulnerable Components

Exploitability	2
Prevalence	3
Detectability	2
Tech Impact	2

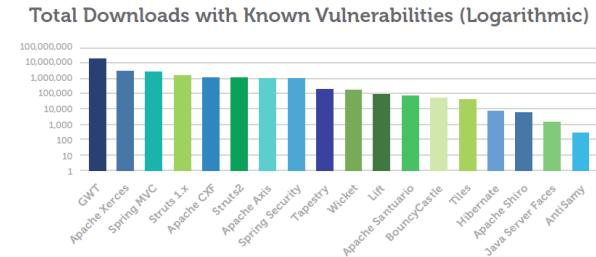
Total Downloads with Known Vulnerabilities (Logarithmic)



#9 Known Vulnerable Components

Exploitability	2
Prevalence	3
Detectability	2
Tech Impact	2

- Many commonly used components have vulnerabilities
 - See weekly US-CERT list for a frightening reality check!
 - Much OSS doesn't have well researched vulnerabilities
- Few teams consider security of their 3rd party components
 - And keeping everything up to date is disruptive
- Mitigations include automated scanning of 3rd party components, actively review vulnerability lists, keep components patched



#10 Insufficient Logging & Monitoring

Exploitability	2
Prevalence	3
Detectability	1
Tech Impact	2

- Poor logging and monitoring underpins many major exploits
- Common problems:
 - Not logging key events (failed login, high value transaction, ...)
 - Poor messages, no actionable statements
 - Lack of log analysis
- Centralise logging to provide better view and security of logs
 - Identify expected and unexpected log patterns (e.g OWASP coreruleset.org)
 - Know what to do when logs indicate unexpected situation
- Good test is to use OWASP ZAP, SQLMap and check for alerts
- Mitigations include standard log formats, key event logging, centralised logs, incident response plans, intrusion detection and SIEM systems



Summary of Main Vulnerability Types

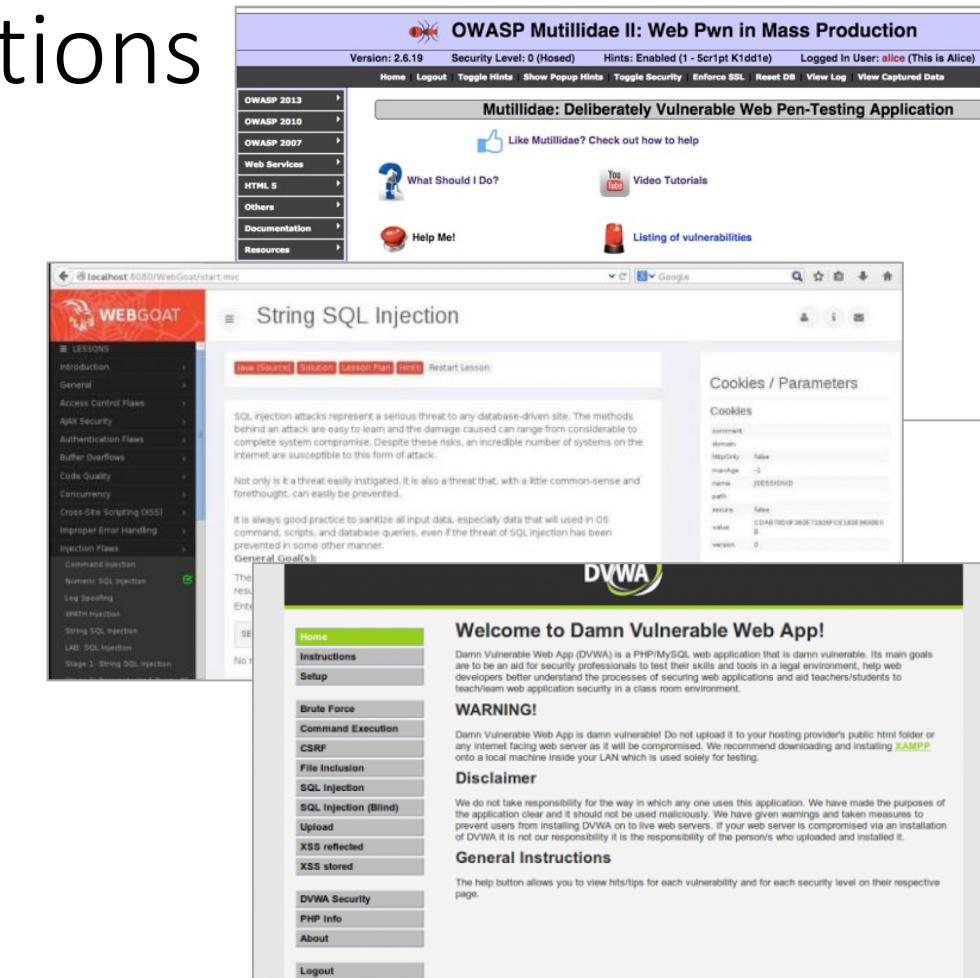
- **Interpreter and page injections**
 - Operating System, SQL, XML, deserialization, XSS, ...
- **Lack of validation**
 - trusting client side restrictions
 - allowing session IDs and cookies to be reused
 - not escaping and validating input data
 - parameter values directly in pages and links
- **Missing data protection**
 - Sensitive data exposure, deserialisation, configuration showing metadata, ...
- **Complexity**
 - Misconfiguration, deserialization, XXE, known vulnerabilities



Useful Tools

Deliberately Vulnerable Applications

- Deliberately insecure webapps
 - So run in a VM!
- OWASP Top 10 in action
 - Mutillidae & DVWA in PHP
 - WebGoat in Java



The screenshot displays three separate web application interfaces:

- OWASP Mutillidae II: Web Pwn in Mass Production**: A PHP-based application with a sidebar menu for OWASP 2013, 2010, 2007, Web Services, HTML 5, Others, Documentation, and Resources. The main content area shows a "Mutillidae: Deliberately Vulnerable Web Pen-Testing Application" with sections for "Like Mutillidae?", "What Should I Do?", "Help Me!", and a "Listing of vulnerabilities".
- String SQL Injection**: A Java-based application from WebGoat. The left sidebar lists various security lessons. The main content area discusses SQL injection attacks and shows a form for a "String SQL Injection" exercise. It includes a "Cookies / Parameters" section showing session variables like "id" (value: 8), "name" (value: JESSIEHOD), and "version" (value: 0).
- Welcome to Damn Vulnerable Web App!**: A PHP/MySQL application from DVWA. The sidebar lists various attack types: Brute Force, Command Execution, CSRF, File Inclusion, SQL Inclusion, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, and About. The main content area includes a "WARNING!" section, a "Disclaimer" section, and a "General Instructions" section.

<http://sourceforge.net/projects/mutillidae/>

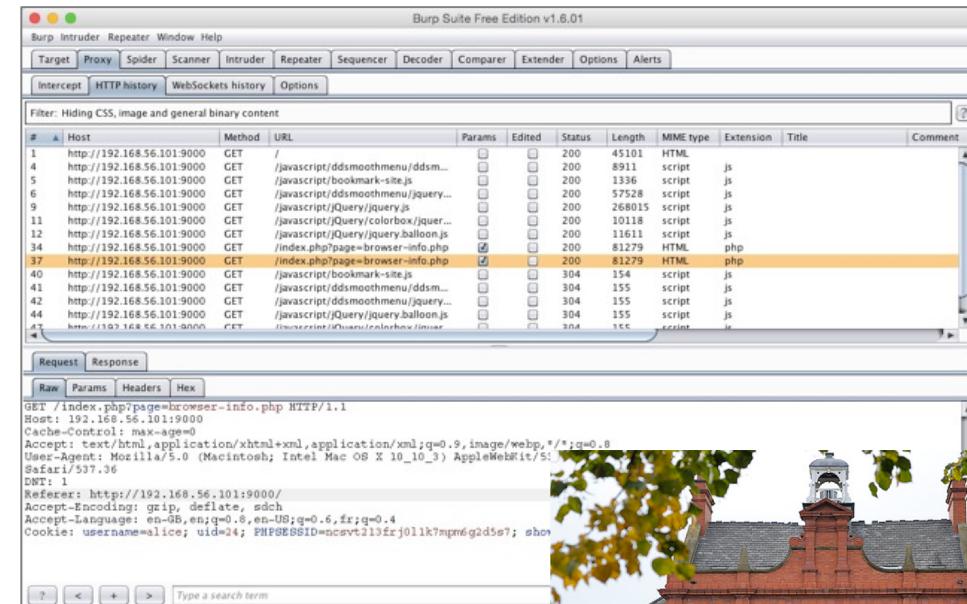
<http://www.dvwa.co.uk/>

<https://github.com/WebGoat/WebGoat/wiki>

<https://github.com/eystsen/pentestlab>

BurpSuite

- Proxy, scanning, pentest tool
- Very capable free version
- Fuller commercial version available
- Inspect traffic, manipulate headers and content, replay, spider, ...
- Made in Knutsford!



The screenshot shows the Burp Suite interface with the following details:

- Toolbar:** Burp, Intruder, Repeater, Window, Help.
- Menu Bar:** Target, Proxy, Spider, Scanner, Intruder, Repeater, Sequencer, Decoder, Comparer, Extender, Options, Alerts.
- Submenu:** Intercept, HTTP history, WebSockets history, Options.
- Table Headers:** #, Host, Method, URL, Params, Edited, Status, Length, MIME type, Extension, Title, Comment.
- Table Data:** A list of 44 network requests, mostly GET requests to various URLs like /index.php?page=... and /javascript/....
- Request/Response Tab:** Request, Response.
- Request Panel:** Raw, Params, Headers, Hex.
- Request Example:**

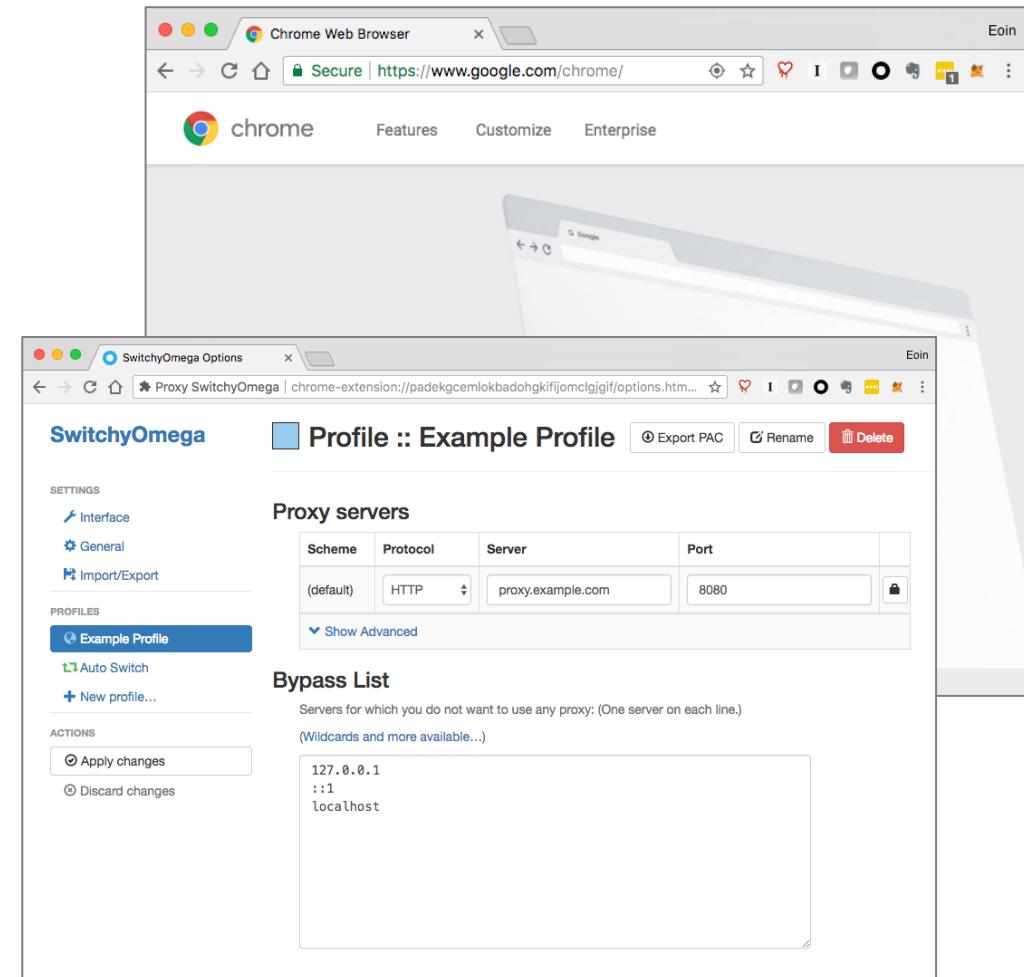
```
GET /index.php?page=bxrver-info.php HTTP/1.1
Host: 192.168.56.101:9000
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_3) AppleWebKit/537.36
Safari/537.36
DNT: 1
Referer: http://192.168.56.101:9000/
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-GB,en;q=0.8,en-US;q=0.6,fr;q=0.4
Cookie: username=alice; uid=24; PHPSESSID=ncsvt13frj011k7mpm6g2d5s7; shor
```
- Search Bar:** Type a search term.



<http://portswigger.net/burp>

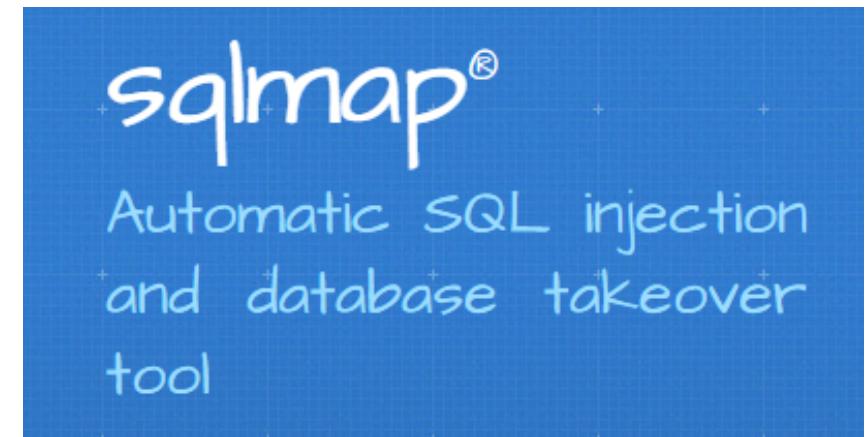
Browser and Proxy Switcher

- Chrome and Switchy Omega or other similar pairing
- Allows easy switching of proxy server to BurpSuite



sqlmap

- Automated SQL injection and database pentesting
- Open source Python command line tool
- Frighteningly effective!

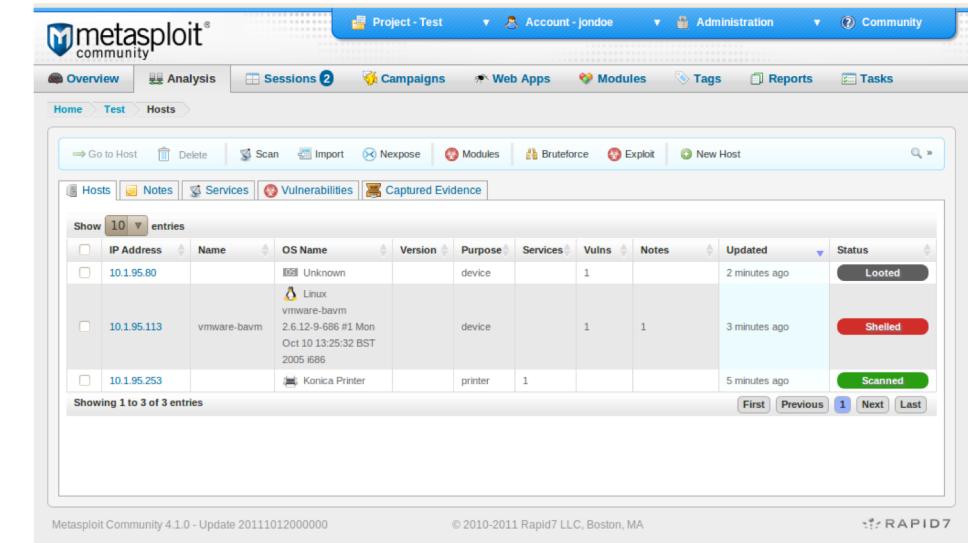


```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch {1.0.5.63#dev} [!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program [*] starting at 17:43:06 [17:43:06] [INFO] testing connection to the target URL [17:43:06] [INFO] heuristics detected web page charset 'ascii' [17:43:06] [INFO] testing if the target URL is stable [17:43:07] [INFO] target URL is stable [17:43:07] [INFO] testing if GET parameter 'id' is dynamic [17:43:07] [INFO] confirming that GET parameter 'id' is dynamic [17:43:07] [INFO] GET parameter 'id' is dynamic [17:43:07] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
```

<http://sqlmap.org>

Metasploit

- The pentester's "standard" tool
- Very wide range of capabilities
- Commercial version available



The screenshot shows the Metasploit Community interface. At the top, there's a navigation bar with links for Overview, Analysis, Sessions, Campaigns, Web Apps, Modules, Tags, Reports, and Tasks. Below that is a sub-navigation bar with Home, Test, and Hosts. The main area is titled 'Hosts' and displays a table of scanned hosts. The columns include IP Address, Name, OS Name, Version, Purpose, Services, Vulns, Notes, Updated, and Status. Three hosts are listed:

IP Address	Name	OS Name	Version	Purpose	Services	Vulns	Notes	Updated	Status
10.1.95.80		Unknown		device		1		2 minutes ago	Looted
10.1.95.113	vmware-bavm	Linux vmware-bavm	2.6.12-9-686 #1 Mon Oct 10 13:25:32 BST 2005 i686	device		1	1	3 minutes ago	Shelled
10.1.95.253		Konica Printer		printer	1			5 minutes ago	Scanned

At the bottom of the interface, it says 'Metasploit Community 4.1.0 - Update 20111012000000' and '© 2010-2011 Rapid7 LLC, Boston, MA'.

<https://www.metasploit.com>

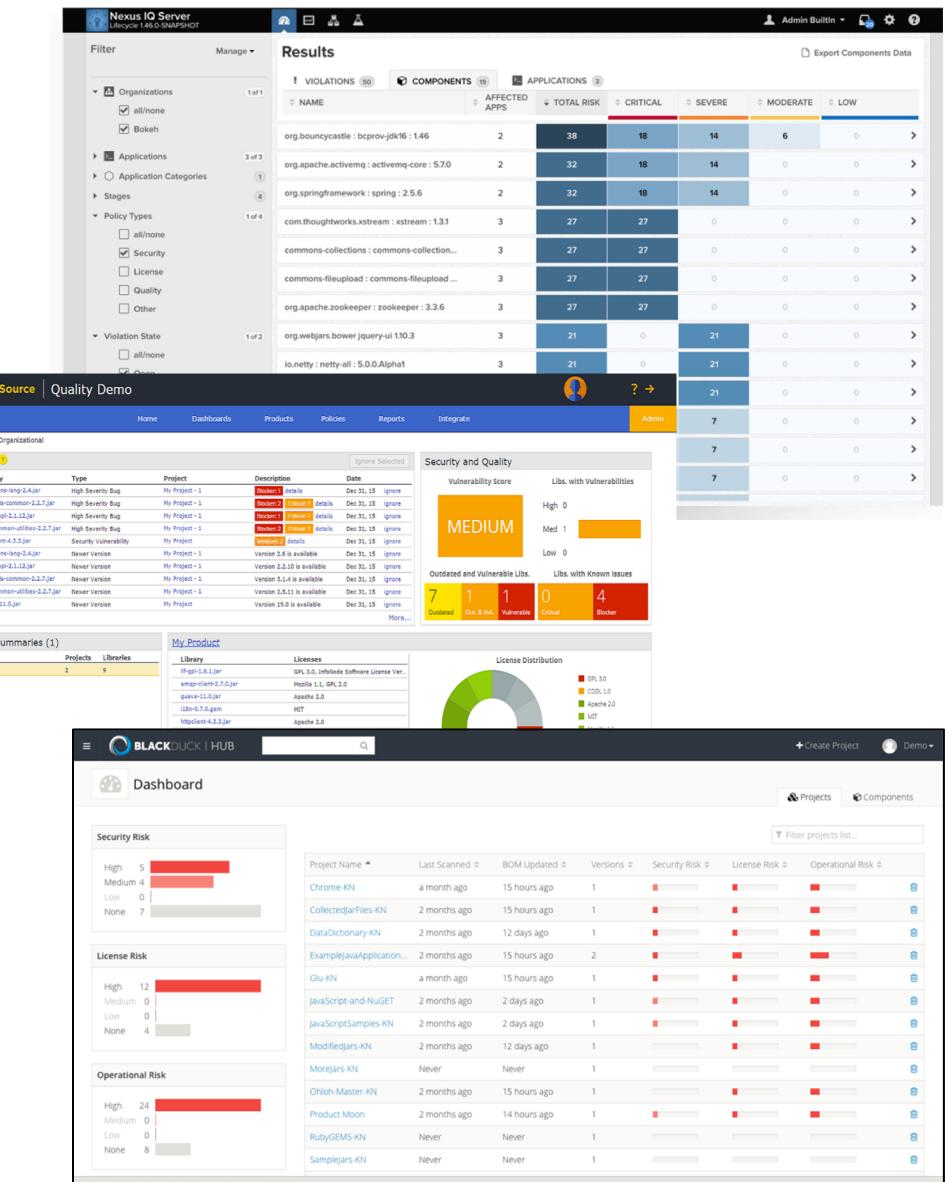
Open Source Scanning

- Example commercial tools for open source security, audit & compliance:
 - BlackDuck
 - Whitesource
 - Sonatype LCM
- Scan builds identifying open source
- Checks for known vulnerabilities
- Alerts and dashboards for monitoring

www.blackduck.com

www.whitesourcesoftware.com

www.sonatype.com/nexus-lifecycle

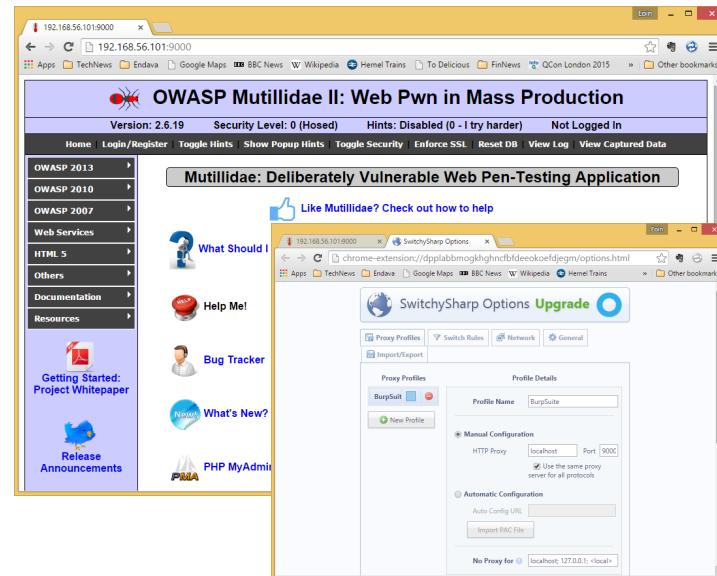


The image displays four screenshots of open source scanning tools:

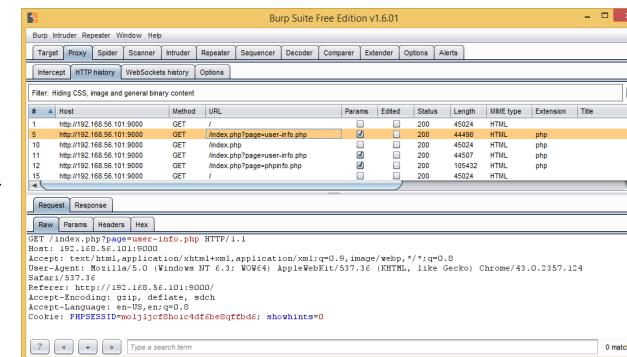
- Nexus IQ Server**: Shows a table of violations across various applications, with columns for Name, Affected Apps, Total Risk, Critical, Severe, Moderate, and Low.
- WhiteSource Quality Demo**: Shows alerts for various libraries, a vulnerability score (Medium), and license distribution.
- BlackDuck Hub**: Shows product summaries, a dashboard with security, license, and operational risks, and a list of projects.
- Sonatype LCM**: Shows a table of violations across various applications, similar to the Nexus IQ Server view.

Demonstrations

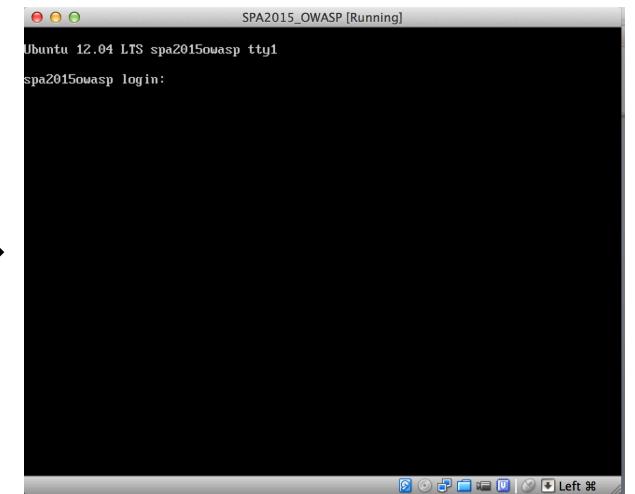
Mutillidae



Browser with proxy plugin



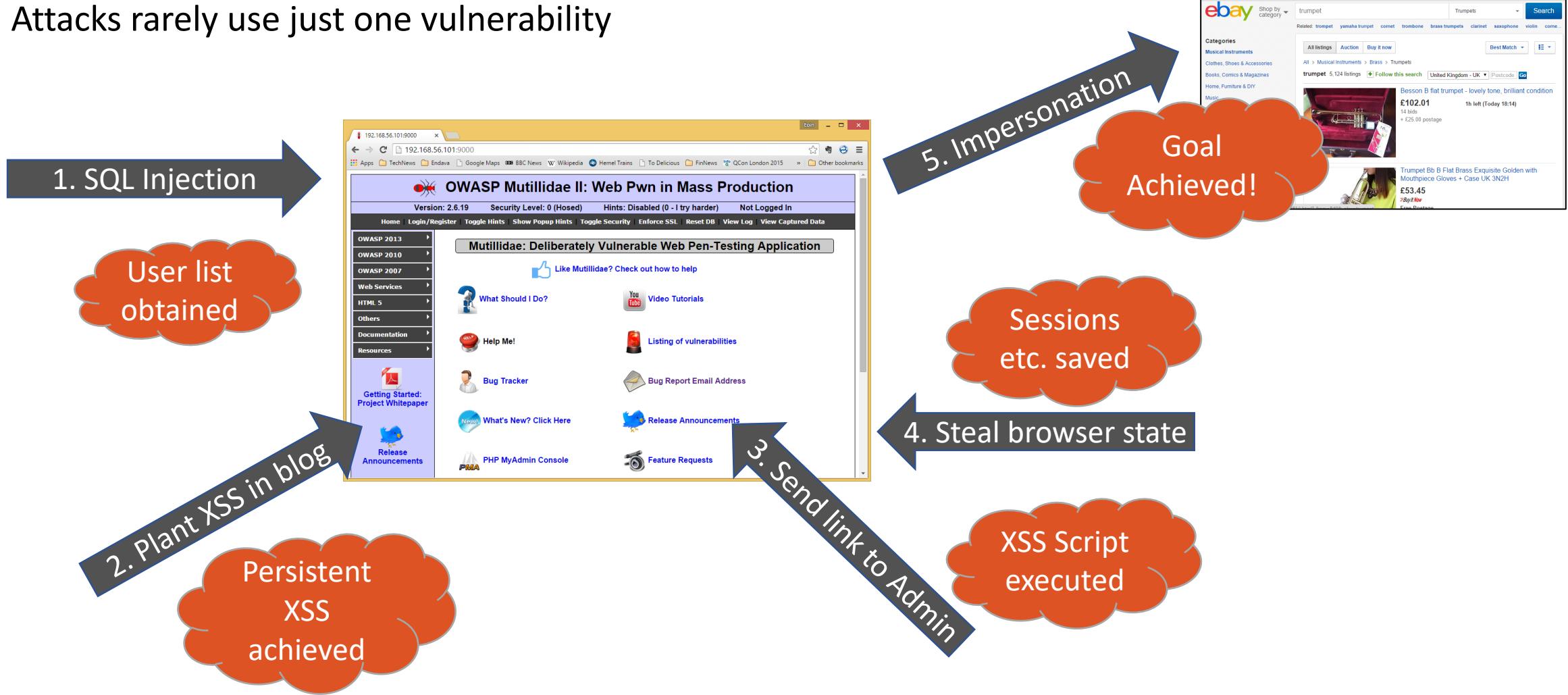
BurpSuite
(proxy)



Mutillidae

An Example Multi-Step Attack - Impersonation

Attacks rarely use just one vulnerability



Practical Work

Practical Work

1. Install a XAMP stack (Apache, PHP, MySQL needed)
 - <https://www.apachefriends.org>
 - <https://bitnami.com/stack/lamp>
2. Install OWASP Mutillidae from Iron Geek
 - <https://sourceforge.net/projects/mutillidae>
3. Install a browser proxy (e.g. SwitchyOmega)
4. Install BurpSuite Community Edition
 - <https://portswigger.net/burp/communitydownload>
5. Git clone <https://github.com/eoinwoods/webappsec>

Defences

Key Web Vulnerability Defences

- Don't trust clients (browsers)
 - Validation, authorisation, ...
- Identify "interpreters", escape inputs, use bind variables, ...
 - Command lines, web pages, database queries, ...
- Protect valuable information at rest and in transit
 - Use encryption judiciously
- Simplicity
 - Verify configuration and correctness
- Standardise and Automate
 - Force consistency, avoid configuration errors

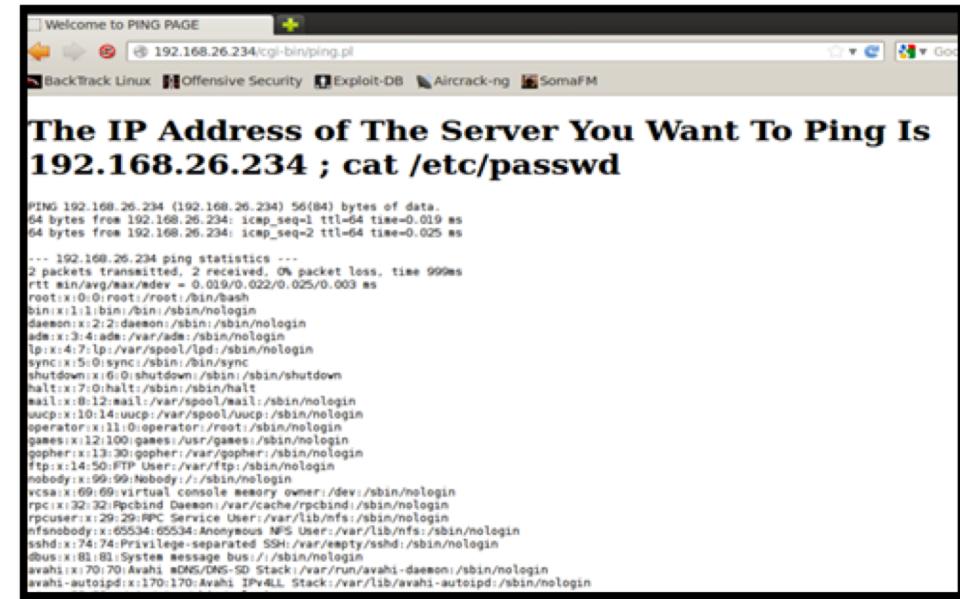
Don't Trust Clients

- Be wary when trusting anything from a browser
 - You don't control it
 - Sophisticated code execution (& injection) platform
 - Output can be manipulated
- Assume or prevent tampering
 - TLS connections to avoid 3rd party interception
 - Short lived sessions
 - Reauthenticate regularly & before sensitive operations
 - Consider multi-factor authentication
 - Use opaque tokens not real object references for params
 - Validate everything



Watch out for injection

- Many pieces of software act as interpreters
 - Browser for HTML and JavaScript
 - Operating system shells – system("mv \$1 \$2")
 - Databases – query languages
 - Configuration files
 - XML parsers
- Assume that someone will work it out!
 - Avoid creating commands using string manipulation
 - Use libraries and bind variables
 - Escape all strings being passed to an “interpreter”
 - Use a third party “escaping” library (e.g. OWASP)
 - Reject excessively long strings (e.g. username > 30 char)



The IP Address of The Server You Want To Ping Is
192.168.26.234 ; cat /etc/passwd

```
PING 192.168.26.234 (192.168.26.234) 56(84) bytes of data.  
64 bytes from 192.168.26.234: icmp_seq=1 ttl=64 time=0.019 ms  
64 bytes from 192.168.26.234: icmp_seq=2 ttl=64 time=0.025 ms  
--- 192.168.26.234 ping statistics ---  
2 packets transmitted, 2 received. 0% packet loss. time 990ms  
rtt min/avg/max/mdev = 0.019/0.022/0.025/0.003 ms  
root:x:0:0:root:/root:/bin/bash  
bin:x:1:1:bin:/bin:/sbin/nologin  
daemon:x:2:2:daemon:/sbin:/sbin/nologin  
adm:x:3:4:adm:/var/adm:/sbin/nologin  
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin  
sync:x:5:0:sync:/sbin:/bin/sync  
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown  
halt:x:7:0:halt:/sbin:/sbin/halt  
operator:x:11:0:operator:/root:/sbin/nologin  
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin  
operator:x:11:0:operator:/root:/sbin/nologin  
games:x:12:100:games:/usr/games:/sbin/nologin  
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin  
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin  
nobody:x:99:99:Nobody:/sbin/nologin  
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin  
rpc:x:32:32:Rpcbind Daemon:/var/cache/rpcbind:/sbin/nologin  
rpcbind:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin  
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin  
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin  
dbs:x:80:80:MySQL database user:/sbin/nologin  
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin  
avahi-autospd:x:170:170:Avahi IPV4/IPv6 Stack:/var/lib/avahi-autospd:/sbin/nologin
```

Protect Valuable Information

- Defence in depth – assume perimeter breach
 - Encrypt messaging as standard
 - Consider database encryption
 - Consider file or filesystem encryption
- However encryption complicates using the data
 - Slows everything down
 - Can you query while encrypted? (Homomorphic encryption?)
 - Message routing on sensitive fields (in headers)
 - Managing and rotating the keys
 - What about restore on disaster recovery?



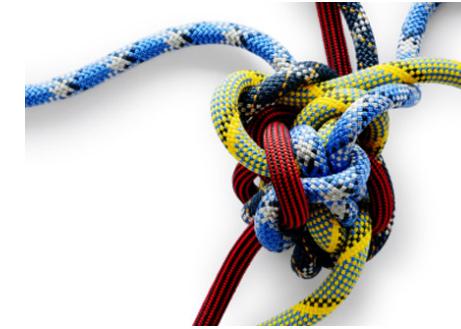
<http://getacoder.com>



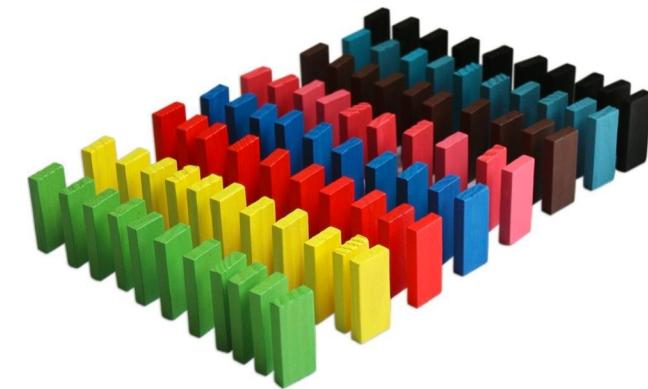
<http://slate.com>

Simplicity & Standardisation

- Complexity is the enemy of security
 - “*You can’t secure what you don’t understand*” - Schneier
 - Special cases will be forgotten
- Simplify, Standardise and Automate
 - Simpler things are easier to check and secure
 - Standardising an approach means there are no special cases to forget to handle
 - Automation eliminates human inconsistencies from the process so avoiding a type of risk



<http://innovationmanagement.se/>



<https://www.aliexpress.com>

Summary

Summary

- Much of the technology we use is inherently insecure
 - Mitigation needs to be part of application development
- Attacking systems is becoming industrialised
 - Digital transformation is providing more valuable, insecure targets
- Fundamental attack vectors appear again and again
 - Injection, interception, page manipulation, validation, configuration, ...
- Most real attacks exploit a series of vulnerabilities
 - Each vulnerability may not look serious, the combination is
- Most mitigations not difficult but need to be applied consistently
 - ... and may conflict with other desirable qualities

Thank You

Eoin Woods
Endava
[@eoinwoodz](https://twitter.com/eoinwoodz)
eoin.woods@endava.com

