Web Security: Part I

Emilio Coppa
coppa@diag.uniroma1.it

Sapienza University of Rome

Credits

These slides are based on teaching material originally created by:

- Marco Squarcina (marco.squarcina@tuwien.ac.at), S&P Group, TU WIEN
- Mauro Tempesta (mauro.tempesta@tuwien.ac.at), S&P Group, TU WIEN
- Leonardo Querzoni (querzoni@diag.uniroma1.it), Sapienza University of Rome
- Fabrizio D'Amore (damore@diag.uniroma1.it), Sapienza University of Rome

The Cost of Vulnerabilities

DEFINITIONS

- A vulnerability is a weakness which allows an attacker to reduce system's information assurance.
- A vulnerability is the intersection of three elements:
  - a system susceptibility or flaw
  - attacker access to the flaw
  - attacker capability to exploit the flaw

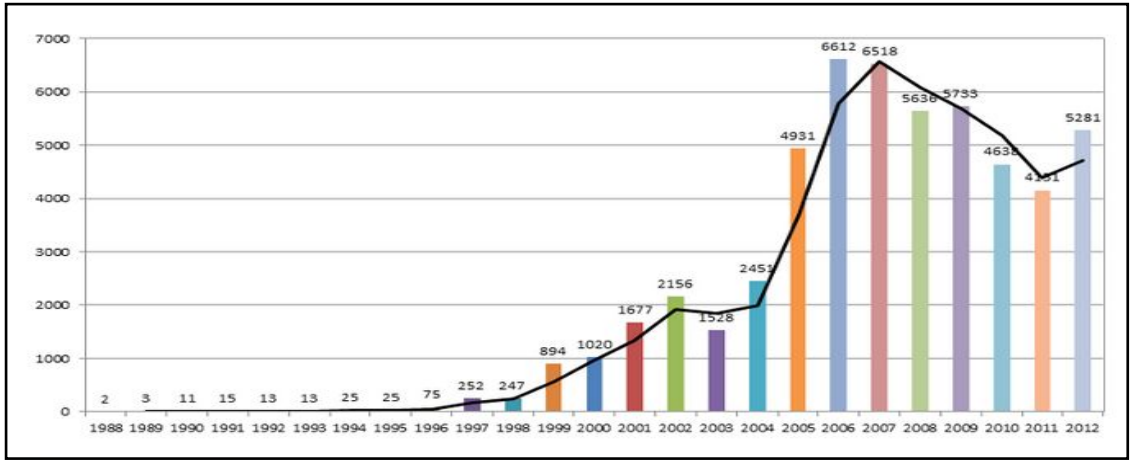
DEFINITIONS

- Causes:
  - Bugs
  - Design defects
  - Misconfigurations
  - Aging
- Fostering factors
  - System complexity
  - Connectivity
  - Incompetence

MOTIVATIONS

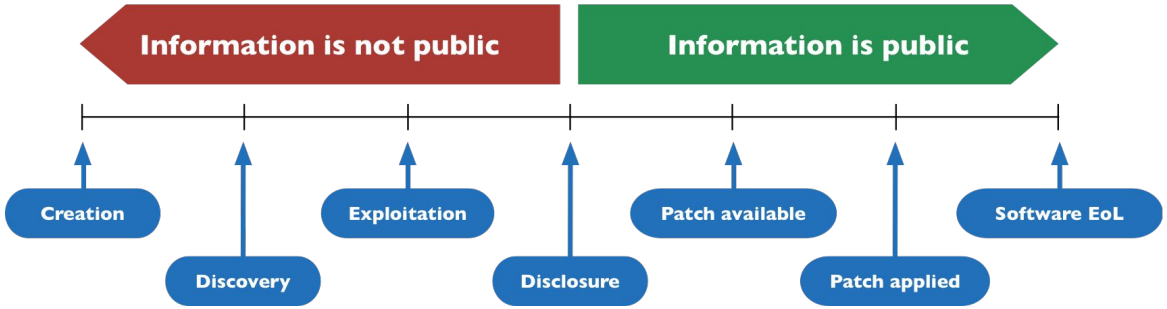
- Vulnerability in software is one of the major reasons for insecurity
  - 20 flaws per thousand lines of code (Dacey 2003)
  - Steady increase in vulnerability exploitations
- Fully secure software is unlikely
- 95% of breaches could be prevented by keeping systems up-to-date with patches (Dacey 2003)

Source: 25 Years of Vulnerabilities: 1988-2012,

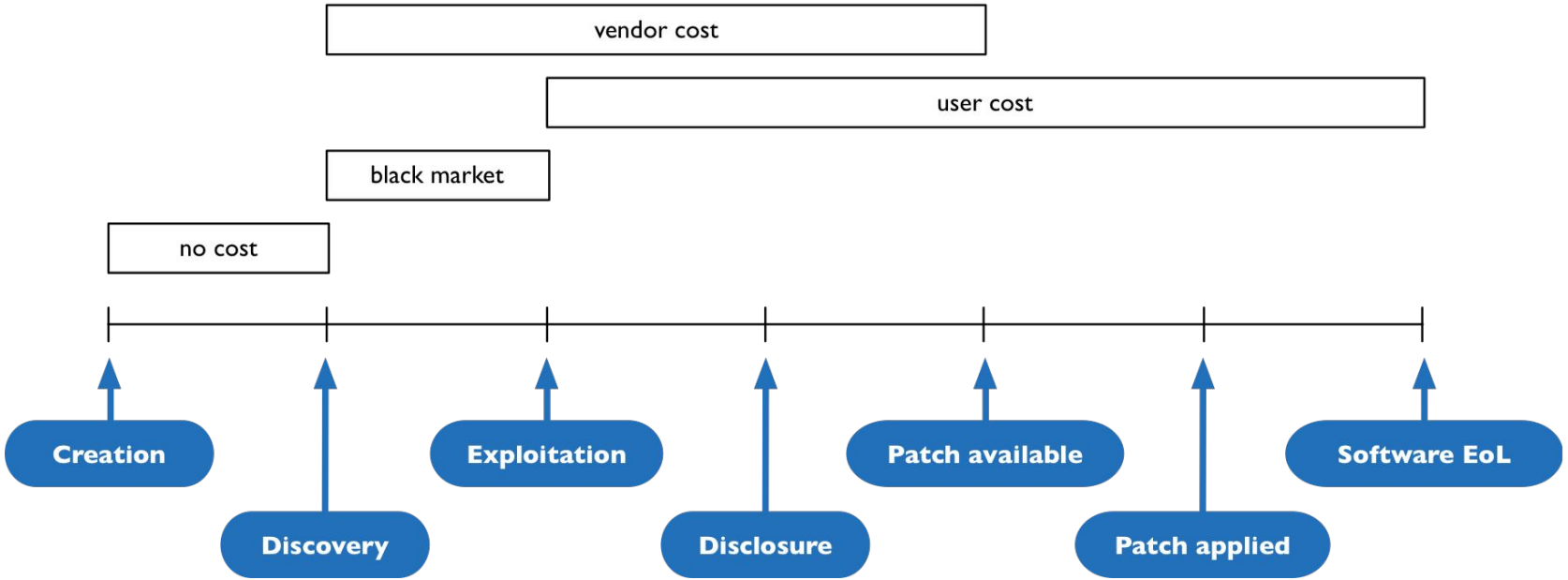


VULNERABILITY LIFE CYCLE

- **CREATION**
- **DISCOVERY**
  - Malicious users
  - Benign users (final users, security firms, researchers, ...)
- **EXPLOITATION**
- **DISCLOSURE**
  - Keep it secret
  - Publicly disclose
  - Sell
- **PATCH**



VULNERABILITY LIFE CYCLE: WHO PAYS THE COST?



Source: S. Frei, D. Schatzmann, B. Plattner and B. Trammell, Modelling the Security Ecosystem - The Dynamics of (In)Security,

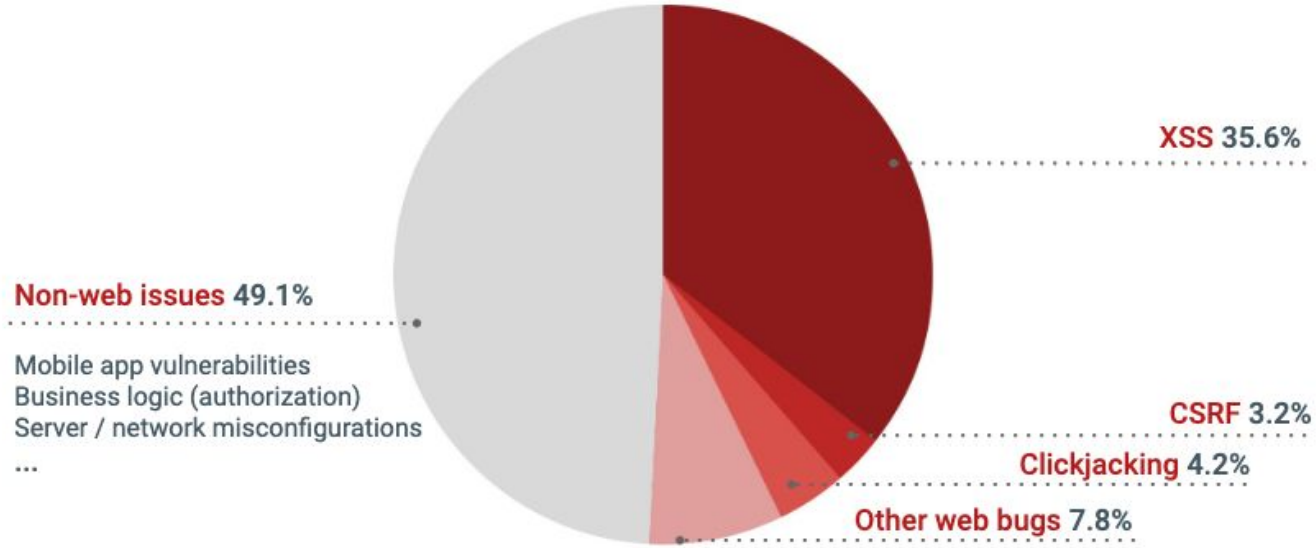
RESPONSIBLE DISCLOSURE

- Forget about malicious users
- What process should a responsible user follow to disclose the vulnerability?
  - No consensus
  - Different vendors provides different guidelines for disclosure
  - CERT (Computer Emergency Response Team) allows a 45-days grace period. OIS allows a 30-days grace period
  - Security firms follow their internal guidelines

DISCLOSURE POLICY EFFECTS

- **Full Vendor Disclosure**
  - Promotes secrecy
  - Gives full control of the process to the vendor
- **Immediate Public Disclosure**
  - Promotes transparency
  - Gives the vendor a strong incentive to fix the problem
  - Allows vulnerable users to take intermediate measures
  - Immediate exposure to risks
- **Hybrid Disclosure**
  - Promotes both secrecy and transparency

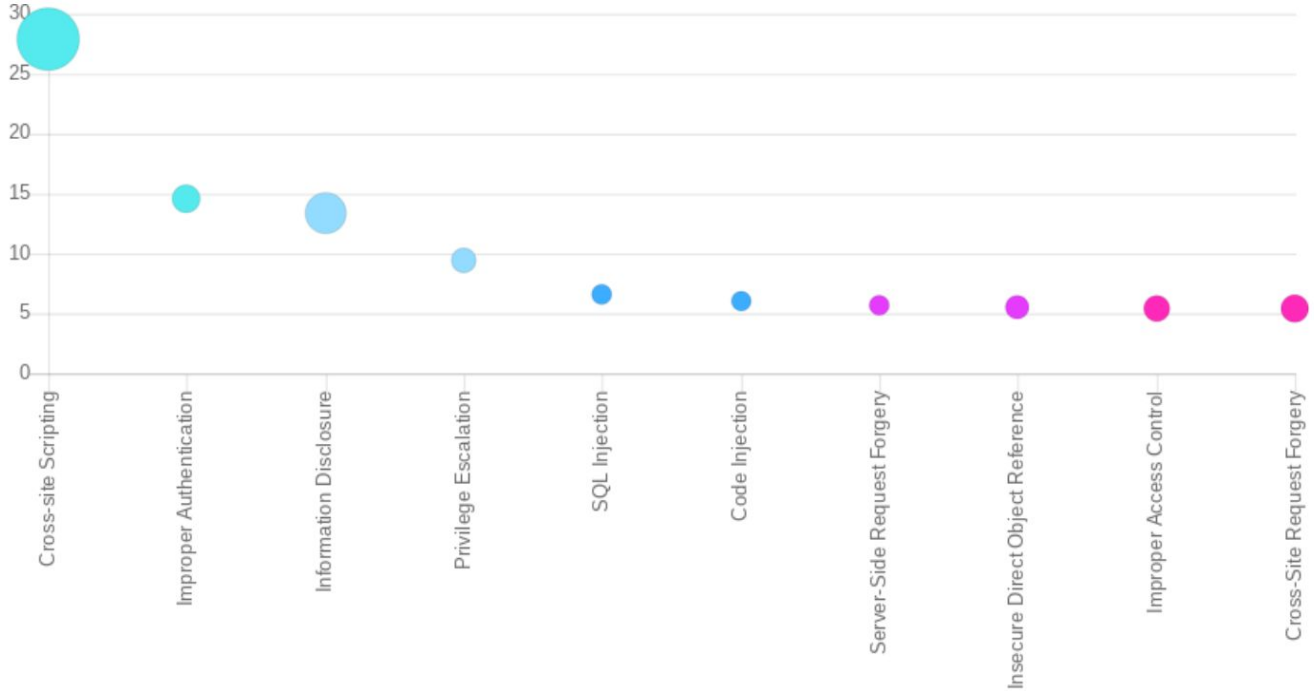
Google VRP, 2018



- Total Google Vulnerability Reward Program payouts, covering regular user-facing products (including web applications)
- 3.4 million $ of total rewards in 2018

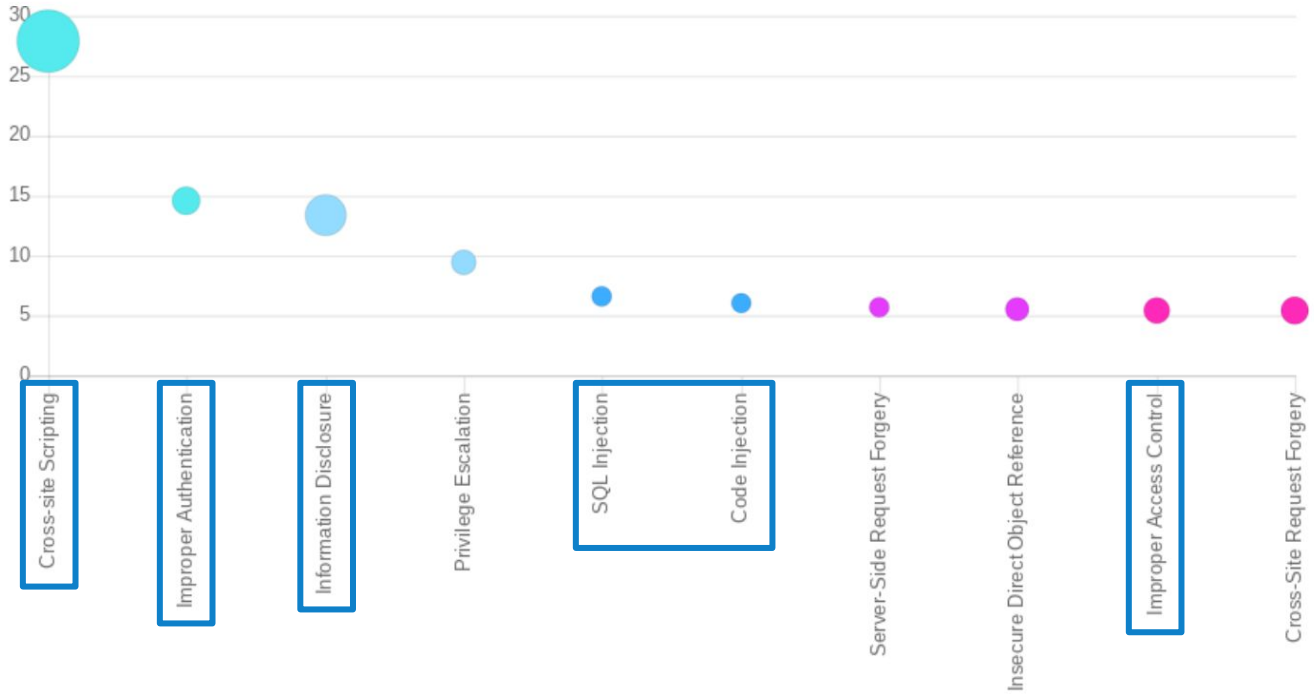
Source: https://www.arturjanc.com/usenix2019/

HackerOne Top 10, 2018



➔ **Bubble size represents volume of reports, Y-axis represents that Weakness Types percent of the total bounties paid to all Top 10 combined**

HackerOne Top 10, 2018



- ➔ Bubble size represents volume of reports, Y-axis represents that Weakness Types percent of the total bounties paid to all Top 10 combined
- ➔ Only 6 vulnerability types are on the OWASP Top 10, XXE is #15

Eligible Research

We acquire zero-day exploits and innovative security research related to the following products:

Operating Systems

Remote code execution or local privilege escalation, or VM escape:

- Microsoft Windows
- Linux / BSD
- Apple macOS
- ESXi / HyperV

Web Browsers

Remote code execution, or sandbox bypass/escape, or both:

- Google Chrome
- Microsoft Edge
- Mozilla Firefox
- Apple Safari

Clients / Files

Remote code execution or information disclosure:

- MS Office (Word/Excel)
- MS Outlook / Mail App
- Mozilla Thunderbird
- Archivers (7-Zip/WinRAR/Tar)

Mobiles / Smartphones

Remote code execution, or privilege escalation, or any other research:

- Apple iOS
- Apple watchOS
- Android
- Windows Mobile

Web Servers

Remote code execution or information disclosure:

- Apache HTTP Server
- Microsoft IIS Server
- nginx web server
- PHP / ASP
- OpenSSL / mod_ssl

Email Servers

Remote code execution or information disclosure:

- MS Exchange
- Dovecot
- Postfix
- Exim
- Sendmail

Web Apps / Panels

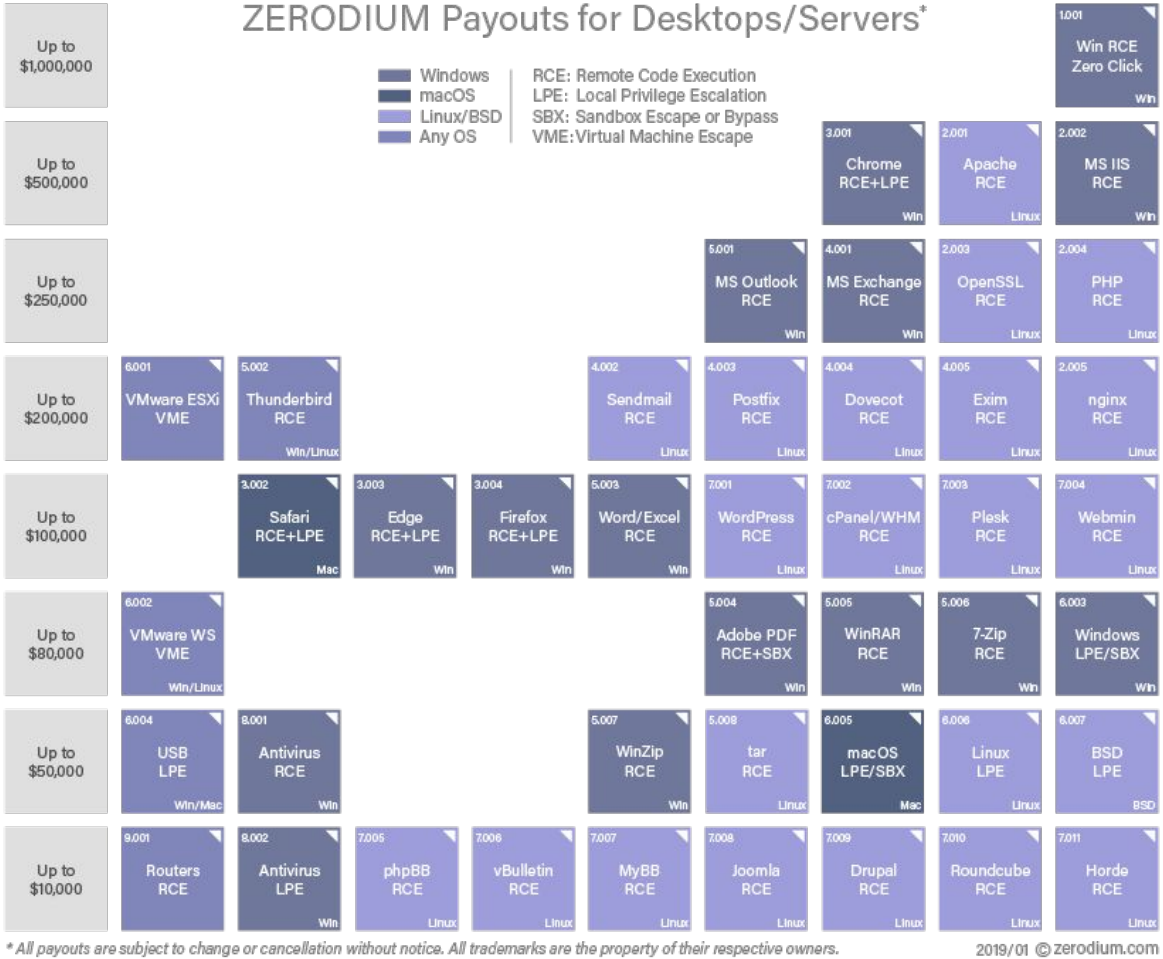
Remote code execution or information disclosure:

- cPanel / Plesk / Webmin
- WP / Joomla / Drupal
- vBulletin / MyBB / phpBB
- IPS Suite / IP.Board
- Roundcube / Horde

Research / Techniques

Research, exploits or new techniques related to:

- WiFi / Baseband RCE
- Routers / IoT RCE
- AntiVirus RCE/LPE
- Tor De-anonymization
- Mitigations Bypass


ZERODIUM Payouts for Desktops/Servers*
Windows
macOS
Linux/BSD
Any OS
RCE: Remote Code Execution
LPE: Local Privilege Escalation
SBX: Sandbox Escape or Bypass
VME: Virtual Machine Escape
Up to $1,000,000
1.001 Win RCE Zero Click Win
Up to $500,000
3.001 Chrome RCE+LPE Win
2.001 Apache RCE Linux
2.002 MS IIS RCE Win
Up to $250,000
5.001 MS Outlook RCE Win
4.001 MS Exchange RCE Win
2.003 OpenSSL RCE Linux
2.004 PHP RCE Linux
Up to $200,000
6.001 VMware ESXi VME
5.002 Thunderbird RCE Win/Linux
4.002 Sendmail RCE Linux
4.003 Postfix RCE Linux
4.004 Dovecot RCE Linux
4.005 Exim RCE Linux
2.005 nginx RCE Linux
Up to $100,000
3.002 Safari RCE+LPE Mac
3.003 Edge RCE+LPE Win
3.004 Firefox RCE+LPE Win
5.003 Word/Excel RCE Win
7.001 WordPress RCE Linux
7.002 cPanel/WHM RCE Linux
7.003 Plesk RCE Linux
7.004 Webmin RCE Linux
Up to $80,000
6.002 VMware WS VME Win/Linux
5.004 Adobe PDF RCE+SBX Win
5.005 WinRAR RCE Win
5.006 7-Zip RCE Win
6.003 Windows LPE/SBX Win
Up to $50,000
6.004 USB LPE Win/Mac
8.001 Antivirus RCE Win
5.007 WinZip RCE Win
5.008 tar RCE Linux
6.005 macOS LPE/SBX Mac
6.006 Linux LPE Linux
6.007 BSD LPE BSD
Up to $10,000
9.001 Routers RCE
8.002 Antivirus LPE Win
7.005 phpBB RCE Linux
7.006 vBulletin RCE Linux
7.007 MyBB RCE Linux
7.008 Joomla RCE Linux
7.009 Drupal RCE Linux
7.010 Roundcube RCE Linux
7.011 Horde RCE Linux
* All payouts are subject to change or cancellation without notice. All trademarks are the property of their respective owners.
2019/01 © zerodium.com

Example: ProxyLogon (2021)

- **ProxyLogon** is a recent vulnerability found on Microsoft Exchange Servers
  - discovered by Orange Tsai (DEVCORE Research Team)

- Prerequisites for the attack:
  - an open HTTPS port (443), i.e., the server is running!

- Outcome:
  - an unauthenticated attacker can execute arbitrary commands on the system (spawn a remote shell, leak all files, ...)

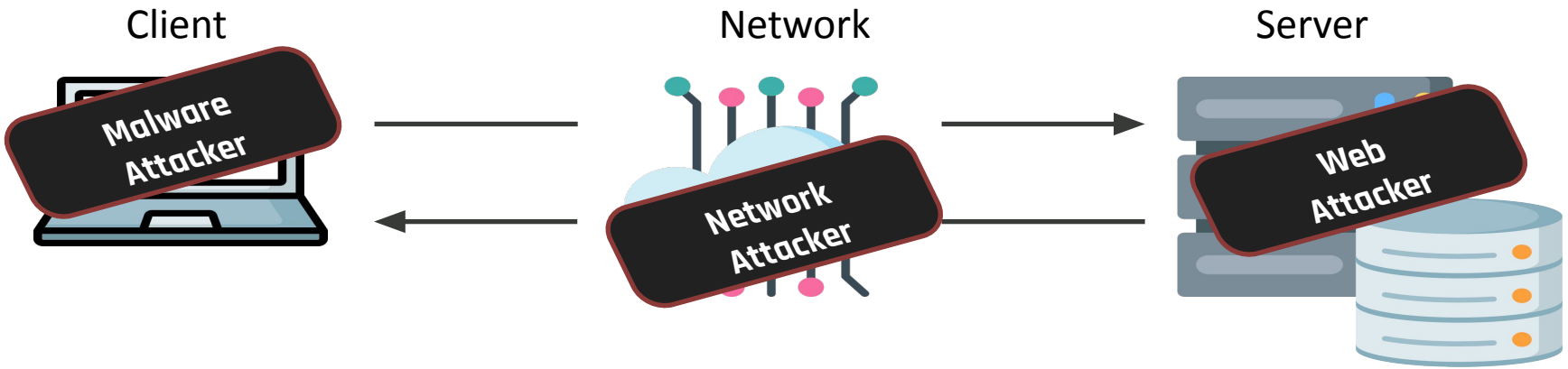
Example: ProxyLogon (2)

Vulnerability Disclosure Timeline:

October 01, 2020	DEVCORE started reviewing the security on Microsoft Exchange Server
December 10, 2020	DEVCORE discovered the first pre-auth proxy bug (**CVE-2021-26855**)
December 27, 2020	DEVCORE escalated the first bug to an authentication bypass to become admin
December 30, 2020	DEVCORE discovered the second post-auth arbitrary-file-write bug (**CVE-2021-27065**)
December 31, 2020	DEVCORE chained all bugs together to a workable pre-auth RCE exploit
January 05, 2021	DEVCORE sent (18:41 GMT+8) the advisory and exploit to Microsoft through the MSRC portal directly
January 06, 2021	MSRC acknowledged the pre-auth proxy bug (MSRC case 62899)
January 06, 2021	MSRC acknowledged the post-auth arbitrary-file-write bug (MSRC case 63835)
January 08, 2021	MSRC confirmed the reported behavior
January 11, 2021	DEVCORE attached a 120-days public disclosure deadline to MSRC and checked for bug collision
January 12, 2021	MSRC flagged the intended deadline and confirmed no collision at that time
February 02, 2021	DEVCORE checked for the update
February 02, 2021	MSRC replied "they are splitting up different aspects for review individually and got at least one fix which should meet our deadline"
February 12, 2021	MSRC asked the title for acknowledgements and whether we will publish a blog
February 13, 2021	DEVCORE confirmed to publish a blog and said will postpone the technique details for two weeks, and will publish an easy-to-understand advisory (without technique details) instead
February 18, 2021	DEVCORE provided the advisory draft to MSRC and asked for the patch date
February 18, 2021	MSRC pointed out a minor typo in our draft and confirmed the patch date is 3/9
February 27, 2021	MSRC said they are almost set for release and wanted to ask if we're fine with being mentioned in their advisory
February 28, 2021	DEVCORE agreed to be mentioned in their advisory
March 03, 2021	MSRC said they are likely going to be pushing out their blog earlier than expected and won't have time to do an overview of the blog
March 03, 2021	MSRC published the patch and advisory and acknowledged DEVCORE officially
March 03, 2021	DEVCORE has launched an initial investigation after informed of **active exploitation advisory from Volexity**
March 04, 2021	DEVCORE has confirmed the in-the-wild exploit was the same one reported to MSRC
March 05, 2021	DEVCORE hasn't found concern in the investigation
March 08, 2021	As more cybersecurity companies have found the signs of intrusion at Microsoft Exchange Server from their client environment, DEVCORE later learned that HAFNIUM was using ProxyLogon exploit during the attack in late February from **Unit 42**, **Rapid 7**, and **CrowdStrike**.
Auguest 06, 2021	DEVCORE has published the **technique details and the story afterward**

The Cursed Web

Types of Attackers



Web Attacker

Different scenarios:

- Attacker controls the domain attacker.com, for which it can acquire a valid TLS certificate. The user visits attacker.com (e.g., because of phishing, search results, click-hijacking, ...)
- Variation “gadget attacker”: an iframe with malicious content included in an otherwise honest webpage visited by the user
- Variation “related-domain attacker”: the attacker controls a related-domain of the target website, e.g., attacker.example.com
- The attacker is a user of a website. The target could be the website or other users. The website should be vulnerable to some attacks.

Network and Malware Attackers

- Network attacker
  - Passive: wireless eavesdropper
  - Active: evil Wi-Fi router, DNS poisoning
- Malware attacker
  - Malicious code executes directly on victim's computer
  - software bugs, malware, ...

The Cursed Web

- Delusive simplicity for creating web apps
- Lack of security awareness
- Time- & resource limits during development
- Rapid increase in code complexity

- **Company's security focus shifts towards web**
  - Security perimeter moves from the network to the application layer
  - Web apps intentionally expose functionality to the Internet while being connected to internal servers (e.g., databases)
  - Blurred lines between mobile and web apps
    - Web content tightly integrated into mobile apps
    - Unintentional exposure of backend web APIs

Fundamental Problems of the Web Ecosystem

- **Network protocol issues**
  - MiTM (SSL Strip), mixed-content sites
  - Cookies leaked over HTTP…
- **Mixing code and data**
  - SQL injections
  - Cross-site scripting (XSS)
- **Unrestricted attack surface**
  - Cross-site request forgery (CSRF), Cross-site script inclusion (XSSI)
  - Clickjacking, Cross-site search (XS-Search)
- **Legacy design**
  - Unsafe legacy APIs, Dangerous web features
  - Poor security boundaries in cookie design/adoption

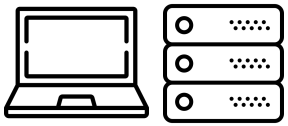
➔ **Partial list of attacks/issues caused by these fundamental problems**

Countermeasures

Client

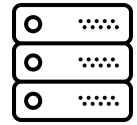
- XSS Filters
- Sandboxes
- Site Isolation

Hybrid



- HSTS
- CSP
- CORS
- Fetch Metadata
- Trusted Types
- Cookie policies
- ...

Server



- Prepared statements
- Server-side filtering
- Web Application Firewalls
- CSRF tokenization

Countermeasures

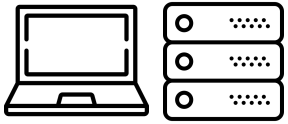
Client

Found to introduce vulns and removed from browsers [URL]

- ~~XSS Filters~~
- Sandboxes
- Site Isolation

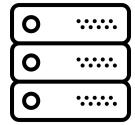
Defense-in-depth mechanisms

Hybrid



- HSTS
- CSP
- CORS
- Fetch Metadata
- Trusted Types
- Cookie policies
- ...

Server



- Prepared statements
- Server-side filtering
- Web Application Firewalls
- CSRF tokenization

Countermeasures

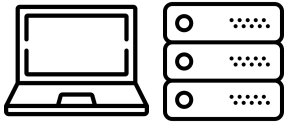
Client

Found to introduce vulns and removed from browsers [URL]

- ~~XSS Filters~~
- Sandboxes
- Site Isolation

Defense-in-depth mechanisms

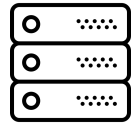
Hybrid



- HSTS
- CSP
- CORS
- Fetch Metadata
- Trusted Types
- Cookie policies
- ...

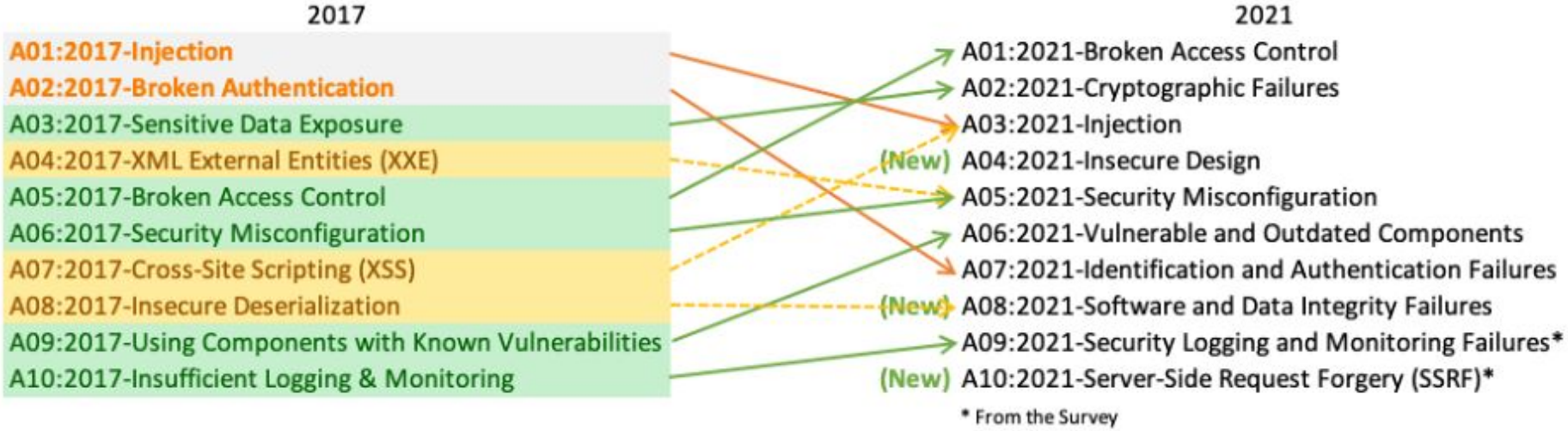
Policy-based mechanisms

Server



- Prepared statements
- Server-side filtering
- Web Application Firewalls
- CSRF tokenization

Most Critical Web Security Risks



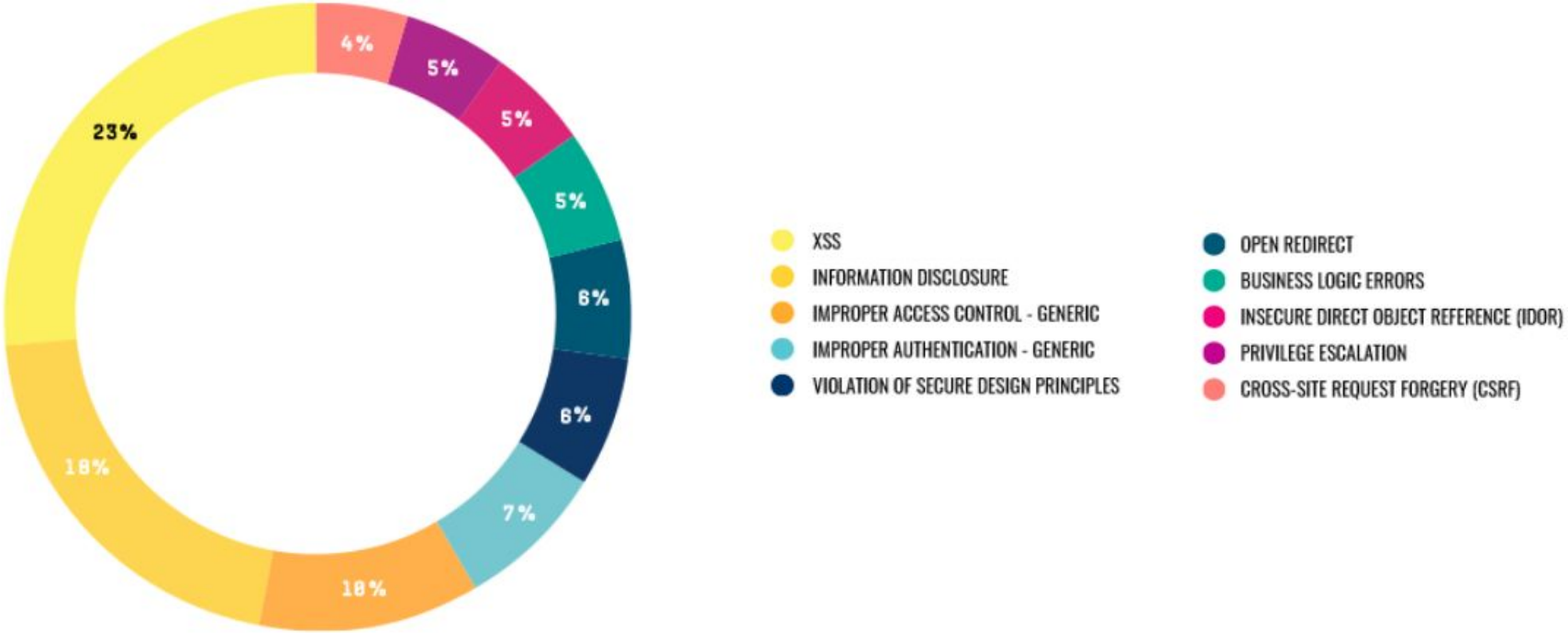
OWASP
Open Web Application Security Project

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

OWASP 2017 top 10: [PDF]

OWASP Cheat sheet: [URL]

Prevalence of Vulnerabilities (HackerOne)

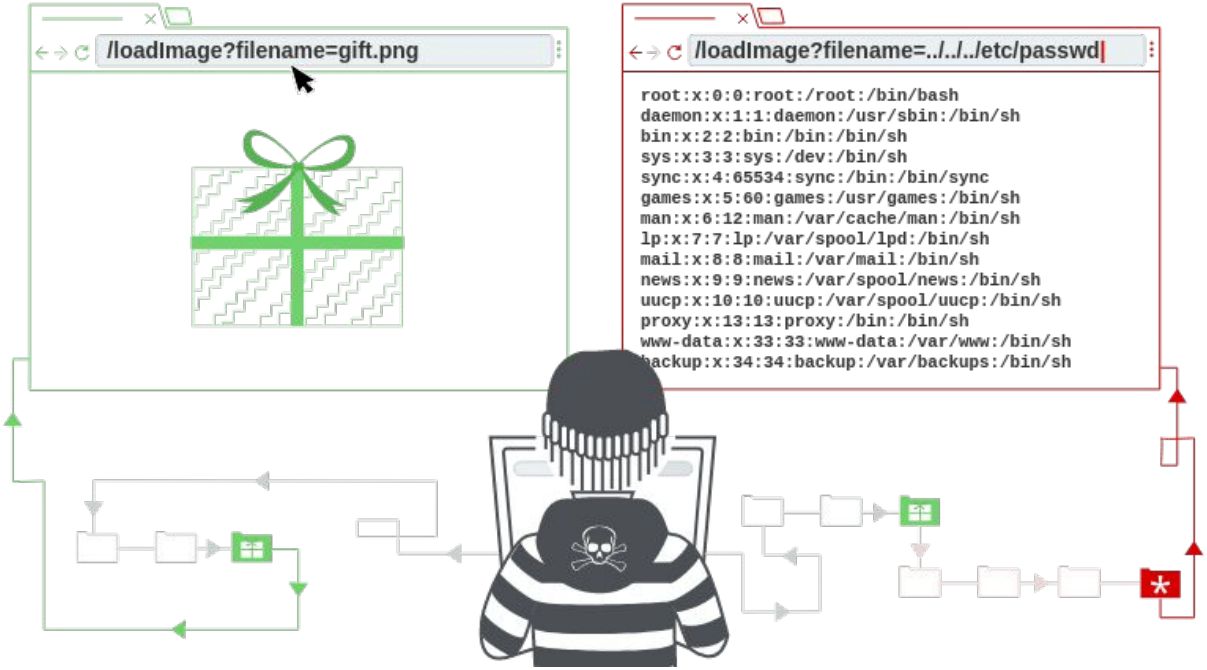


Source: https://www.hackerone.com/hacker-powered-security-report

Threats & Defenses

Path Traversal

Path Traversal in a Nutshell



Source: https://portswigger.net/web-security/file-path-traversal
OWASP > A01:2021 – Broken Access Control **>** Path Traversal

Example of a Path Traversal Attack

show.php

```php
<?php
  echo file_get_contents("pages/" . $_GET["page"]);
?>
```

- Consider a web server whose webroot is /var/www/html (standard location on Linux servers)
  - The webroot is the topmost directory in which the files of a website are stored
  - Files outside the webroot are not accessible
- The webroot contains the file show.php above and a directory pages containing some text files that can be included by the PHP script

Intended Usage

Attack

- Root cause of the problem: The user input provided through via page variable is not (correctly) filtered!
- Attacker can “climb up” multiple levels in the directory hierarchy (and exit the webroot) by using ../ (Linux) or ..\ (Windows) and get access to any file on the web server (sensitive operating system files, TLS keys, etc.)

GET /show.php?page=../../../etc/passwd HTTP/2
Host: example.com

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin

example.com

Preventing Path Traversals

- Ideally: don't use user controlled input as (part of) filenames
- In the real world: Validate all user inputs!
  - If possible, allow only a (static) list of file paths
  - Otherwise, compute the canonical path of the required file and ensure it is not outside the webroot (or the expected directory)

show.php

```php
<?php
  $pdir = "/var/www/html/pages/";
  $file = realpath($pdir . $_GET["file"]);

  if ($file !== false && strncmp($file, $pdir, strlen($pdir)) === 0) {
    echo file_get_contents($file);
  } else {
    echo "Error: invalid input";
  }
?>
```

Preventing Path Traversals - Defense in Depth

- Reduced privileges of web server
  - Restrict access of web server to its own directory
  - Use sandbox environment (chroot jail, SELinux, containers,...) to enforce boundary between web server and the OS

- This is a so-called defense-in-depth mechanism: it is a good idea to deploy it, but it should not be the only adopted defense mechanism!

Important directories

- **Document Root**

  folder in Web server designated to contain Web pages synonymous: start directory, home directory, web publishing directory, remote root etc. typical names of document root: htdocs, httpdocs, html, public_html, web, www etc.

- **Server Root**

  Contains logs & configurations. A few scripts put here their working directory

File permissions

- be aware of permissions given to directories
  - document root, containing HTML documents
  - server root, containing log & configuration files; often CGI scripts run here
  - the Common Gateway Interface (CGI) is a standard (RFC3875) that defines how Web server software can delegate the generation of Web pages to a console application. Such applications are known as CGI scripts; they can be written in any programming language, although scripting languages are often used

- good idea: purposely define user and group for the web server
  - e.g., www & wwwgroup
  - HTML authors should be added to wwwgroup
  - the www should have access only to the right files

Other configurations

Web servers may have additional capabilities, that can increase the risk

- automatic directory listing: an attacker can see the content of directories… what it we have left some sensitive data (e.g., our editor has left a temp copy of our PHP file?), symbolic links, development logs, source code control directories.
- symbolic link following: it may allow an attacker to access unexpected directories
- server side include (SSI): ".shtml" dynamic pages in the 90s… directives for including other files and executing commands. Still enabled somewhere.
- user maintained directory: Still used in several organization, e.g., example.com/~user