Software Security

Intro

Alessandra Gorla

Vulnerabilities

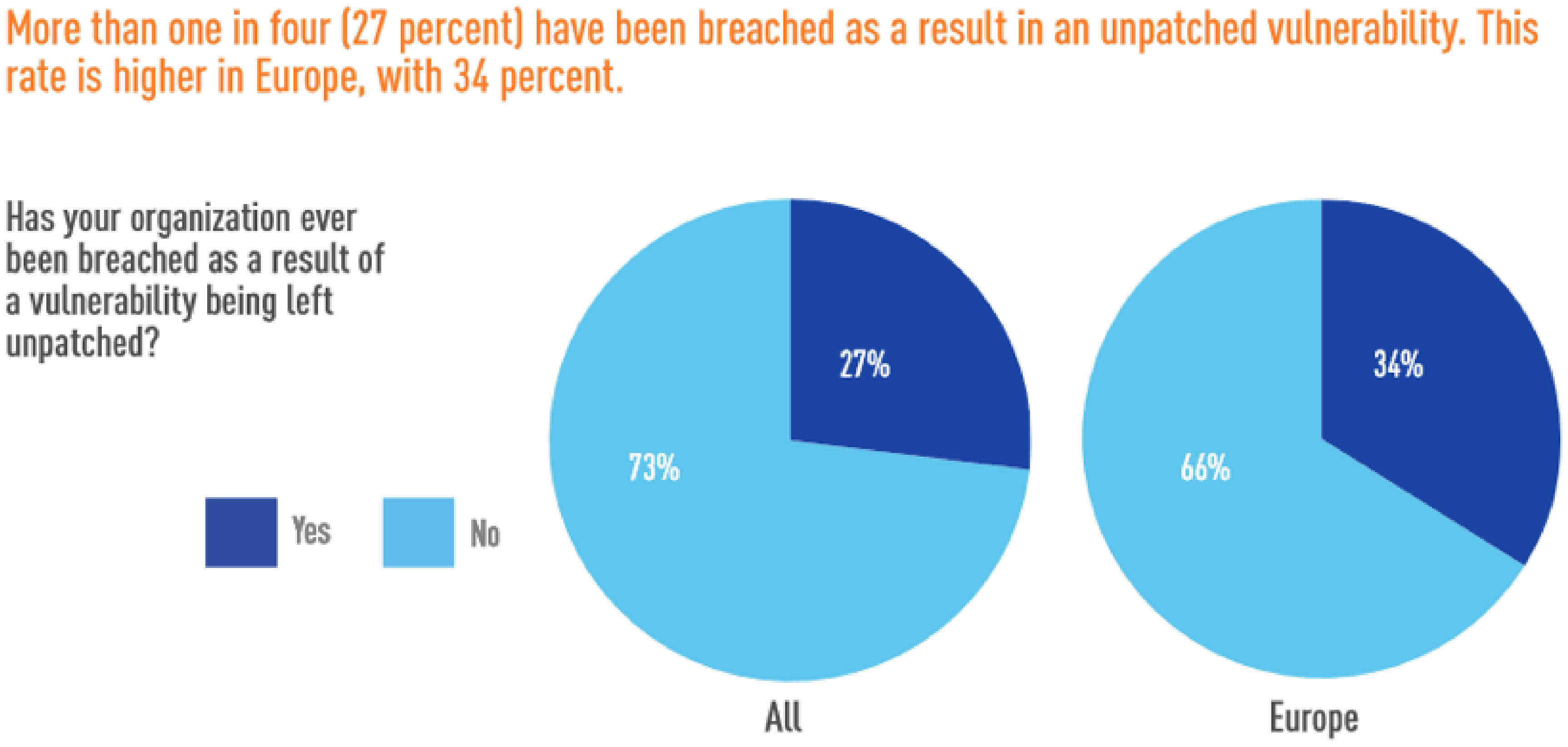
- **Vulnerabilities** are flaws in a computer system or network that weaken its overall security
- Can affect: Software, Hardware, Protocols
- We will focus on **software vulnerabilities** (or security bugs)
  - Not all SW bugs are vulnerabilities, security implications needed
- Vulnerabilities can be **exploited** by an attacker
  - To perform unexpected/unauthorized actions that compromise the confidentiality/integrity/availability of resources
  - **Exploitable vulnerability**: A vulnerability that can be exploited
- Vulnerability **risk:** probability x potential impact if exploited

Why are Bugs / Vulnerabilities Introduced?

- Growing software complexity
- Fast development and update cycles
- Changing requirements
- Human mistakes
- Copy-paste
- Unclear documentation
- Large number of dependencies (e.g., third party libraries)
- Unsafe programming languages
- …

Vulnerability Impact: Breaches

Privacy laws require data controllers to reduce the risk of user data



More than one in four (27 percent) have been breached as a result in an unpatched vulnerability. This rate is higher in Europe, with 34 percent.

Has your organization ever been breached as a result of a vulnerability being left unpatched?

https://www.tripwire.com/state-of-security/unpatched-vulnerabilities-breaches

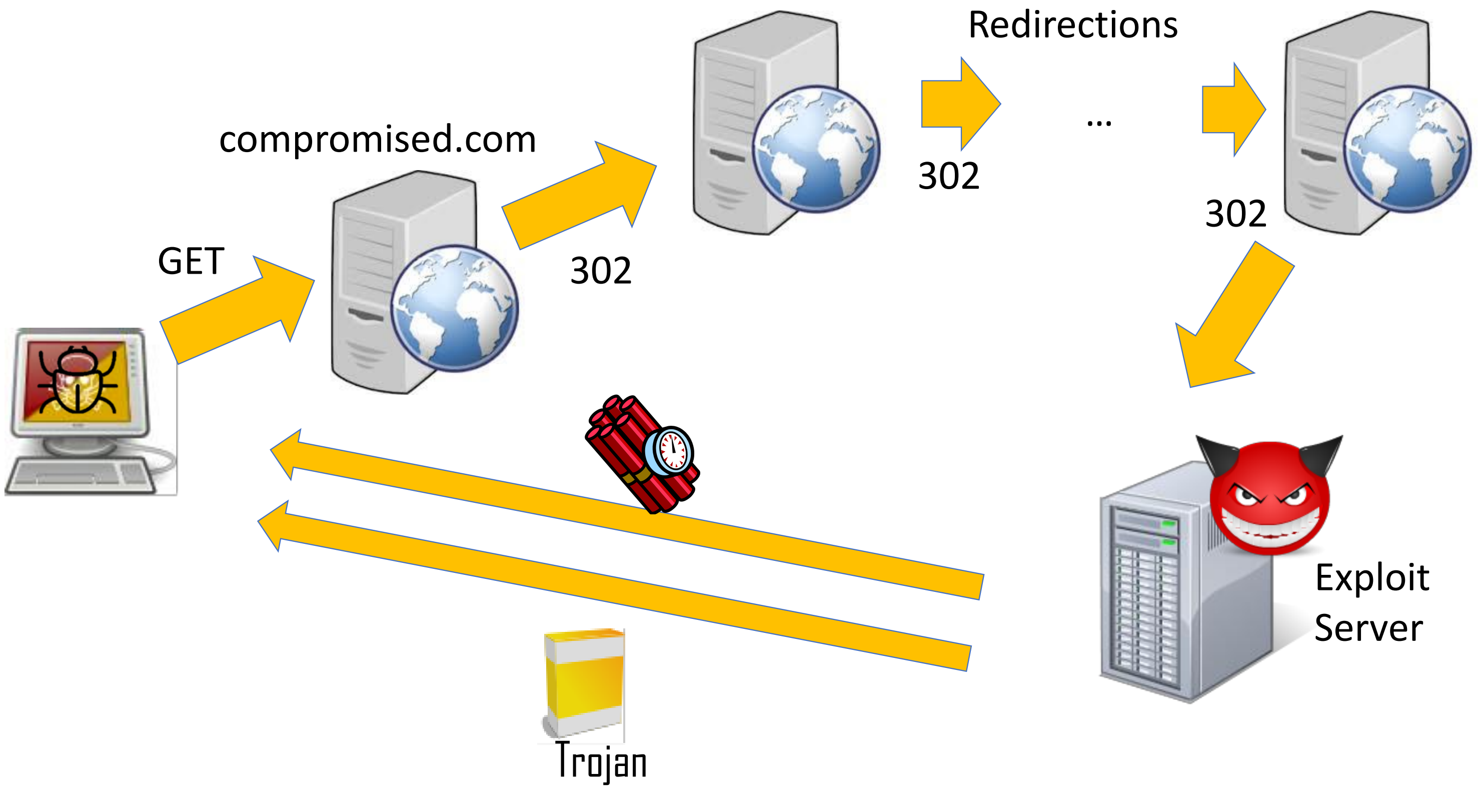
Vulnerability Impact: Malware Delivery

- Vulnerabilities are important distribution vector for malware
- A Windows host is exposed to 297 vulnerabilities in a year

  The security exposure of software portfolios. RSA Conference, March 2010

  - Likely more nowadays
- Spear-phishing attacks
  - Email with file attached, convince the user to open attachment
  - Target vulnerabilities in document readers, editors, multimedia players
- Scanning attacks
  - Target vulnerabilities in network services (e.g., Web servers, Email servers)
- Drive-by download attacks
  - User visits malicious website
  - Target vulnerabilities in browsers and their plugins (e.g., Java, PDF)

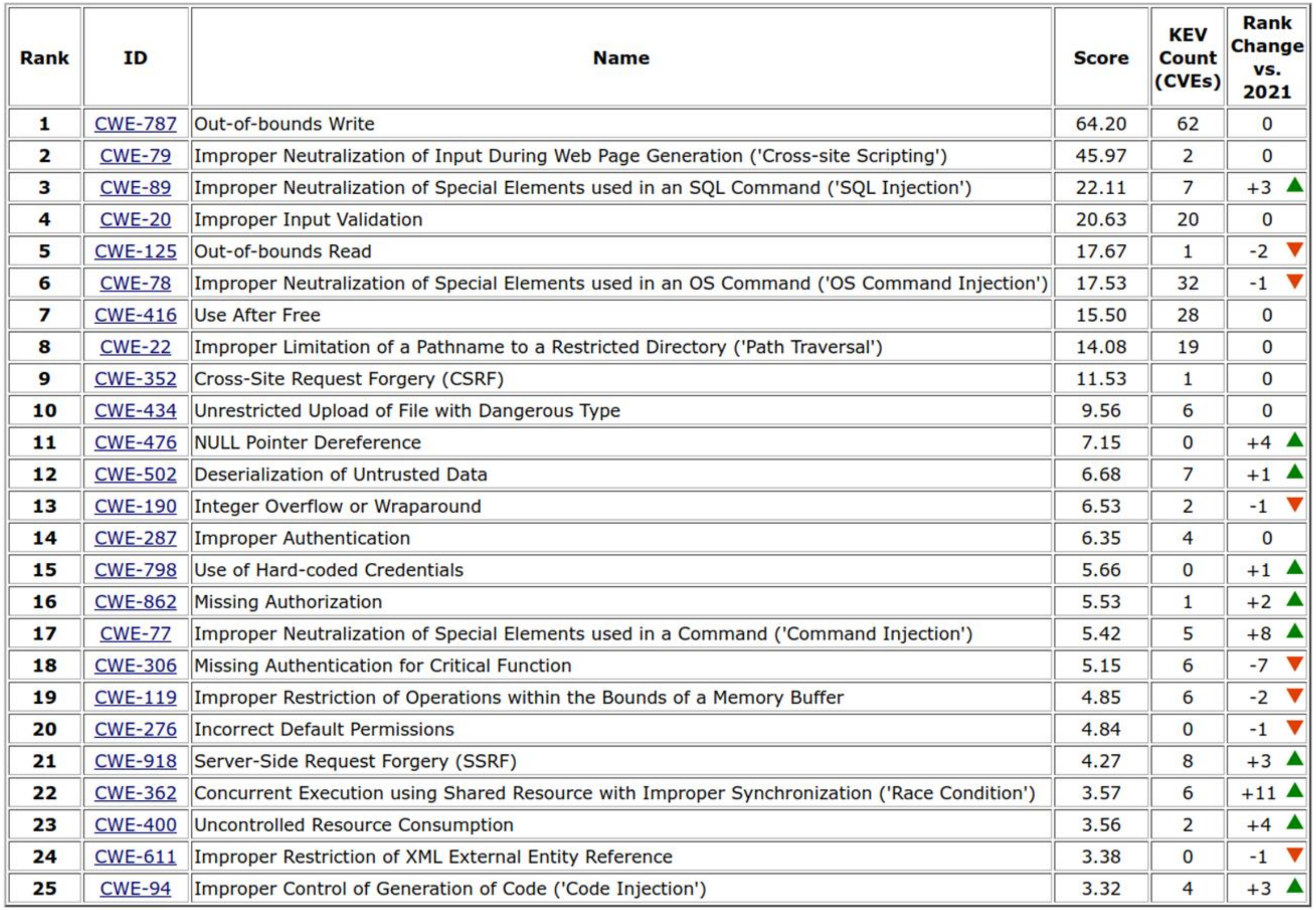
Vulnerability Impact: Drive-by Download



Vulnerability Types

- Memory Safety
  - Buffer Overflow (Stack, Heap)
  - Out-of-bounds read
  - NULL Pointer Dereference
  - Double Free
  - Use-After-Free
- Format String Vulnerabilities
- Integer Overflow/Underflow
- Web
  - Cross-Site Scripting (XSS)
  - SQL injection
  - Cross-Site Request Forgery (CSRF)
- Command Injection
- Path Traversal
- Cryptographic Misuse
- Missing access control
- Hardcoded credentials
- Time of Check Time of Use (TOCTOU)
- …

Common Weaknesses Enumeration Top 25 (2022)



Rank	ID	Name	Score	KEV Count (CVEs)	Rank Change vs. 2021	
1	CWE-787	Out-of-bounds Write	64.20	62	0	Memory
2	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.97	2	0	Web
3	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	22.11	7	+3 ▲	Web
4	CWE-20	Improper Input Validation	20.63	20	0	
5	CWE-125	Out-of-bounds Read	17.67	1	-2 ▼	Memory
6	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	17.53	32	-1 ▼	
7	CWE-416	Use After Free	15.50	28	0	Memory
8	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.08	19	0	
9	CWE-352	Cross-Site Request Forgery (CSRF)	11.53	1	0	Web
10	CWE-434	Unrestricted Upload of File with Dangerous Type	9.56	6	0	Web
11	CWE-476	NULL Pointer Dereference	7.15	0	+4 ▲	Memory
12	CWE-502	Deserialization of Untrusted Data	6.68	7	+1 ▲	
13	CWE-190	Integer Overflow or Wraparound	6.53	2	-1 ▼	
14	CWE-287	Improper Authentication	6.35	4	0	
15	CWE-798	Use of Hard-coded Credentials	5.66	0	+1 ▲	
16	CWE-862	Missing Authorization	5.53	1	+2 ▲	
17	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	5.42	5	+8 ▲	
18	CWE-306	Missing Authentication for Critical Function	5.15	6	-7 ▼	
19	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	4.85	6	-2 ▼	Memory
20	CWE-276	Incorrect Default Permissions	4.84	0	-1 ▼	
21	CWE-918	Server-Side Request Forgery (SSRF)	4.27	8	+3 ▲	Web
22	CWE-362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	3.57	6	+11 ▲	
23	CWE-400	Uncontrolled Resource Consumption	3.56	2	+4 ▲	
24	CWE-611	Improper Restriction of XML External Entity Reference	3.38	0	-1 ▼	
25	CWE-94	Improper Control of Generation of Code ('Code Injection')	3.32	4	+3 ▲	

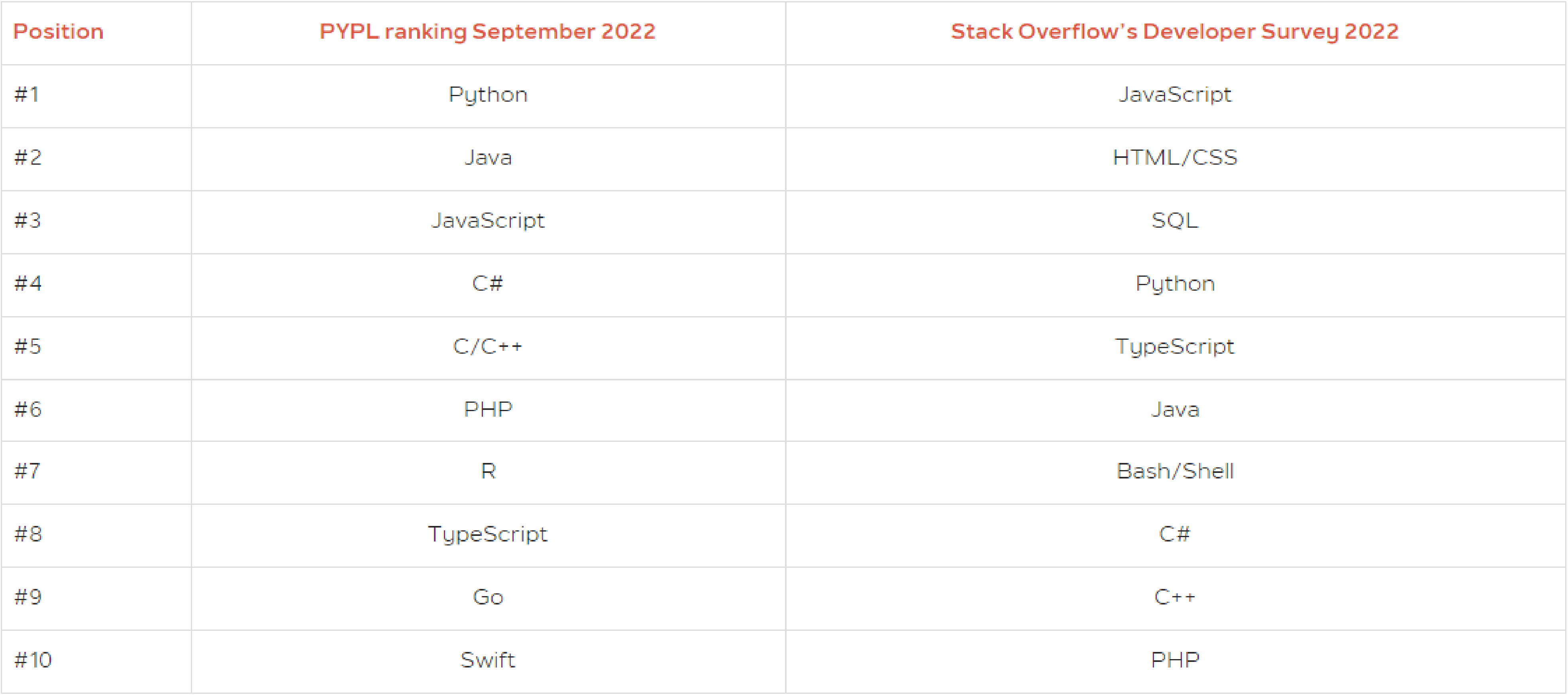
https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html#cwe_top_25

Vulnerabilities and Programming Languages

- Some vulnerabilities are specific to some programming languages
  - Memory safety → C/C++
  - Many modern programming languages are memory-safe
    Java, Python, C#, Ruby, Haskell, Scala, Go, Objective Caml, Rust
  - C/C++ widely used for performance (OSes, Browsers, Servers, …)
  - **Use a memory-safe programming language when possible!**

- Some vulnerabilities affect many / all languages
  - Command Injection
  - Cryptographic misuse
  - Hardcoded credentials
  - …

Top Programming Languages

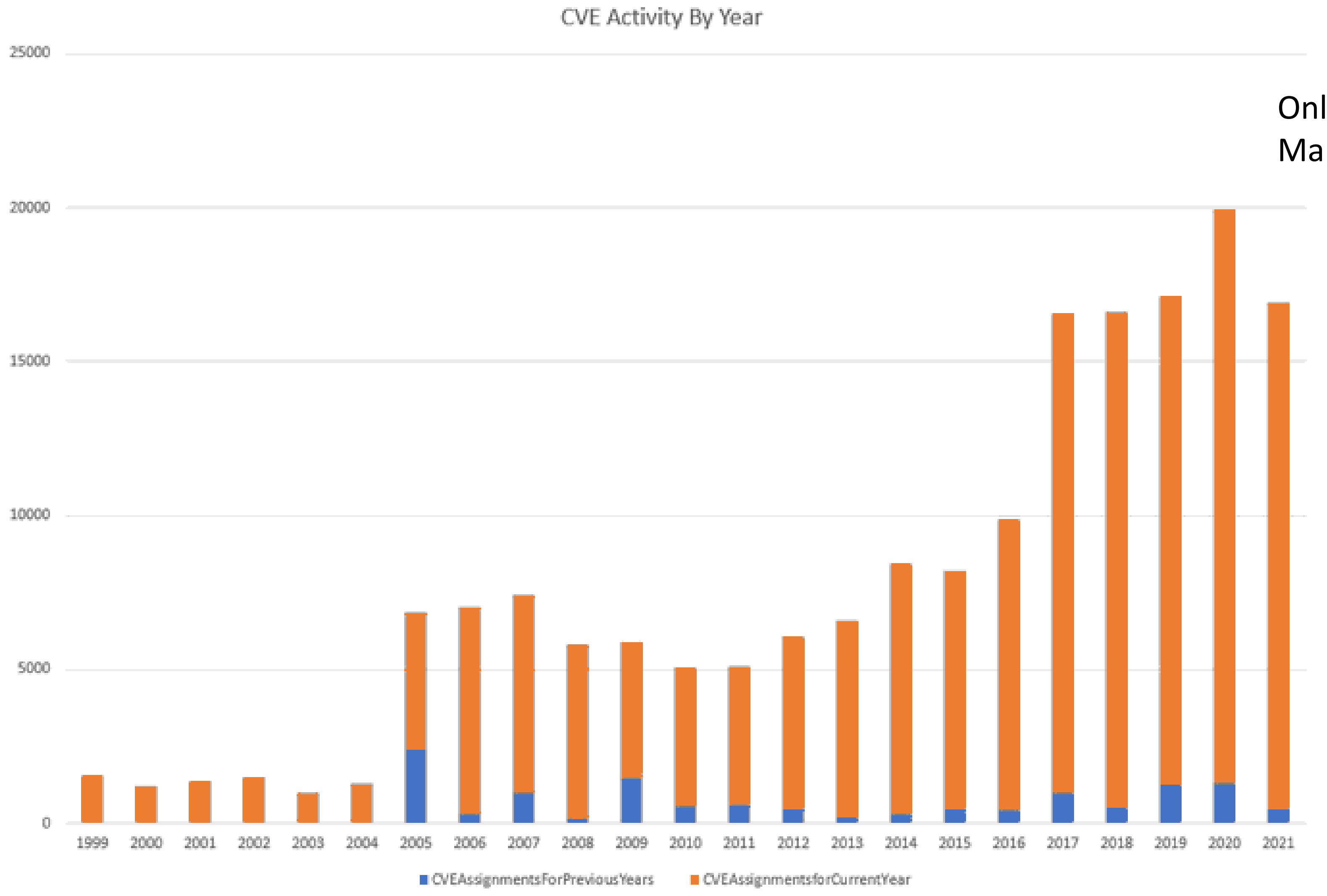


Position	PYPL ranking September 2022	Stack Overflow's Developer Survey 2022
#1	Python	JavaScript
#2	Java	HTML/CSS
#3	JavaScript	SQL
#4	C#	Python
#5	C/C++	TypeScript
#6	PHP	Java
#7	R	Bash/Shell
#8	TypeScript	C#
#9	Go	C++
#10	Swift	PHP

Vulnerability Databases

- Create inventories of vulnerabilities
  - Largely incomplete: focus on certain programs or services
  - Assign unique identifiers to vulnerability
  - Vulnerability details: program version affected, platform…
  - **Given program version → Get list of vulnerabilities affecting it**
- CVE: Common Vulnerabilities and Exposures
  - Assigns popular CVE identifiers
    - Heartbleed **(**CVE-2014-0160), FREAK **(**CVE-2015-0204)
  - Publicly available data
- Open Cloud Vulnerability Database (Open CVDB)
  - Security bugs in cloud services
  - https://www.cloudvulndb.org/

CVE Vulnerability Identifiers Issued



Only vulnerabilities in DB
Many more exist!

Vulnerability Entry: HeartBleed

- CVE-2014-0160
- Program: OpenSSL
- Versions affected: from 1.0.1 up to 1.0.1g (excluded)
- Published Date: 04/07/2014
- Class: CWE-131 (Improper Restriction of Operations within the Bounds of a Memory Buffer)
- Description

  The (1) TLS and (2) DTLS implementations in OpenSSL 1.0.1 before 1.0.1g do not properly handle Heartbeat Extension packets, which allows remote attackers to obtain sensitive information from process memory via crafted packets that trigger a buffer over-read, as demonstrated by reading private keys, related to d1_both.c and t1_lib.c, aka the Heartbleed bug.
- Source / Credit: RedHat Inc.
- CVSS Base Score: 7.5 (High)
- References: [Advisories, Solutions, Tools]

Common Vulnerability Scoring System (CVSS)

- Numerical score to capture vulnerability severity
- CVSS consists of 3 metric groups: Base, Temporal, Environmental
  - Base in range [0,10]. Captures static characteristics:
    network attack vector, attack complexity, user interaction needed, privileges required, impact to confidentiality/integrity/availability
  - Temporal & Environmental affect base score, e.g., patch availability
- CVSS can be translated into qualitative severity
  - Low (0,4), Medium [4,7), High [7,9), Critical [9,10]
- CVSS often used for prioritization
  - Easy to use incorrectly, e.g., does not capture impact

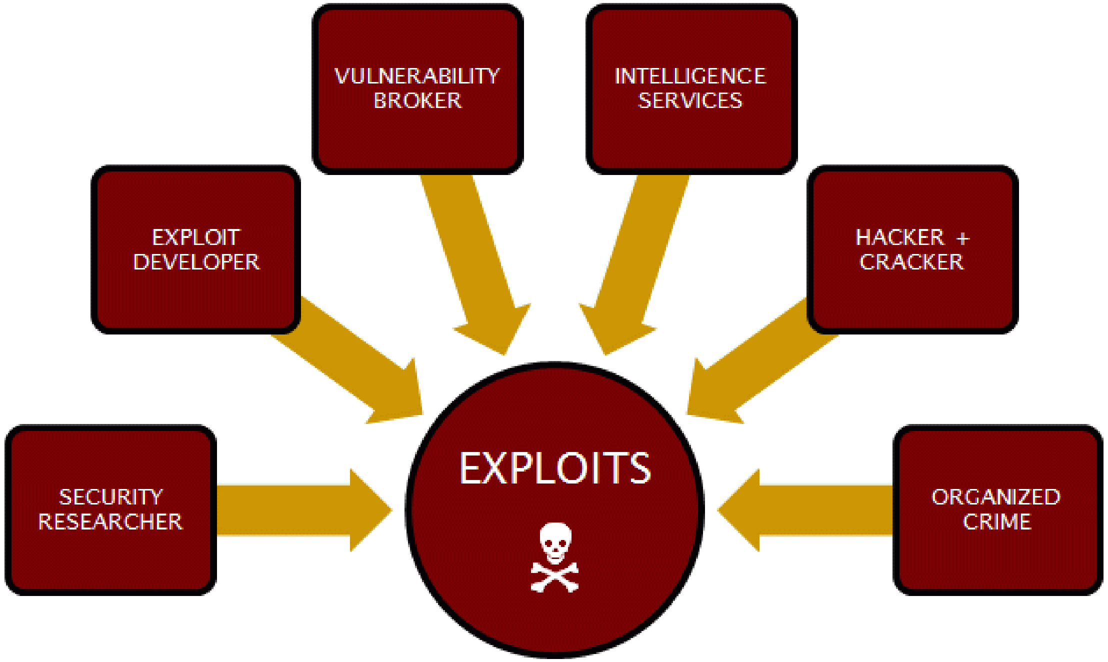
Exploits

Exploits

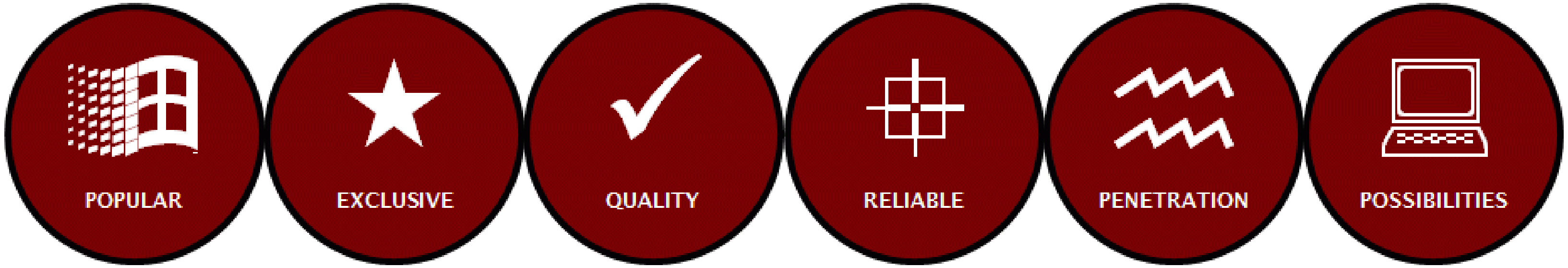
- **Program input** used by attacker to trigger vulnerability
  - Needs to reach the **vulnerability point / state** (e.g., vulnerable function)
  - Needs to satisfy **vulnerability condition** (e.g., overflow buffer)
  - Needs to **bypass defenses** (e.g., OS defenses)
- **Input Validation** / Filtering is key, but hard to do completely right
- Exploit vectors:
  - Input files, Received network traffic , Environment variables
- Triggering the vulnerability often causes a **crash**
- But attacker can smartly produce inputs that avoid crash and:
  - Run code → **Remote Code Execution** (RCE)
  - **Leak** valuable data (e.g., credentials)
  - **Overwrite** sensitive data (e.g., security checks)

Exploit Types

- Proof of Concept (PoC)
  - Demonstrates vulnerability
  - Assumes no defenses
  - Often leads to crash
  - Very useful in debugging
- Full exploit
  - Has a useful payload (run code, leak data, …)
  - Bypasses defenses
  - Costly

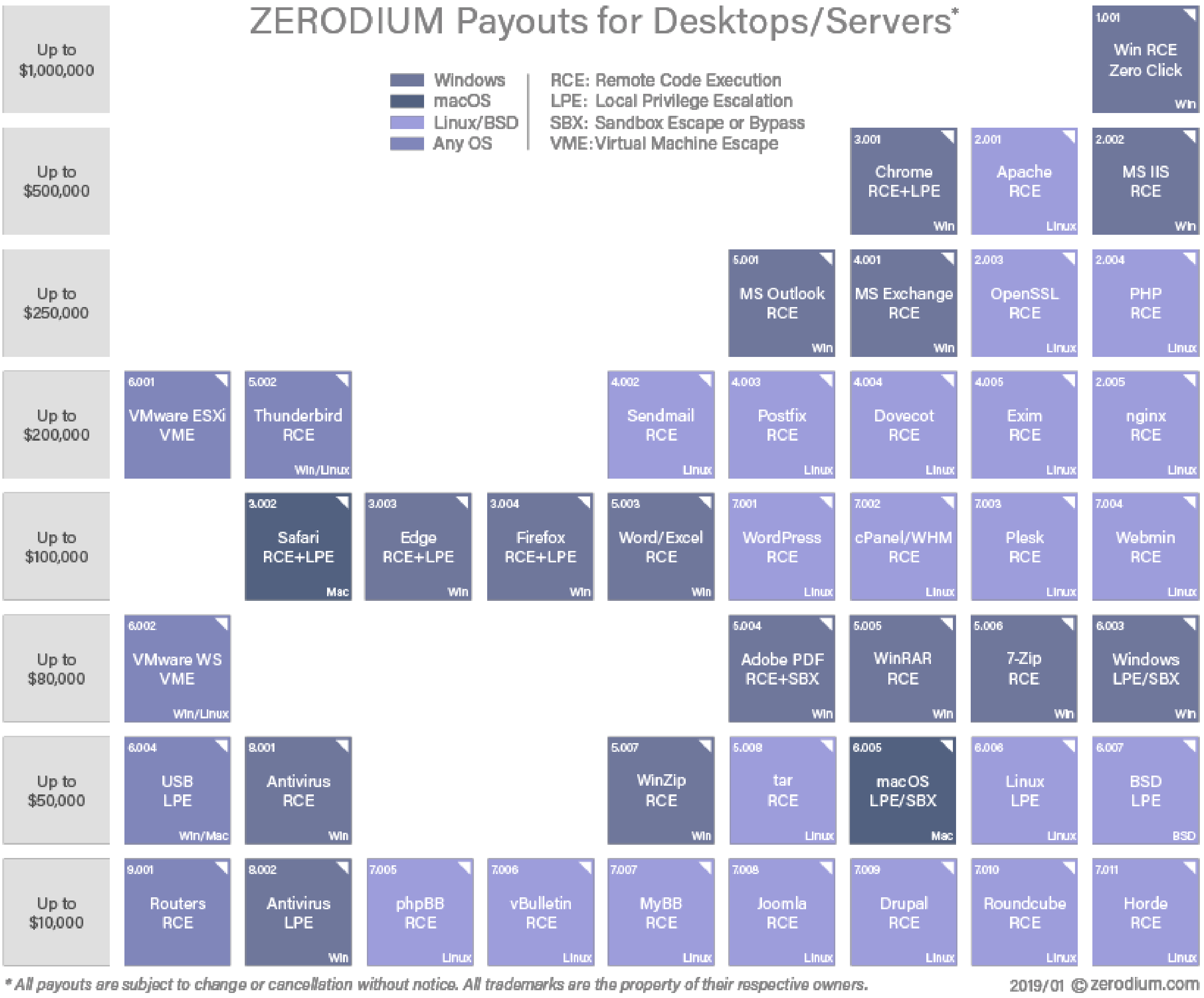


A Market for Exploits: Determining Exploit Price



- Popularity of the target program
- Exclusivity of the exploit
- Quality of the exploit
- Reliability of attack (i.e., avoids crashes that may indicate attack)
- Penetration: security mechanisms that is able to bypass
- Possibilities enabled: DoS, DB access, RCE, …

Prices Paid: Desktops / Servers



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

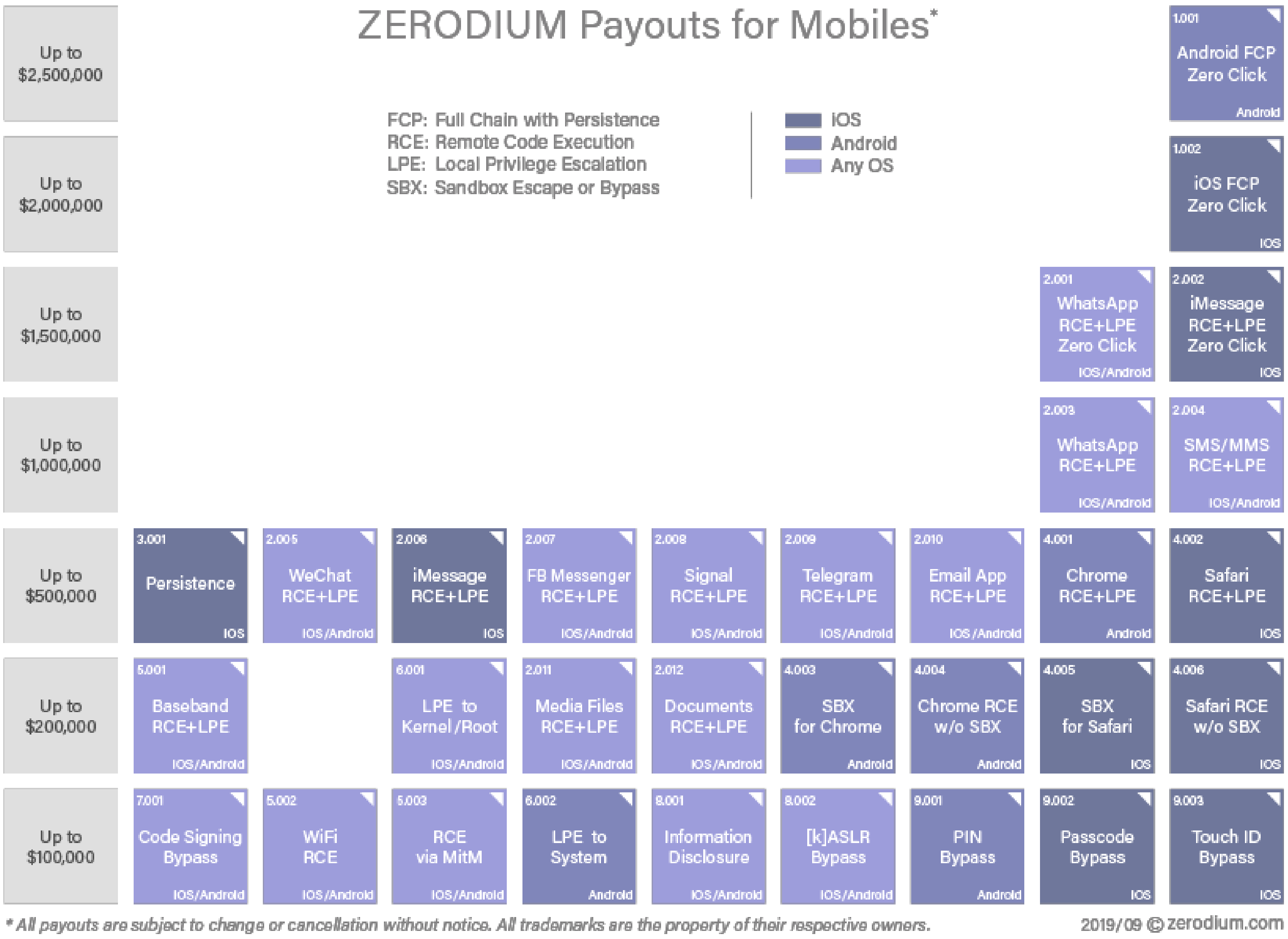
Up to $1,000,000									1.001 Win RCE Zero Click Win
Up to $500,000							3.001 Chrome RCE+LPE Win	2.001 Apache RCE Linux	2.002 MS IIS RCE Win
Up to $250,000						5.001 MS Outlook RCE Win	4.001 MS Exchange RCE Win	2.003 OpenSSL RCE Linux	2.004 PHP RCE Linux
Up to $200,000	6.001 VMware ESXi VME	5.002 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

https://zerodium.com
/program.html

Prices Paid: Mobile



ZERODIUM Payouts for Mobiles*

FCP: Full Chain with Persistence
RCE: Remote Code Execution
LPE: Local Privilege Escalation
SBX: Sandbox Escape or Bypass

- iOS
- Android
- Any OS

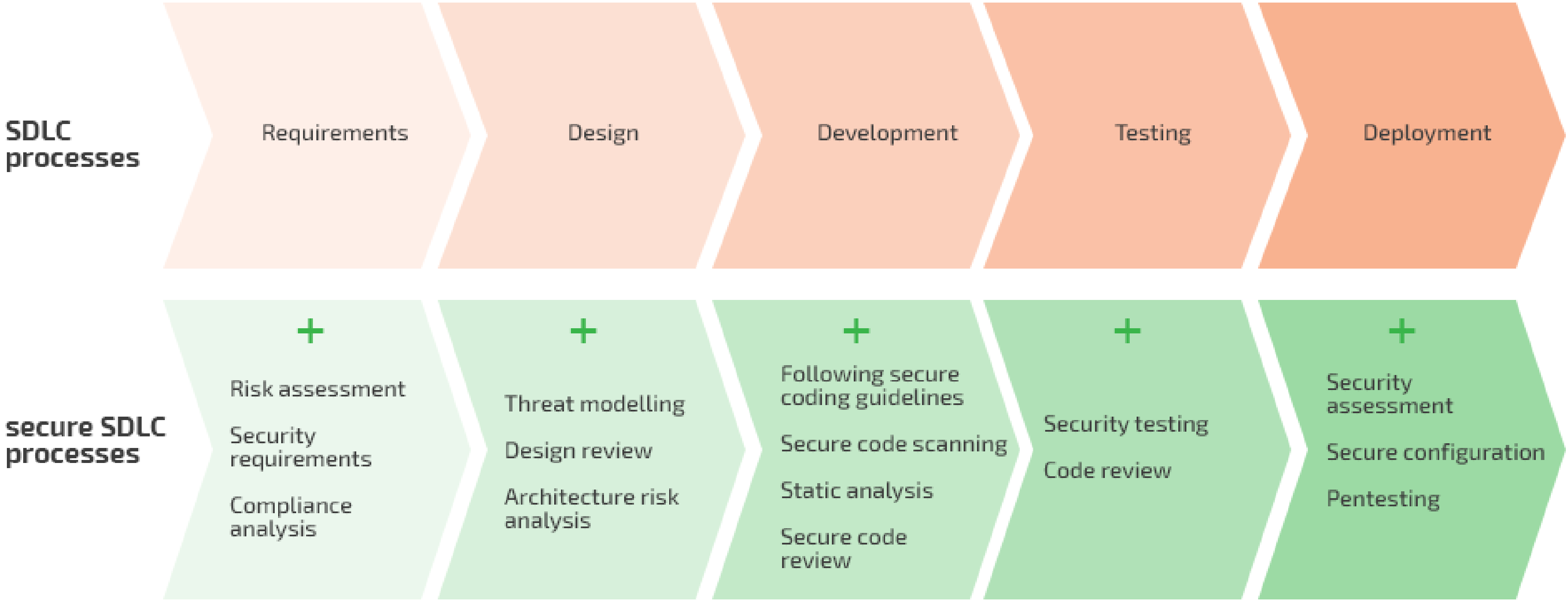
Up to $2,500,000									1.001 Android FCP Zero Click Android
Up to $2,000,000									1.002 iOS FCP Zero Click iOS
Up to $1,500,000								2.001 WhatsApp RCE+LPE Zero Click iOS/Android	2.002 iMessage RCE+LPE Zero Click iOS
Up to $1,000,000								2.003 WhatsApp RCE+LPE iOS/Android	2.004 SMS/MMS RCE+LPE iOS/Android
Up to $500,000	3.001 Persistence iOS	2.005 WeChat RCE+LPE iOS/Android	2.006 iMessage RCE+LPE iOS	2.007 FB Messenger RCE+LPE iOS/Android	2.008 Signal RCE+LPE iOS/Android	2.009 Telegram RCE+LPE iOS/Android	2.010 Email App RCE+LPE iOS/Android	4.001 Chrome RCE+LPE Android	4.002 Safari RCE+LPE iOS
Up to $200,000	5.001 Baseband RCE+LPE iOS/Android		6.001 LPE to Kernel/Root iOS/Android	2.011 Media Files RCE+LPE iOS/Android	2.012 Documents RCE+LPE iOS/Android	4.003 SBX for Chrome Android	4.004 Chrome RCE w/o SBX Android	4.005 SBX for Safari iOS	4.006 Safari RCE w/o SBX iOS
Up to $100,000	7.001 Code Signing Bypass iOS/Android	5.002 WiFi RCE iOS/Android	5.003 RCE via MitM iOS/Android	6.002 LPE to System Android	8.001 Information Disclosure iOS/Android	8.002 [k]ASLR Bypass iOS/Android	9.001 PIN Bypass Android	9.002 Passcode Bypass iOS	9.003 Touch ID Bypass iOS

* *All payouts are subject to change or cancellation without notice. All trademarks are the property of their respective owners.*

2019/09 © zerodium.com

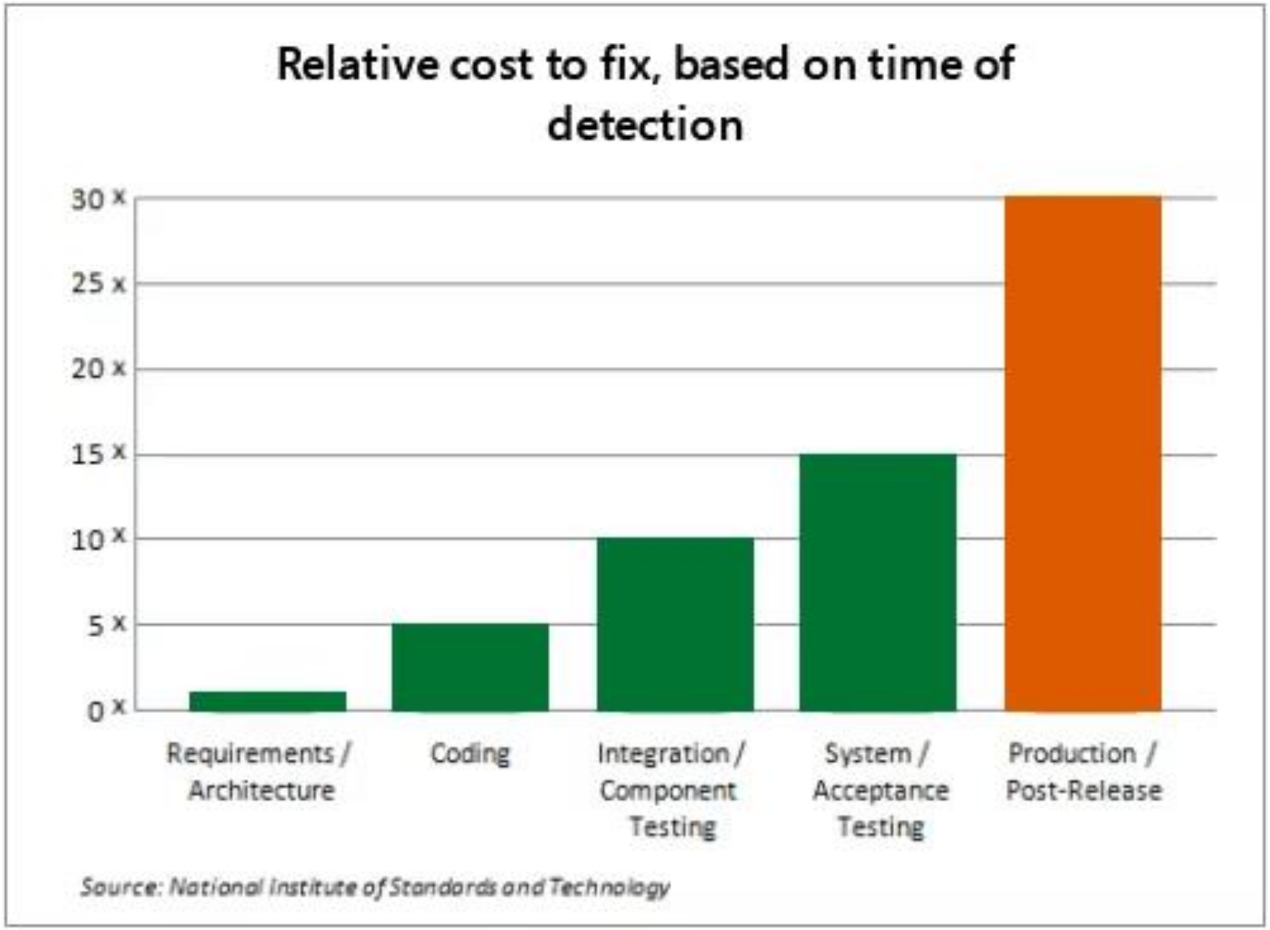
Vulnerability Protections

Security in Software Development Life Cycle



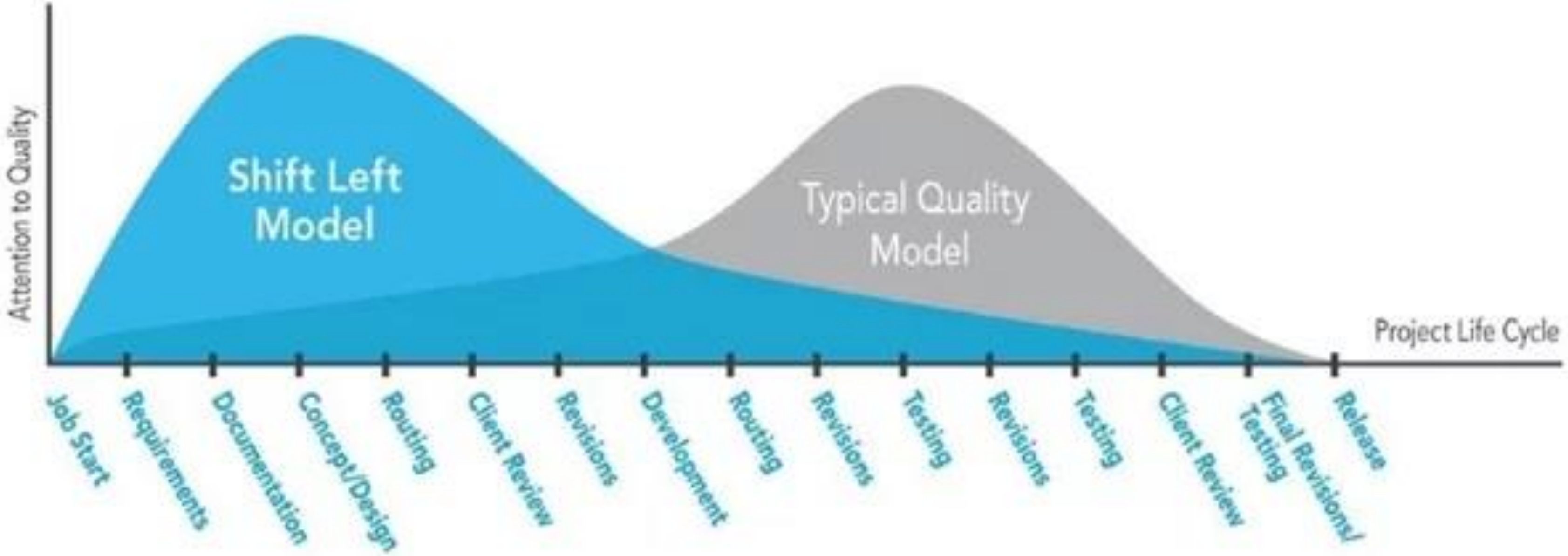
	Requirements	Design	Development	Testing	Deployment
SDLC processes	Requirements	Design	Development	Testing	Deployment
secure SDLC processes	+ Risk assessment Security requirements Compliance analysis	+ Threat modelling Design review Architecture risk analysis	+ Following secure coding guidelines Secure code scanning Static analysis Secure code review	+ Security testing Code review	+ Security assessment Secure configuration Pentesting

Relative Cost of Fixing a Vulnerability



Source: National Institute of Standards and Technology

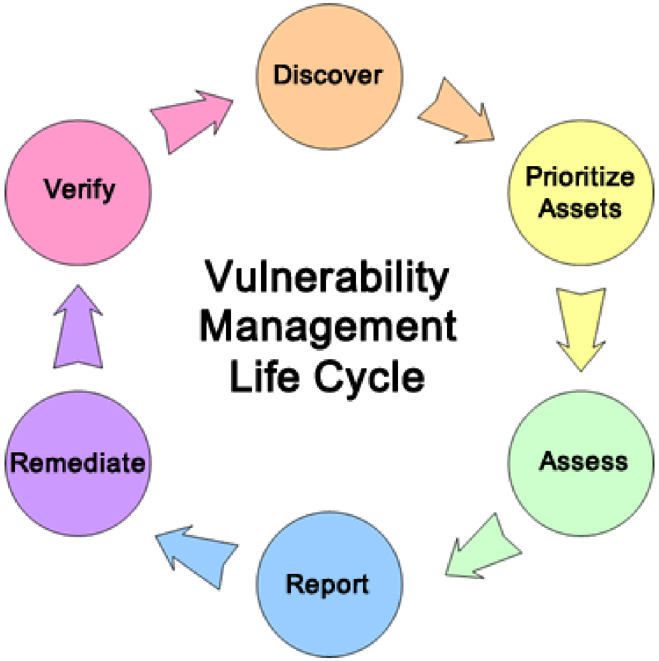
The Sooner Found, the Cheaper: Shift-Left



Source: http://www.shiftleftqa.com/#about

Vulnerability Management

- Cyclical Process an organization should follow
  1. Asset discovery
  2. Prioritize
  3. Risk assessment
  4. Vulnerability discovery & reporting
  5. Vulnerability remediation
  6. Verify remediation
  7. Repeat
- Developer process
  - Receive vulnerability report
  - Determine priority
  - Develop remediation: patch the vulnerability
  - Deploy patch

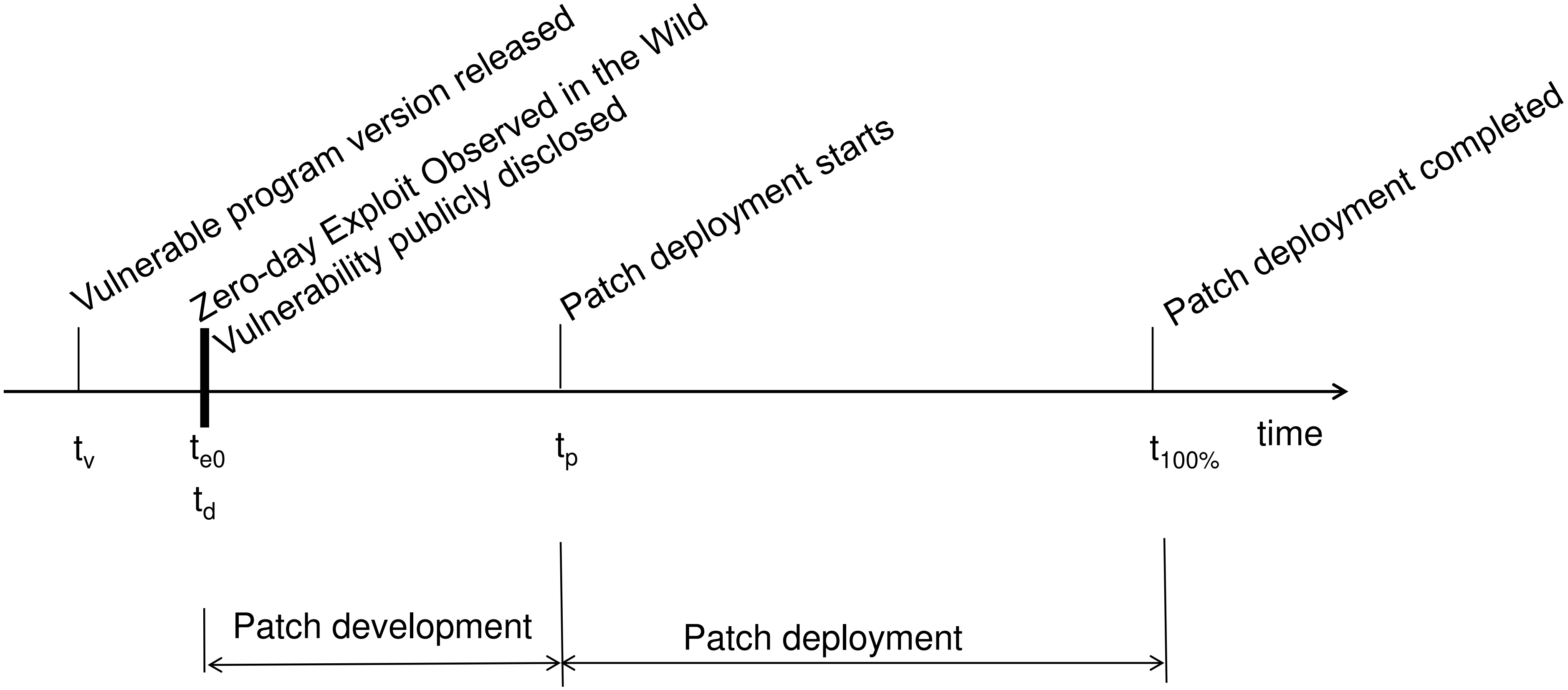


Vulnerability Lifecycle

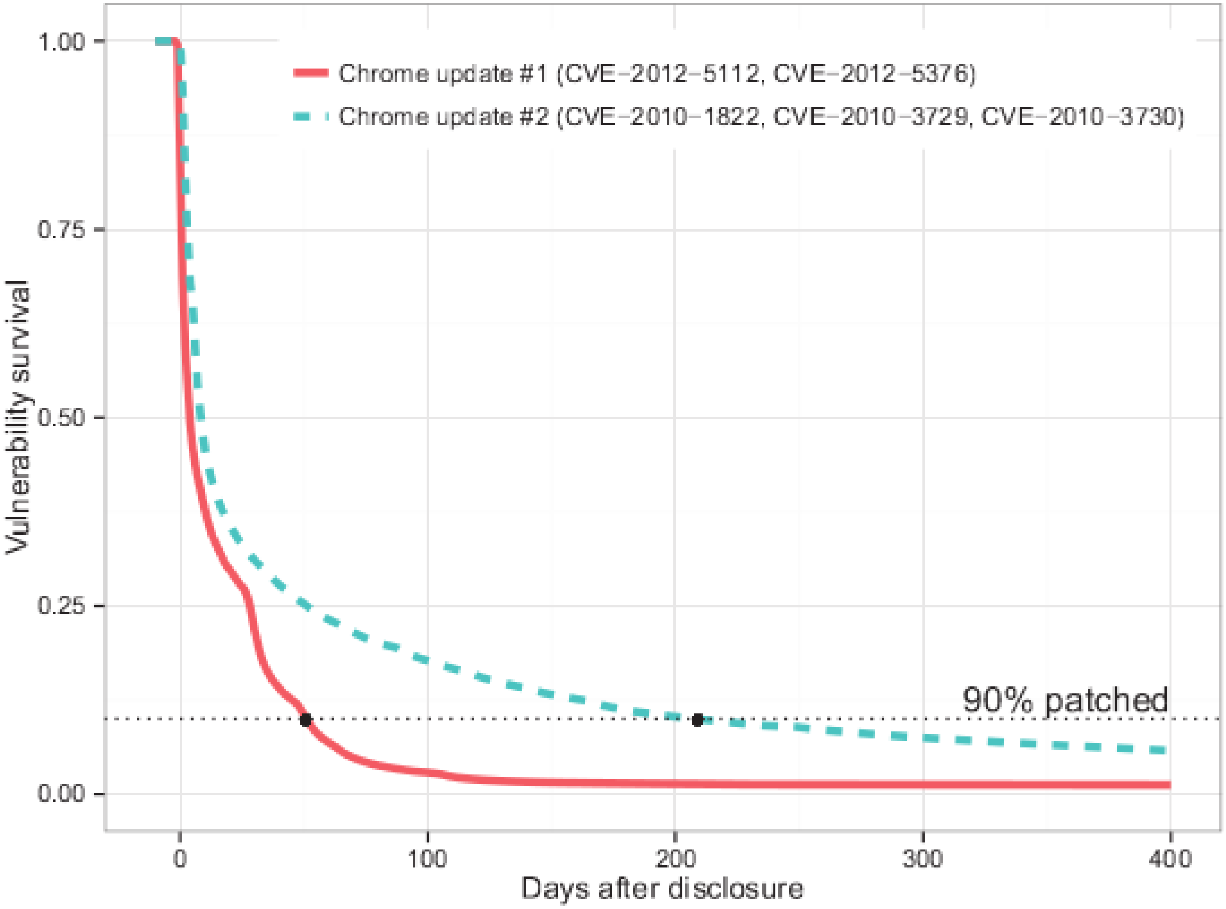
Vulnerability Disclosure

- Vulnerabilities often found by users or security analysts
- Responsible Disclosure (or Coordinated Disclosure)
  - First report to the affected parties, e.g., publisher of vulnerable SW
  - Allow time to remedy/patch before making vulnerability details public
    - Google Project Zero has a 90-day disclosure deadline
- Full Disclosure
  - All vulnerability details publicized, even if no patch available
  - **Zero-Day vulnerability**: previously unknown, no patch exists
  - Puts pressure on SW publisher to quickly address the vulnerability
- Bug bounty programs incentivize responsible disclosure

Vulnerability Lifecycle: Zero-Day


Vulnerable program version released
Zero-day Exploit Observed in the Wild
Vulnerability publicly disclosed
Patch deployment starts
Patch deployment completed
t_v
t_{e0}
t_d
t_p
$t_{100\%}$
time
Patch development
Patch deployment

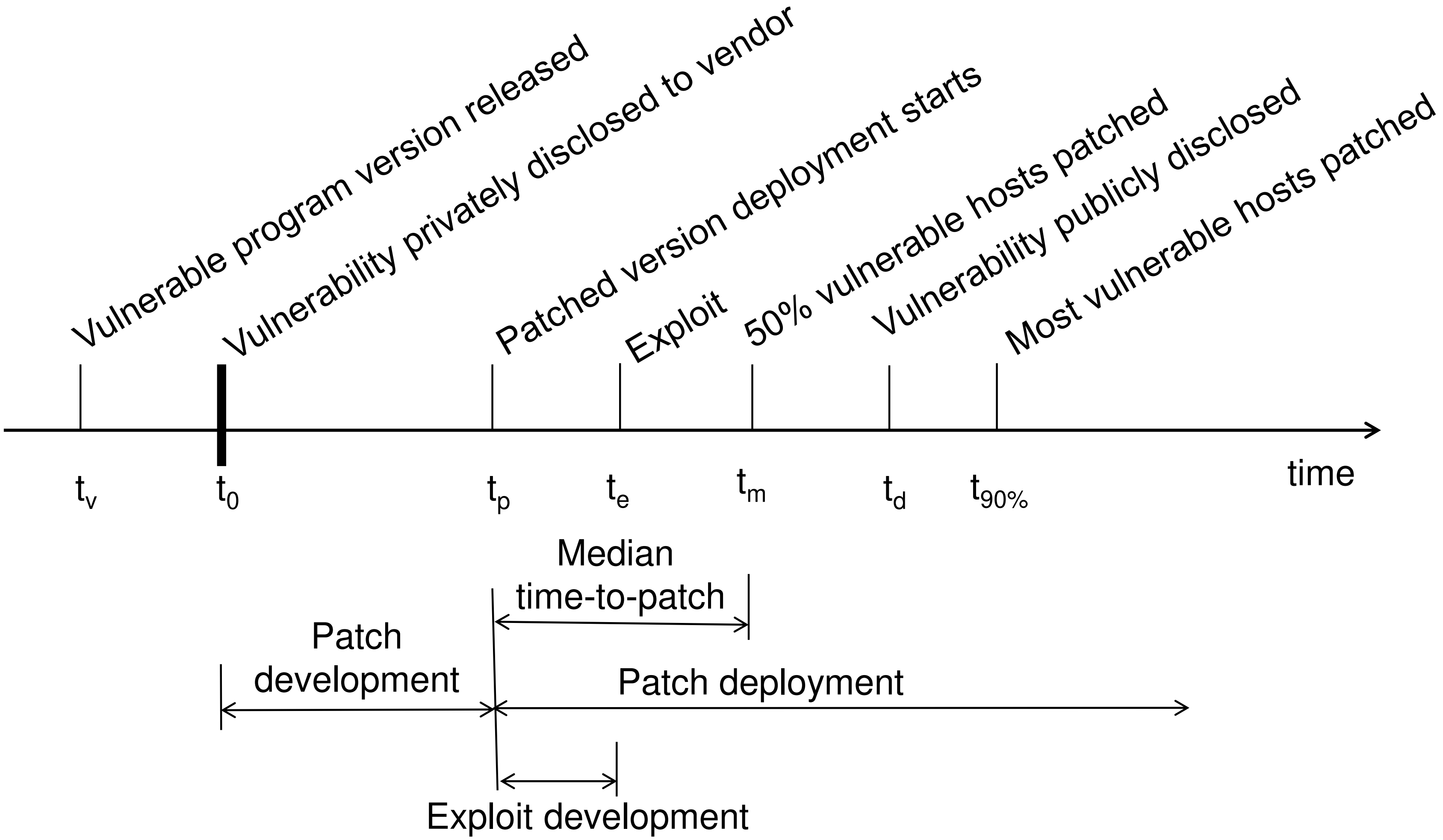
Vulnerable Population Decay



- Patch deployment not instantaneous
- Vulnerable population decays over time
- Patching speed varies
    - Per program
    - Per vulnerability
- Vulnerable population never really reaches zero

The Attack of the Clones: A Study of the Impact of Shared Code on Vulnerability Patching. IEEE S&P 2015

Vulnerability Lifecycle: Responsible Disclosure

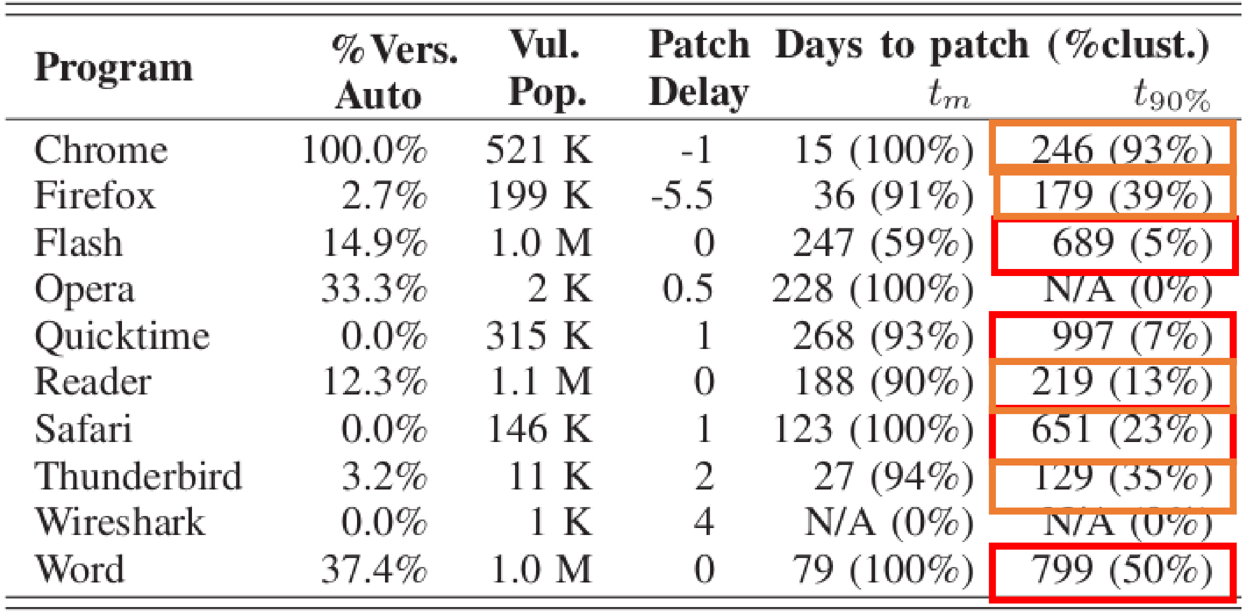

Vulnerable program version released
Vulnerability privately disclosed to vendor
Patched version deployment starts
Exploit
50% vulnerable hosts patched
Vulnerability publicly disclosed
Most vulnerable hosts patched
t_v
t_0
t_p
t_e
t_m
t_d
$t_{90\%}$
time
Median time-to-patch
Patch development
Patch deployment
Exploit development

Race between Patches and Exploits

- Once a vulnerability is disclosed or patched, exploit creators start building their exploits
- Once an exploit is ready, a fraction of hosts are still vulnerable
  - Median fraction of hosts patched when exploit released ≤ 14%
- Patch deployment speed
  - Median time to patch half of the vulnerable hosts is 45 days
  - Significant differences between applications
  - Automatic updates provide a fundamental speed boost

The Attack of the Clones: A Study of the Impact of Shared Code on Vulnerability Patching. IEEE S&P 2015

Patch Deployment for Different Programs



Program	%Vers.	Vul.	Patch	Days to patch (%clust.)	
	Auto	Pop.	Delay	t_m	$t_{90\%}$
Chrome	100.0%	521 K	-1	15 (100%)	246 (93%)
Firefox	2.7%	199 K	-5.5	36 (91%)	179 (39%)
Flash	14.9%	1.0 M	0	247 (59%)	689 (5%)
Opera	33.3%	2 K	0.5	228 (100%)	N/A (0%)
Quicktime	0.0%	315 K	1	268 (93%)	997 (7%)
Reader	12.3%	1.1 M	0	188 (90%)	219 (13%)
Safari	0.0%	146 K	1	123 (100%)	651 (23%)
Thunderbird	3.2%	11 K	2	27 (94%)	129 (35%)
Wireshark	0.0%	1 K	4	N/A (0%)	N/A (0%)
Word	37.4%	1.0 M	0	79 (100%)	799 (50%)

Patching most hosts may take months

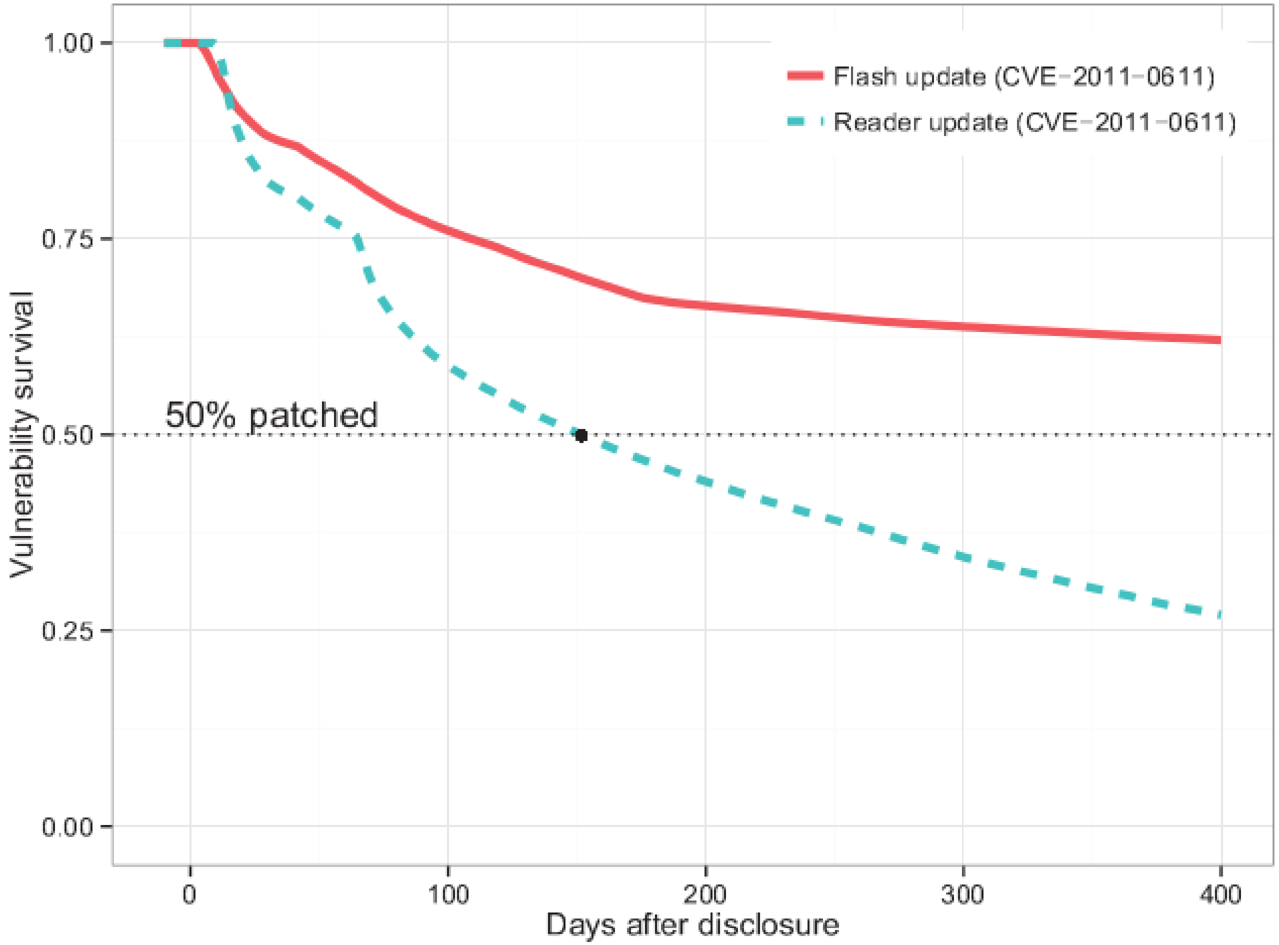
Or years!

The Attack of the Clones: A Study of the Impact of Shared Code on Vulnerability Patching. IEEE S&P 2015

Patch Deployment Challenges

- Each software vendor has its own patching policies
  - Automatic vs manual deployment
  - Frequency of updates
  - Version numbers
- Shared code
  - Vulnerable libraries shared by multiple programs → Only some patched
  - Multiple versions of a program installed → Only some patched
- Programs that are never closed
  - Browsers, OS, Servers

Shared Code Patching



- Vulnerability patched in library shared by multiple programs
- Patching speed varies for each program

The Attack of the Clones: A Study of the Impact of Shared Code on Vulnerability Patching. IEEE S&P 2015

Patching: Best Practices

- Apply patches as soon as possible!
  - Risk of incompatibility largely outweighed by risk of exploitation

- Build automatic updates into programs
  - Periodically checks for updates
  - Silently download and apply them
  - Challenging with programs that always run