Web Security IV:
Web Frameworks & Wrapup

CS 1660: Introduction to Computer Systems Security

Web Frameworks

Web Development

Usually managed by a 3-tier architecture with a client–server approach articulate in 3 layers logically separated in which:

- Presentation

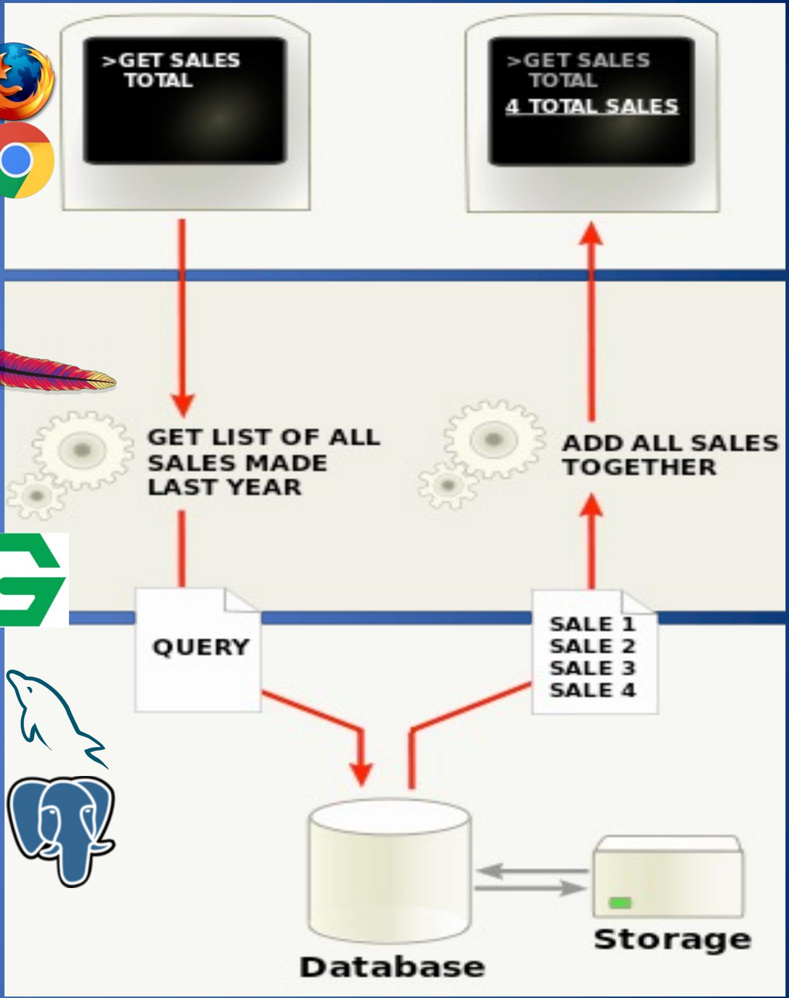
  This level of the application is the user interface. The interface is used to translate tasks and results to something the user can understand.

- Logic

  This layer coordinates the application of the web site, and it moves and processes data between the two surrounding layers

- Data tiers

  Information stored and retrieved from a database or file system. The information is passed back to the logic tier for processing, and then eventually back to the user



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

Threat and risk modeling process

- Browser may attack
  - Server
  - Other browsers
- Server may attack
  - Browser
  - Machine of browser
  - Other servers

- User may trust
  - Server to protect user data
  - Server to protect browser from other servers
  - Browser to protect user data
  - Browser to protect user from malicious server

Web Frameworks

Usually we do not develop website using just a text editor we use Web Frameworks that bring services e.g.:

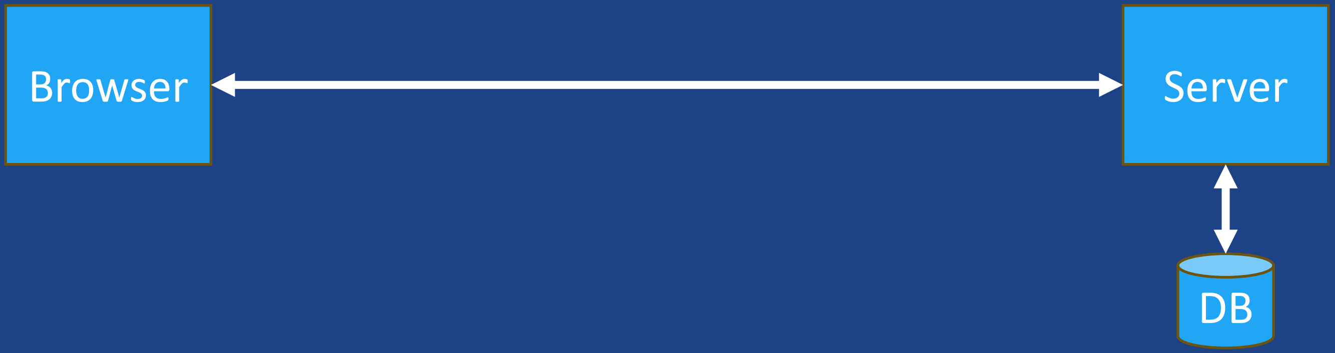
- URL routing
- Input form managing and validation
- HTML, XML, JSON, AJAX, etc.
- Database connection
- Web security against Cross-site request forgery (CSRF), SQL Injection, Cross-site Scripting (XSS), etc.
- Session repository and retrieval

- Apache Tomcat
- Spring MVC
- AngularJS
- JBoss
- Node.js
- Django
- Apache Struts

Web Security Standard solutions

- Usually web security is built in the framework or external libraries:
  - –Authentication and session management (e.g. cookies generation)
  - –Input validation (sanitization) through common patterns (email, credit card, etc.) or char escaping
  - –Avoid building SQL from user input
  - –Password: hash and salting
  - –Etc.

What have we learned?

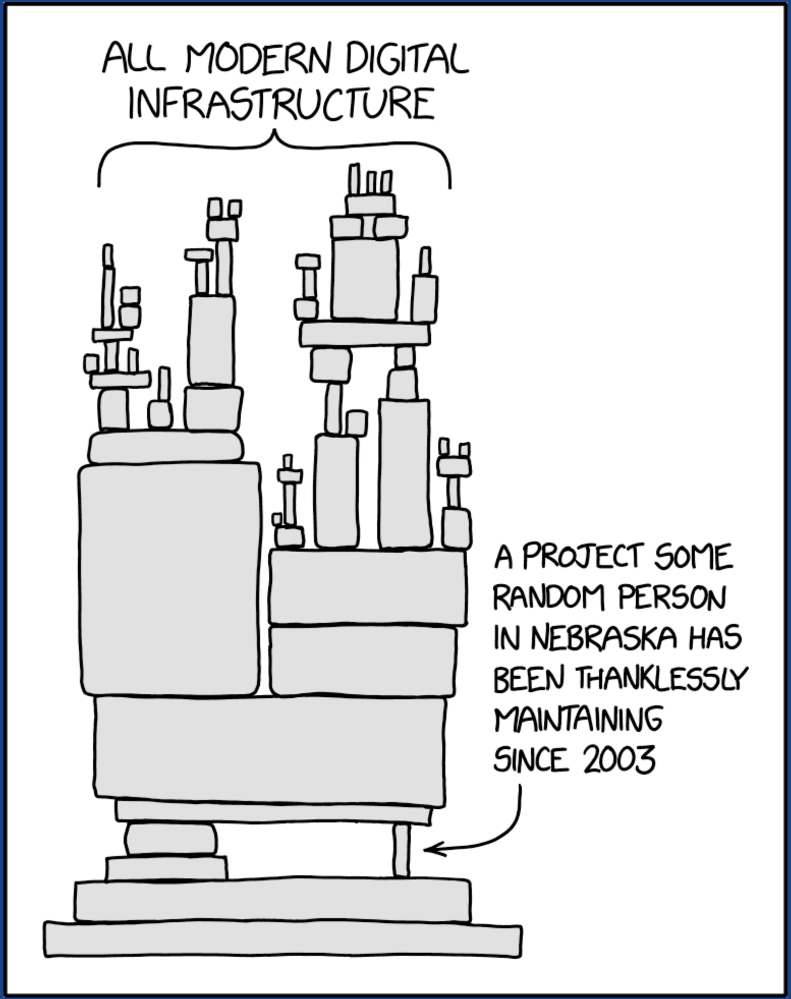


- Several *classes* of attacks that operate on different parts of the system
- Capabilities differ based on where vulnerability is located
- Problems across multiple components

The software stack…

What happens when a vulnerability is discovered?

What can go wrong?


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

“Dependency”: https://xkcd.com/2347/

Software Ecosystem + Security

- Modern software is built from many independently-maintained components

- Every component has different processes and development resources available for updates and security. Some have none.

Software Ecosystem + Security

- Modern software is built from many independently-maintained components
- Every component has different processes and development resources available for updates and security. Some have none.

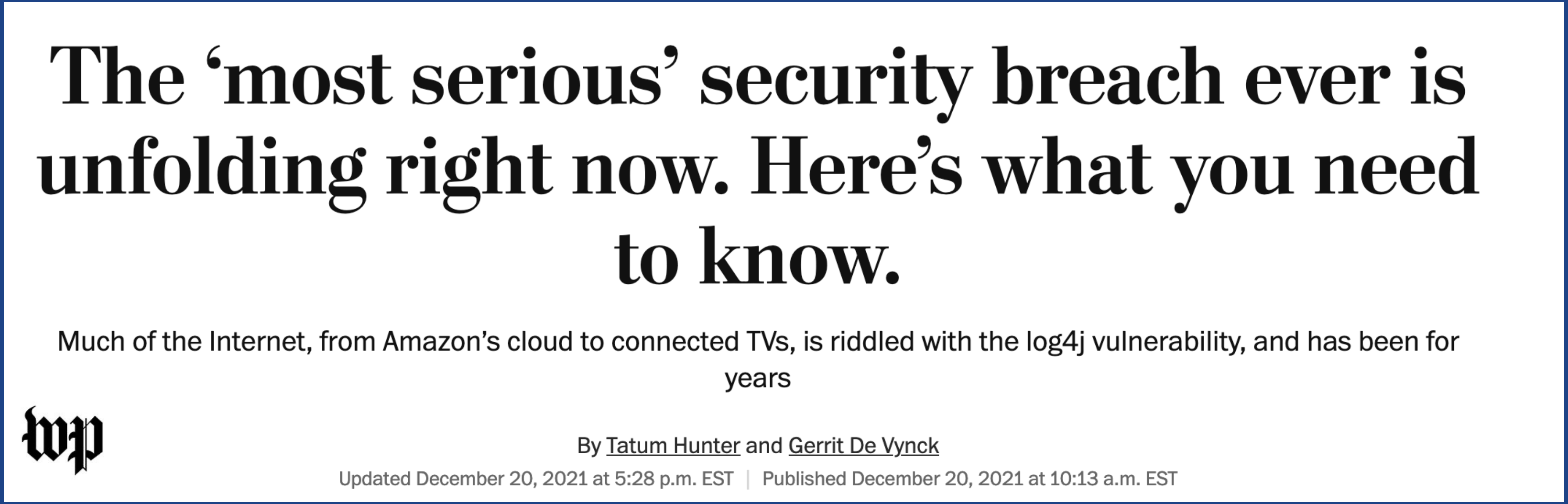
Requires a coordinated effort among many groups to monitor and update systems!
=> As much a social problem as a technical one!

When vulnerabilities occur…

- How to find a fix? (If it can be fixed…)

- How to distribute the update?

Example: log4j vulnerability



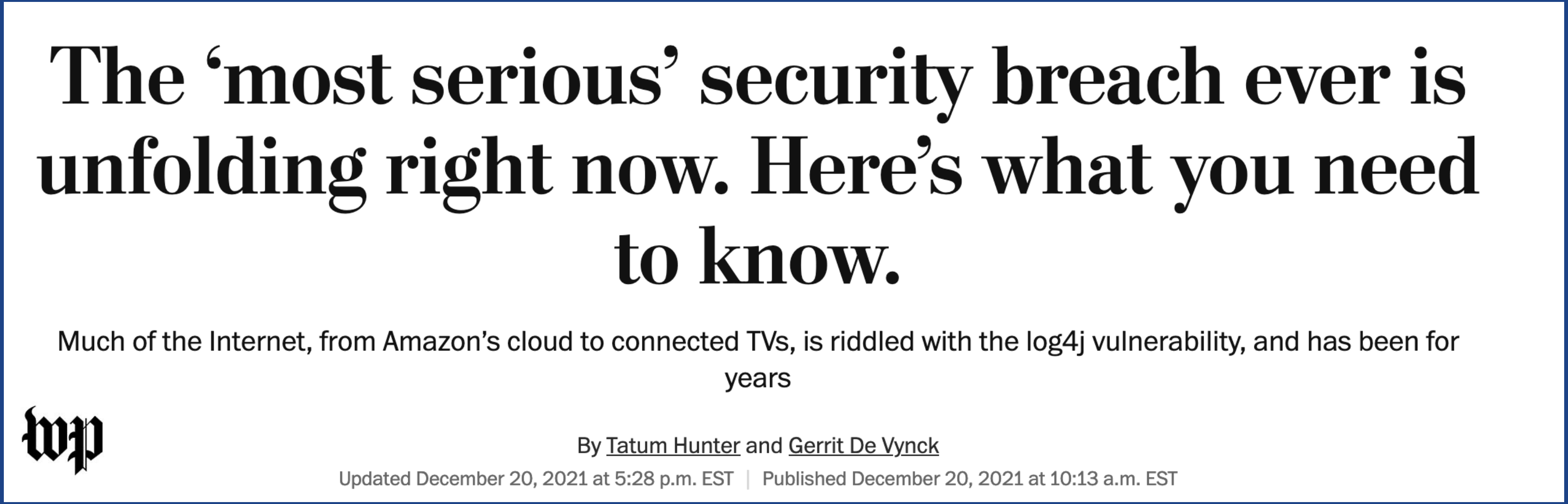
The 'most serious' security breach ever is unfolding right now. Here's what you need to know.

Much of the Internet, from Amazon's cloud to connected TVs, is riddled with the log4j vulnerability, and has been for years

By Tatum Hunter and Gerrit De Vynck

Updated December 20, 2021 at 5:28 p.m. EST | Published December 20, 2021 at 10:13 a.m. EST

Example: log4j vulnerability



The 'most serious' security breach ever is unfolding right now. Here's what you need to know.

Much of the Internet, from Amazon's cloud to connected TVs, is riddled with the log4j vulnerability, and has been for years

By Tatum Hunter and Gerrit De Vynck

Updated December 20, 2021 at 5:28 p.m. EST | Published December 20, 2021 at 10:13 a.m. EST

"Zero-day" arbitrary code execution in open-source Java library log4j since at least 2013, discovered in 2021

=> Estimated to have affected 93% of enterprise cloud environments

How do we find vulnerabilities?

What happens afterward?

Who finds vulnerabilities?

- *Hopefully* part of normal software development

- Security researchers (independent, academic, private)

Who finds vulnerabilities?

- *Hopefully* part of normal software development
- Security researchers (independent, academic, private)
- Might only find out once vulnerability has been exploited…

Who finds vulnerabilities?

- *Hopefully* part of normal software development
- Security researchers (independent, academic, private)
- Might only find out once vulnerability has been exploited…

=> “Zero day”: a vulnerability unknown to anyone capable of mitigating it (known only to attackers)

How to track them?

CVE (Common Vulnerabilities and Exposure): a standard numbering/tracking system for vulnerabilities across software projects

Eg. `CVE-2021-44228:  Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1) …`

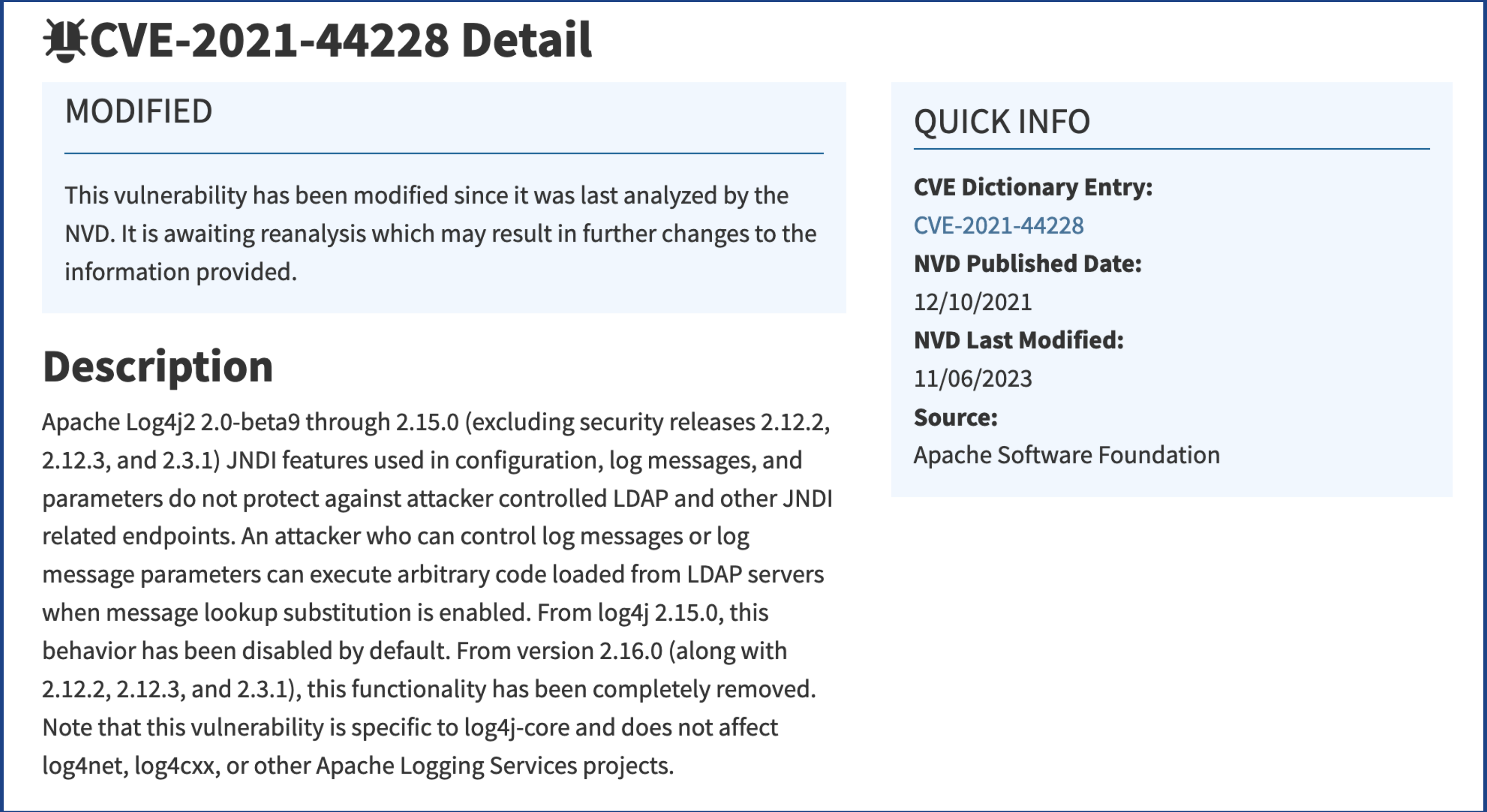
How to track them?

CVE (Common Vulnerabilities and Exposure): a standard numbering/tracking system for vulnerabilities across software projects

Eg. `CVE-2021-44228: Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1)`

How it works

- Primary numbering/databases maintained by MITRE corporation (US gov. funded) & NIST
- Software vendors assign CVEs based on vulnerability reports
- Many other vulnerability databases/resources use CVE numbers



CVE-2021-44228 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

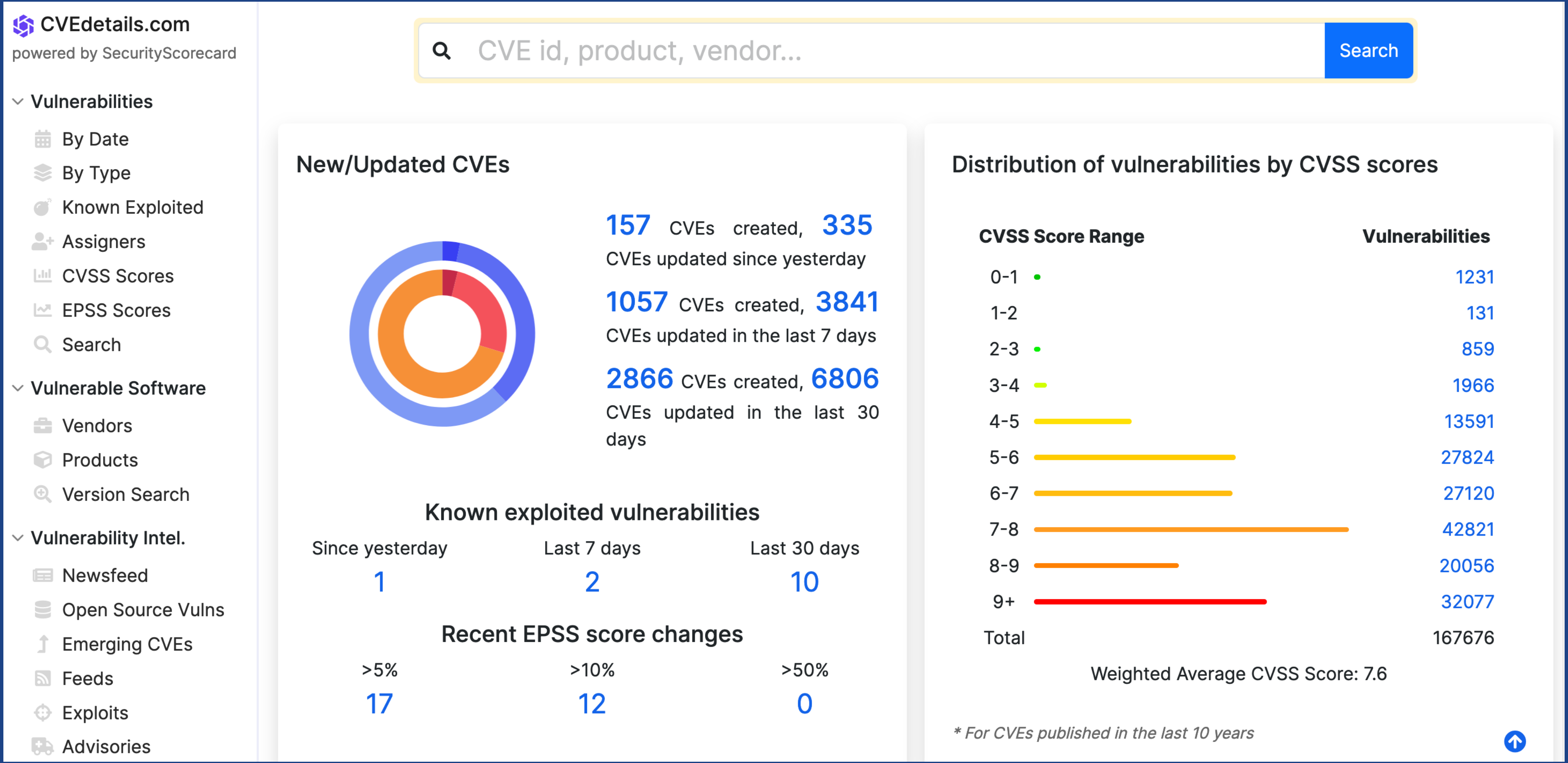
Description

Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1) JNDI features used in configuration, log messages, and parameters do not protect against attacker controlled LDAP and other JNDI related endpoints. An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers when message lookup substitution is enabled. From log4j 2.15.0, this behavior has been disabled by default. From version 2.16.0 (along with 2.12.2, 2.12.3, and 2.3.1), this functionality has been completely removed. Note that this vulnerability is specific to log4j-core and does not affect log4net, log4cxx, or other Apache Logging Services projects.

QUICK INFO

CVE Dictionary Entry:
CVE-2021-44228
NVD Published Date:
12/10/2021
NVD Last Modified:
11/06/2023
Source:
Apache Software Foundation

https://nvd.nist.gov/vuln/detail/CVE-2021-44228
https://www.kb.cert.org/vuls/id/930724



CVEdetails.com
powered by SecurityScorecard

Search: CVE id, product, vendor...

- Vulnerabilities
  - By Date
  - By Type
  - Known Exploited
  - Assigners
  - CVSS Scores
  - EPSS Scores
  - Search
- Vulnerable Software
  - Vendors
  - Products
  - Version Search
- Vulnerability Intel.
  - Newsfeed
  - Open Source Vulns
  - Emerging CVEs
  - Feeds
  - Exploits
  - Advisories

New/Updated CVEs

157 CVEs created, 335 CVEs updated since yesterday

1057 CVEs created, 3841 CVEs updated in the last 7 days

2866 CVEs created, 6806 CVEs updated in the last 30 days

Known exploited vulnerabilities

Since yesterday	Last 7 days	Last 30 days
1	2	10

Recent EPSS score changes

>5%	>10%	>50%
17	12	0

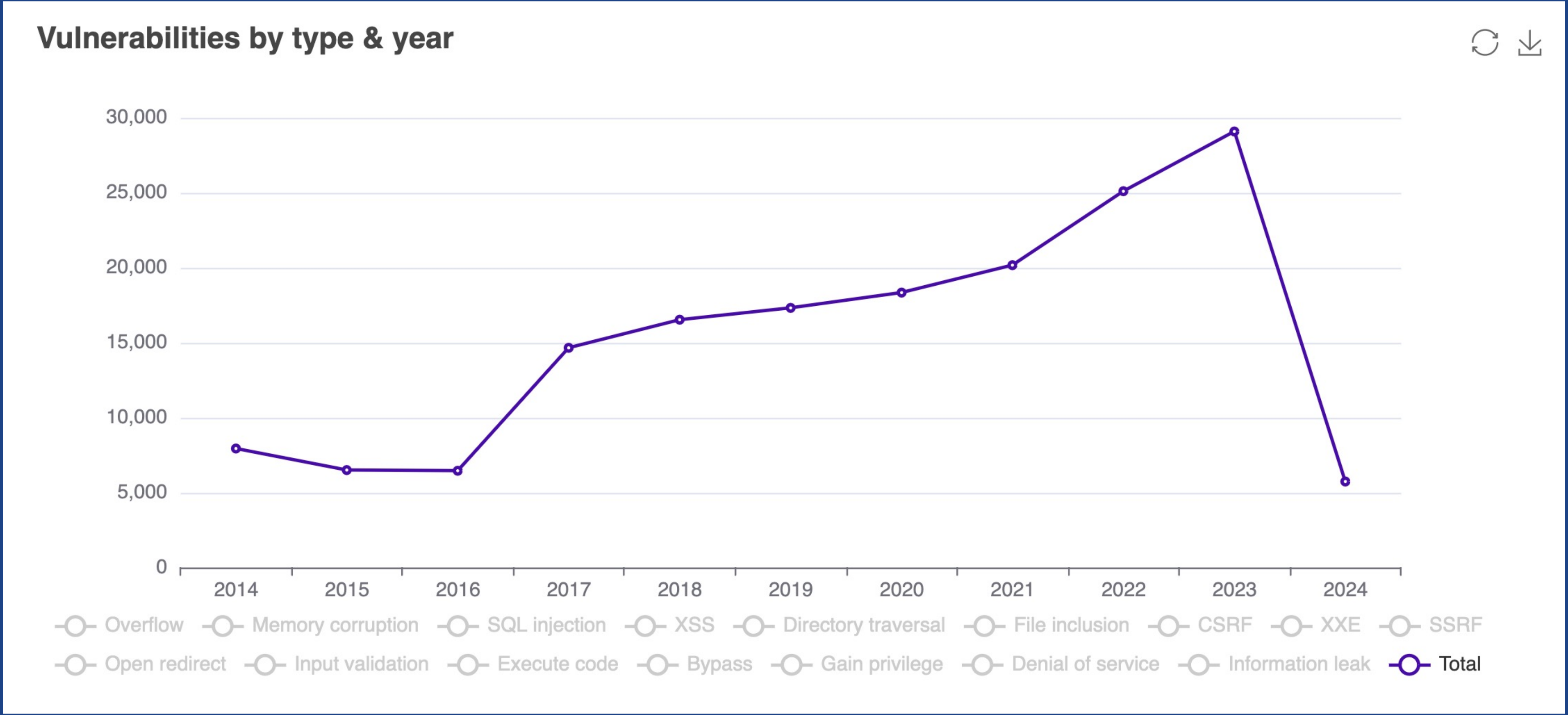
Distribution of vulnerabilities by CVSS scores

CVSS Score Range	Vulnerabilities
0-1	1231
1-2	131
2-3	859
3-4	1966
4-5	13591
5-6	27824
6-7	27120
7-8	42821
8-9	20056
9+	32077
Total	167676

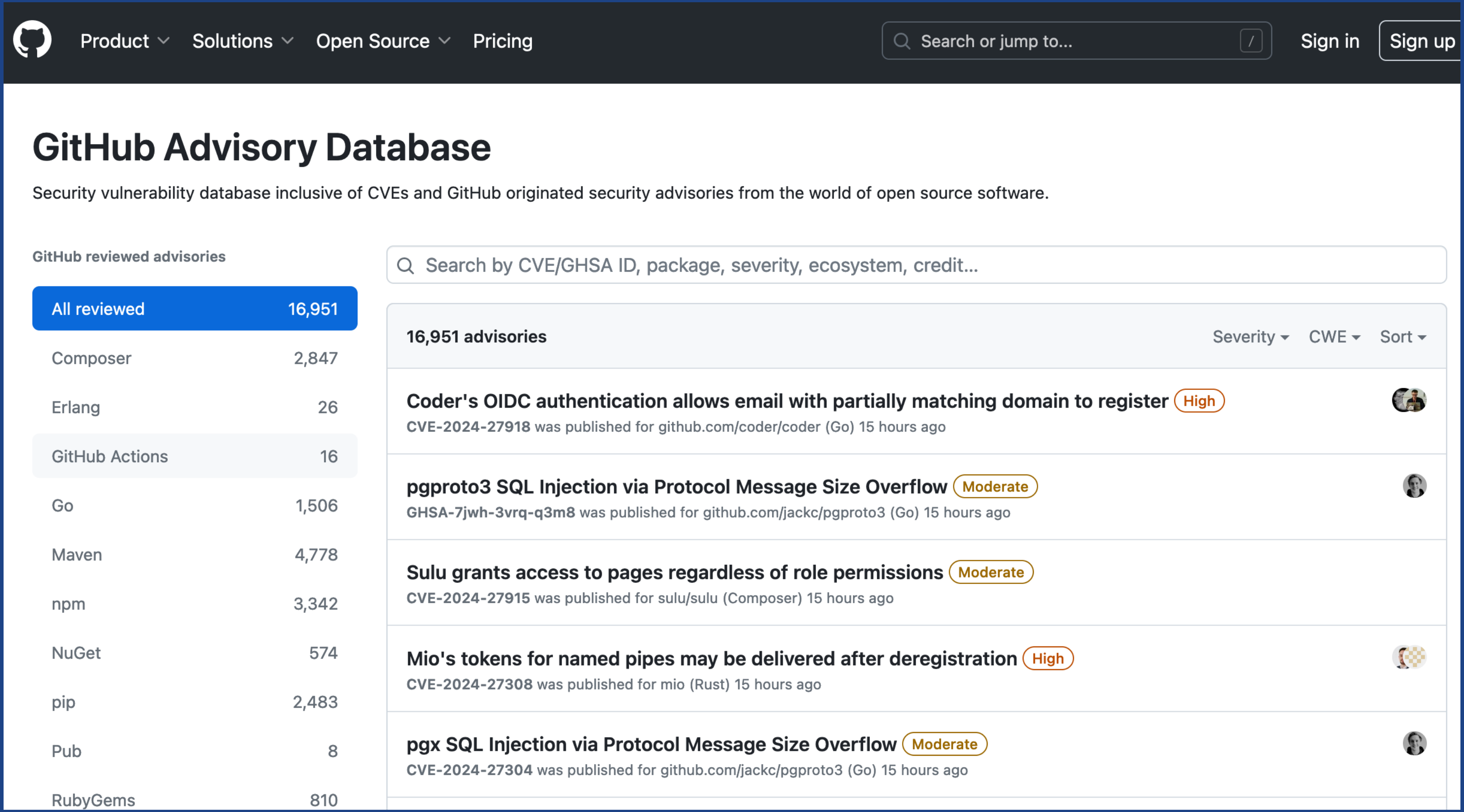
Weighted Average CVSS Score: 7.6

** For CVEs published in the last 10 years*

https://www.cvedetails.com/



https://www.cvedetails.com/



GitHub Advisory Database

Security vulnerability database inclusive of CVEs and GitHub originated security advisories from the world of open source software.

GitHub reviewed advisories

All reviewed	16,951
Composer	2,847
Erlang	26
GitHub Actions	16
Go	1,506
Maven	4,778
npm	3,342
NuGet	574
pip	2,483
Pub	8
RubyGems	810

Search by CVE/GHSA ID, package, severity, ecosystem, credit...

16,951 advisories Severity CWE Sort

Coder's OIDC authentication allows email with partially matching domain to register High
CVE-2024-27918 was published for github.com/coder/coder (Go) 15 hours ago

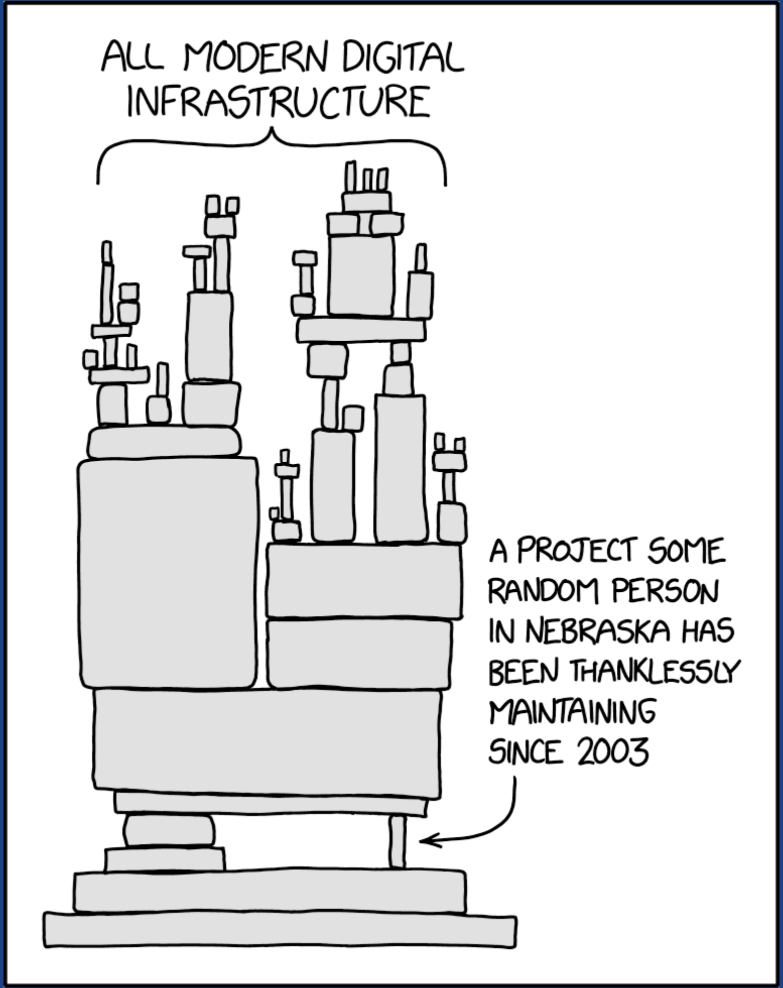
pgproto3 SQL Injection via Protocol Message Size Overflow Moderate
GHSA-7jwh-3vrq-q3m8 was published for github.com/jackc/pgproto3 (Go) 15 hours ago

Sulu grants access to pages regardless of role permissions Moderate
CVE-2024-27915 was published for sulu/sulu (Composer) 15 hours ago

Mio's tokens for named pipes may be delivered after deregistration High
CVE-2024-27308 was published for mio (Rust) 15 hours ago

pgx SQL Injection via Protocol Message Size Overflow Moderate
CVE-2024-27304 was published for github.com/jackc/pgproto3 (Go) 15 hours ago

What happens after discovery?


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

“Dependency”: https://xkcd.com/2347/

Say you find a vulnerability. Do you….

- Tell the world immediately so everyone knows about the problem

- Report to developers so they can fix it before going public

Say you find a vulnerability. Do you….

- Tell the world immediately so everyone knows about the problem

  => Full disclosure

- Report to developers so they can fix it before going public

  => Coordinated disclosure

Say you find a vulnerability. Do you….

- Tell the world immediately so everyone knows about the problem

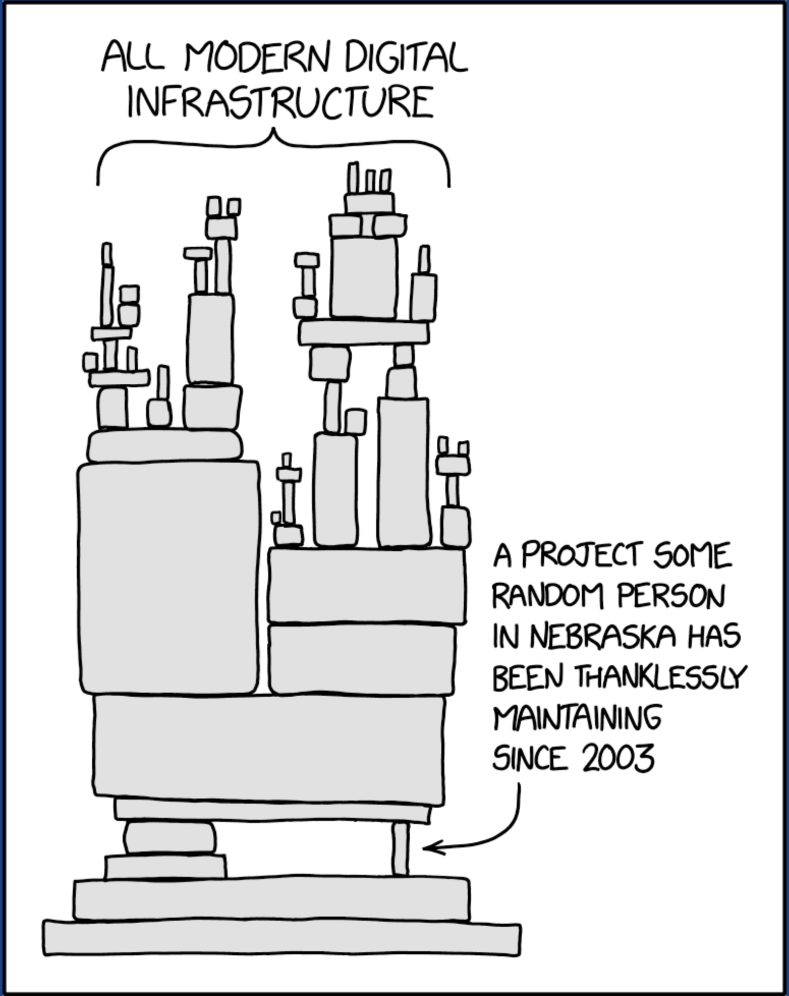
  => Full disclosure

- Report to developers so they can fix it before going public

  => Coordinated disclosure

- Use or sell it for profit

  => Zero-days…


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

"Dependency": https://xkcd.com/2347/

Coordinated disclosure in practice

- Usually, report vulnerability privately to software maintainer first

- ”Embargo” period where discussion is private => software companies ideally coordinate to push fixes ASAP

- Go public once once fixes/mitigations are available

Coordinated disclosure in practice

- Usually, report vulnerability privately to software maintainer first
- ”Embargo” period where discussion is private => software companies ideally coordinate to push fixes ASAP
- Go public once once fixes/mitigations are available

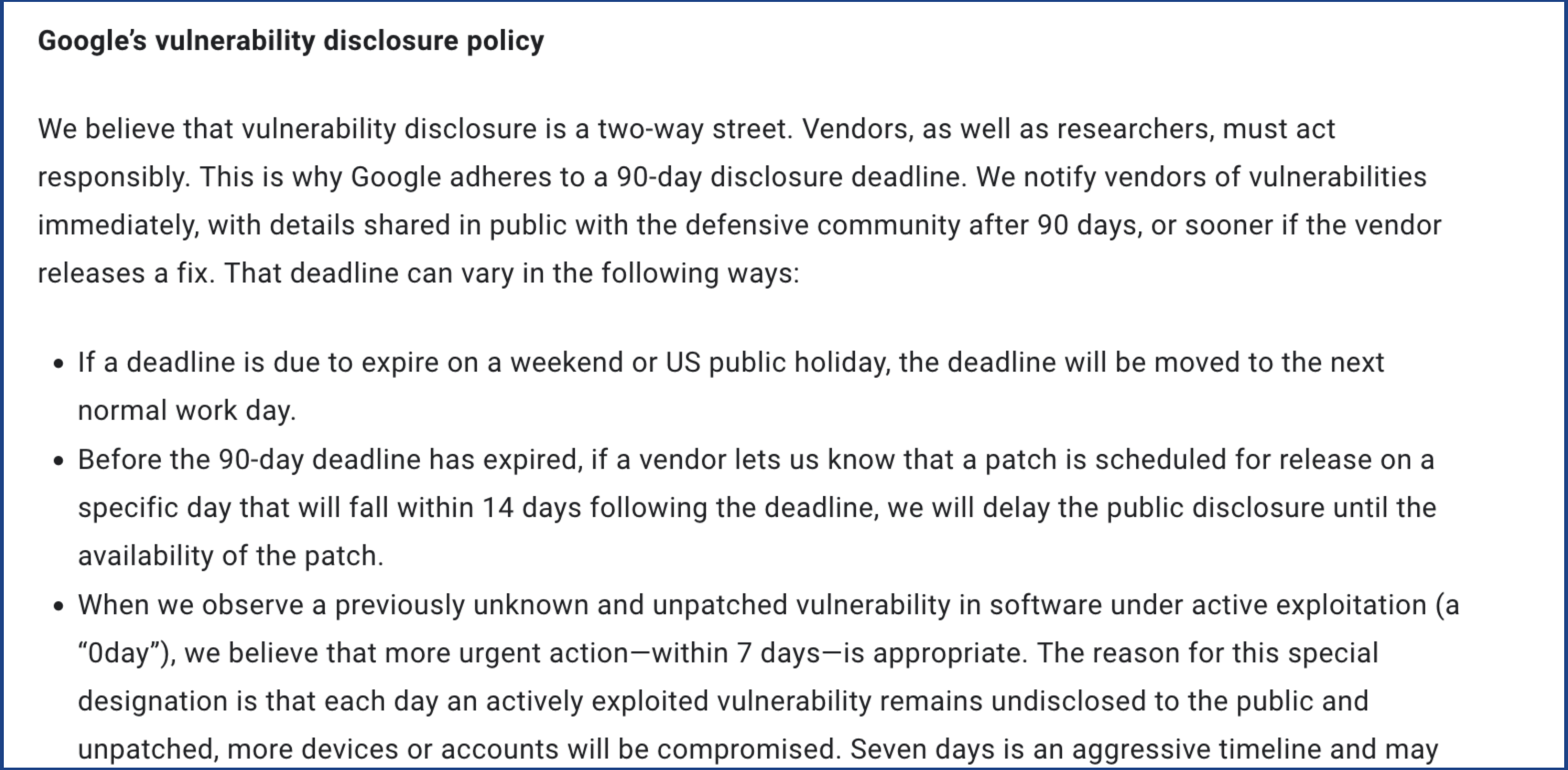
Problems?

Coordinated disclosure in practice

- Usually, report vulnerability privately to software maintainer first
- ”Embargo” period where discussion is private => software companies ideally coordinate to push fixes ASAP
- Go public once once fixes/mitigations are available

=> How to incentivize?
=> How to keep companies from stalling?

Google's Project Zero



Google's vulnerability disclosure policy

We believe that vulnerability disclosure is a two-way street. Vendors, as well as researchers, must act responsibly. This is why Google adheres to a 90-day disclosure deadline. We notify vendors of vulnerabilities immediately, with details shared in public with the defensive community after 90 days, or sooner if the vendor releases a fix. That deadline can vary in the following ways:

- If a deadline is due to expire on a weekend or US public holiday, the deadline will be moved to the next normal work day.
- Before the 90-day deadline has expired, if a vendor lets us know that a patch is scheduled for release on a specific day that will fall within 14 days following the deadline, we will delay the public disclosure until the availability of the patch.
- When we observe a previously unknown and unpatched vulnerability in software under active exploitation (a "0day"), we believe that more urgent action—within 7 days—is appropriate. The reason for this special designation is that each day an actively exploited vulnerability remains undisclosed to the public and unpatched, more devices or accounts will be compromised. Seven days is an aggressive timeline and may

https://about.google/appsecurity/

Some strategies

- Open source: many ”eyes” on the same project => more rigorous auditing for bugs

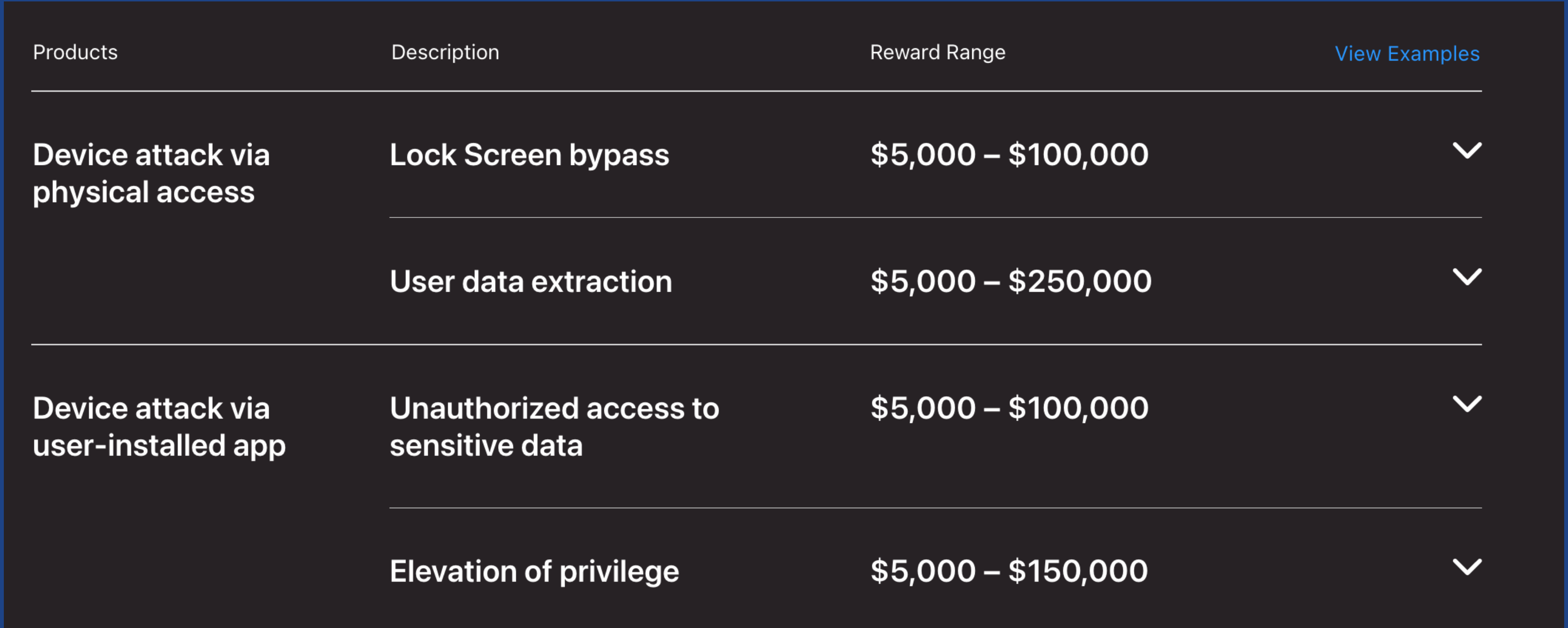
- Incident response plans: make dealing with vulns part of the software development process

- Bug bounties: incentives ($$$) from companies to report bugs to them first => Usually requires coordinated disclosure

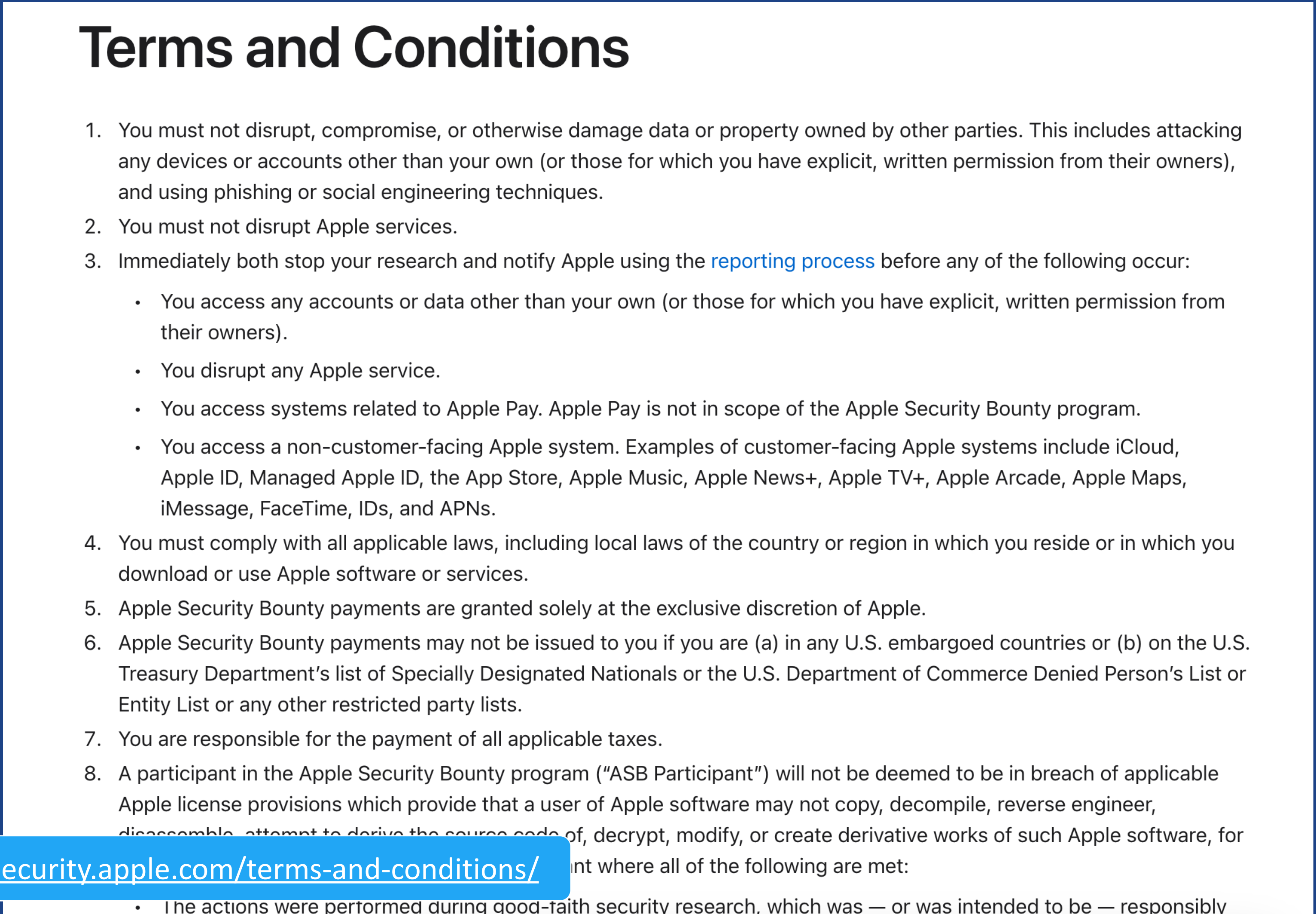
Bounty

Report Categories Guidelines

Apple Security Bounty
Categories



Products	Description	Reward Range	View Examples
Device attack via physical access	Lock Screen bypass	$5,000 – $100,000	
	User data extraction	$5,000 – $250,000	
Device attack via user-installed app	Unauthorized access to sensitive data	$5,000 – $100,000	
	Elevation of privilege	$5,000 – $150,000	



Terms and Conditions

1. You must not disrupt, compromise, or otherwise damage data or property owned by other parties. This includes attacking any devices or accounts other than your own (or those for which you have explicit, written permission from their owners), and using phishing or social engineering techniques.
2. You must not disrupt Apple services.
3. Immediately both stop your research and notify Apple using the reporting process before any of the following occur:
   - You access any accounts or data other than your own (or those for which you have explicit, written permission from their owners).
   - You disrupt any Apple service.
   - You access systems related to Apple Pay. Apple Pay is not in scope of the Apple Security Bounty program.
   - You access a non-customer-facing Apple system. Examples of customer-facing Apple systems include iCloud, Apple ID, Managed Apple ID, the App Store, Apple Music, Apple News+, Apple TV+, Apple Arcade, Apple Maps, iMessage, FaceTime, IDs, and APNs.
4. You must comply with all applicable laws, including local laws of the country or region in which you reside or in which you download or use Apple software or services.
5. Apple Security Bounty payments are granted solely at the exclusive discretion of Apple.
6. Apple Security Bounty payments may not be issued to you if you are (a) in any U.S. embargoed countries or (b) on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Department of Commerce Denied Person's List or Entity List or any other restricted party lists.
7. You are responsible for the payment of all applicable taxes.
8. A participant in the Apple Security Bounty program ("ASB Participant") will not be deemed to be in breach of applicable Apple license provisions which provide that a user of Apple software may not copy, decompile, reverse engineer, disassemble, attempt to derive the source code of, decrypt, modify, or create derivative works of such Apple software, for

https://security.apple.com/terms-and-conditions/

nt where all of the following are met:

   - The actions were performed during good-faith security research, which was — or was intended to be — responsibly

Bonus: Flash