Secure Application Design

Common Attacks and Vulnerabilities

Summer 2025

Jakob Heher, www.isec.tugraz.at
he/his

Common Acronyms

- **C**ommon **V**ulnerabilities and **E**xposures identifier
  - Unique identifier for a particular vulnerability or exposure

`CVE-2017-5754 ("Meltdown")`

- **N**ational **V**ulnerability **D**atabase
  - Information about CVEs, affected software configurations, …
- **C**ommon **P**latform **E**numeration
  - Standardized identification scheme for components

`cpe:2.3:h:intel:core_i7:8700k:*:*:*:*:*:*:*`

`cpe:2.3:a:ntp:ntp:4.2.8:p3:*:*:*:*:*:*`

- **C**ommon **W**eakness **E**numeration
  - Standardized identifier for a category of vulnerabilities

`CWE-79 ("Cross-Site Scripting")`

Common Vulnerability Scoring System

- Tries to assign a numeric “badness” score to a vulnerability
  - Combination of different metrics

- **Attack Vector (AV)**
  - **(P)** Physical: requires access to the actual hardware
  - **(L)** Local: requires access to the logical system
  - **(A)** Adjacent: requires network adjacency to the device
  - **(N)** Network: none of the limitations above apply

Common Vulnerability Scoring System

- Tries to assign a numeric “badness” score to a vulnerabilities
  - Combination of different metrics

- **Attack Complexity (AC)**
  - **(H)** High: Attacker must invest some effort, e.g.:
    - Attacker needs environment-specific information *(e.g., UUIDs, sequence numbers)*
    - Attacker success is not guaranteed *(e.g., needs to win a race condition)*
    - Attacker needs to perform additional network exploitation *(e.g., ARP spoofing)*
  - **(L)** Low: Can be easily reproduced at will

Common Vulnerability Scoring System

- Tries to assign a numeric “badness” score to a vulnerabilities
  - Combination of different metrics

- **Privileges Required (PR)**
  - **(H)** High: attacker needs pre-existing administrative access
  - **(L)** Low: attacker needs pre-existing user-level access
  - **(N)** None: attacker does not need to be authorized for access

Common Vulnerability Scoring System

- Tries to assign a numeric “badness” score to a vulnerabilities
  - Combination of different metrics

- **User Interaction required (UI)**
  - **(R)** Required: attacker needs a genuine user to take some action
  - **(N)** None: vulnerability can be exploited without user interaction

Common Vulnerability Scoring System

- Tries to assign a numeric “badness” score to a vulnerabilities
  - Combination of different metrics

- **Scope (S)**
  - **(U)** Unchanged: vulnerable component == impacted component
  - **(C)** Changed: vulnerable component != impacted component

Common Vulnerability Scoring System

- Tries to assign a numeric “badness” score to a vulnerabilities
  - Combination of different metrics

- **Confidentiality (C)**, **Integrity (I)**, **Availability (A)**
  - **(N)** None
  - **(L)** Low
  - **(H)** High

Common Vulnerability Scoring System

- Tries to assign a numeric “badness” score to a vulnerabilities
  - Combination of different metrics

- **Exploit Code Maturity (E)**
  - **(U)** Unproven: no exploit code is available
  - **(P)** Proof-of-Concept: skilled attacker could craft an attack based on a PoC
  - **(F)** Functional: attack scripts that require limited technical expertise exist
  - **(H)** High: fully-automated vulnerability scans & exploits exist

Common Vulnerability Scoring System

- Tries to assign a numeric “badness” score to a vulnerabilities
  - Combination of different metrics

- **Remediation Level (RL)**
  - **(O)** Official Fix: there is a ready-to-apply remediation from the vendor
  - **(T)** Temporary Fix: there is a temporary solution from the vendor
  - **(W)** Workaround: there is an unofficial temporary solution
  - **(U)** Unavailable: there is no solution available

Common Vulnerability Scoring System

- Tries to assign a numeric “badness” score to a vulnerabilities
  - Combination of different metrics

- **Report Confidence (RC)**
  - **(U)** Unknown: there is little understanding of how to trigger the issue
  - **(T)** Reasonable: the issue consistently occurs, but is not well understood
  - **(C)** Confirmed: the issue is consistently reproducible and well-understood

Common Vulnerability Scoring System

- Tries to assign a numeric “badness” score to a vulnerabilities
  - Combination of different metrics

- Scores in range from 0.0 to 10.0
  - ≥ 9.0: Critical
  - ≥ 7.0: High
  - ≥ 4.0: Medium
  - ≥ 0.1: Low

CVE-2014-0160 Detail

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.

Severity

CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

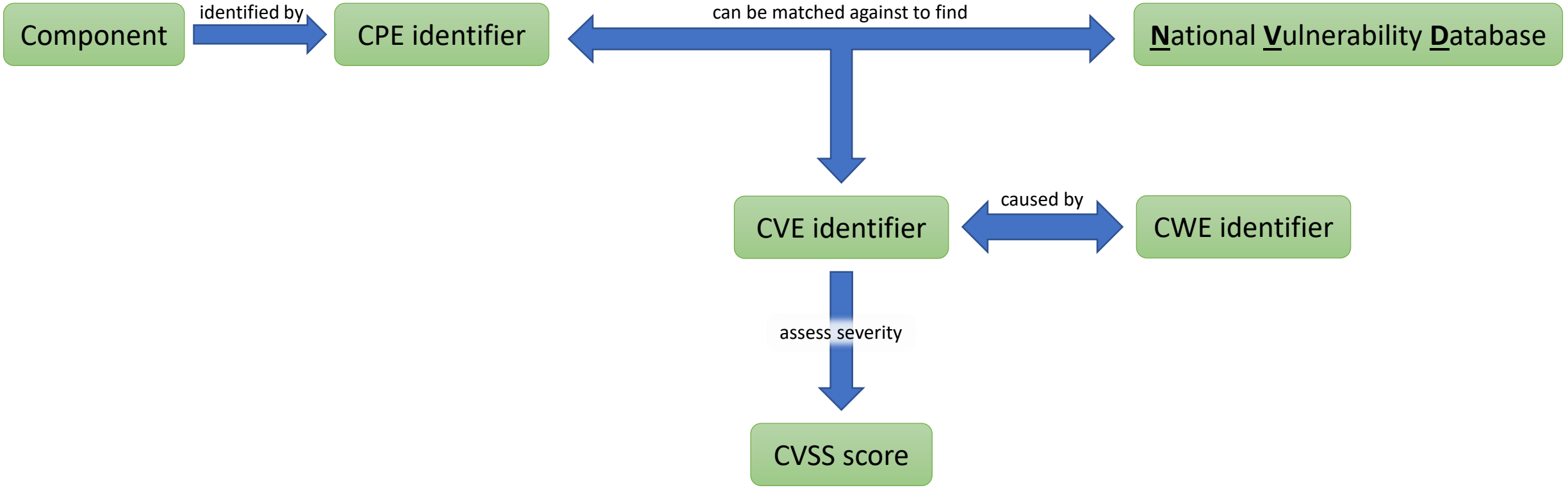
NIST: NVD **Base Score:** 7.5 HIGH **Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.

Evaluator Impact

CVSS V2 scoring evaluates the impact of the vulnerability on the host where the vulnerability is located. When evaluating the impact of this vulnerability to your organization, take into account the nature of the data that is being protected and act according to your organization's risk acceptance. While CVE-2014-0160 does not allow unrestricted access to memory on the targeted host, a successful exploit does leak information from memory locations which have the potential to contain particularly sensitive information, e.g., cryptographic keys and passwords. Theft of this information could enable other attacks on the information system, the impact of which would depend on the sensitivity of the data and functions of that system.


Component
identified by
CPE identifier
can be matched against to find
National Vulnerability Database
CVE identifier
caused by
CWE identifier
assess severity
CVSS score

OWASP Top Ten

- Compiled by the Open Worldwide Application Security Project
- Top Ten “most important” vulnerability categories
  - Top 8 by incidence rate
  - + 2 by community survey
  - = 10 mistakes that should be on your radar
- Data contributed by many pen testing organizations

Broken Access Control

#1 -- OWASP Top 10 2021

Avg. IR: 3.81% of tested applications

CWE-22 Path Traversal

- **`GET /../../../etc/shadow HTTP/1.1`**

- Naïve concatenation leads to a vulnerability:
  - **`/var/www/html/../../../etc/shadow`**

- Possible Solutions:
  a) Reject any path containing `../`?
     - Careful with sanitization: easy to get wrong (`.../...//` ---> `....//` ---> `../`)
  b) Normalize path & check that it starts with intended root directory

CWE-158 Null Byte Injection

- **`/readTextFile.php?name=secret_key.bin%00`**

- This bypasses a fixed suffix:
  - **`./secret_key.bin`**NUL**`.txt`**

- Possible Solutions:
  a) Maintain a whitelist of valid inputs
  b) Normalize path & check it ends with the intended file extension

CWE-59 Improper Link Resolution

- What's a file (soft) link, actually?
  - It depends on your file system and operating system!
- Hyperlinks might point outside the "intended" directory
  - **`/var/www/html/innocent.txt`** might alias **`/etc/shadow`**
- Ways to introduce hyperlinks:
  - File upload
  - Archive extraction
  - …

CWE-352 Cross-Site Request Forgery

- Attackers may cause a genuine user to perform a web query
  - **`<img src="https://victim.org/createAccount.php?name=Eve&pass=evul">`**

- Countermeasures:
  - **DO NOT ALLOW `GET` REQUESTS TO MODIFY THE SERVER'S STATE**
    - GET requests are less restricted in the browser
  - Set a **`SameSite=Lax`** or **`SameSite=Strict`** option on the session cookie
    - Set to **`Lax`** by default in modern browsers
  - Add a randomly-generated CSRF canary as a hidden parameter
    - Attackers would have to "blindly" guess it

CWE-425 Direct Request

- Attackers can enter arbitrary web URLs into their browser

- Example vulnerable flow:
  1. User navigates to **`https://genuine.org/login`**
  2. Username & Password are requested & checked
  3. User is redirected to **`https://genuine.org/admin`**

- The attacker just enters **`https://genuine.org/admin`** directly
  - Does this page check that the user is authorized?

Cryptographic Failures

#2 -- OWASP Top 10 2021

Avg. IR: 4.49% of tested applications

CWE-319 Cleartext Transmission of Sensitive Data

- Most communication channels are inherently untrustworthy
  - Network connections can be intercepted at a variety of stages
- Communicate *only* using encrypted and authenticated channels!
  - Do *not* fall back to insecure technologies on the client!
    - E.g., if an attacker blocks HTTPS traffic, don't fall back to HTTP mode!
  - Do *not* support insecure technologies on the server!
- The one mode an attacker cannot reach is one that is not supported!

CWE-338 Use of Weak Pseudo-Random Numbers

- "True Randomness" in computers is very hard
- We use *pseudo-random number generators* (PRNGs) instead

- Not all of them are suited for cryptography!
  - Cryptographically Secure Pseudo-Random Number Generators (CSPRNGs)
  - Resistant to reverse engineering and cryptanalysis

- Make sure you carefully check what kind of RNG a language offers!
  - E.g., Python's **`random`** vs **`secrets`** modules

CWE-757 Algorithm Downgrade Possible

- Problem: different software supports different cryptosystems
- Idea: negotiate the used cryptosystem during handshake

- Issue: active attackers can manipulate the handshake
  - This can cause weak encryption to be used!

- Possible Solutions:
  - Don’t support weak legacy encryption modes
  - Validate the integrity of the entire handshake afterwards

Other Cryptographic Failures

- **CWE-322** Key Exchange without Entity Authentication
- **CWE-323** Reusing a Nonce/Key Pair
- **CWE-328** Using a Weak Hash
- **CWE-347** Improper Verification of Cryptographic Signatures
- **CWE-523** Unprotected Transport of Credentials
- **CWE-759** Using a One-Way Hash without Salt
- **CWE-916** Password Hash with Insufficient Computational Effort

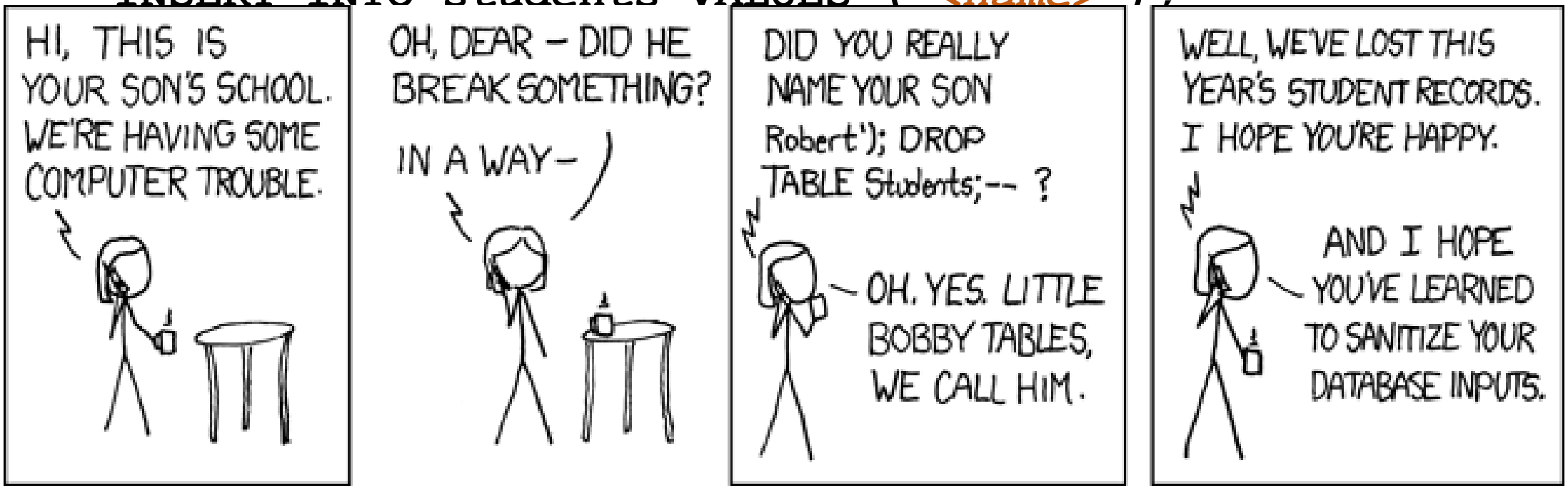
Injection

#3 -- OWASP Top 10 2021

Avg. IR: 3.37% of tested applications

CWE-89 SQL Injection

- Attacker's input is used as part of a SQL query string
  - `INSERT INTO students VALUES ('<name>');`



CWE-89 SQL Injection

- Attacker's input is used as part of a SQL query string
  - **`INSERT INTO students VALUES ('Robert');`**
    `DROP TABLE Students;`
    `--');`

- Solutions:
  - Input validation/sanitization
    - But: what even *is* a valid name?
    - Blacklisting risks missing attack inputs; whitelisting risks being discriminatory
  - Better: Parametrized queries
    - Separate the (static) query string from the user-supplied parameters!

CWE-78 OS Command Injection

- **`ls -l /home/`*`<username>`***
- What if I call my account **`; rm -rf -no-preserve-root /`**?
- Solutions:
  - Input validation/sanitization
  - Better: dedicated parameter arguments
    - **`execve()`** instead of **`system()`**

CWE-79 Cross-Site Scripting

- `<p><?php echo $forum_post; ?></p>`
- What if I post `<script>stealYourData();</script>`?

- Solutions:
  - Input validation/sanitization
  - Better: retrieve data out-of-band (JSON) and insert via `.innerText`
  - Defense in depth: Content Security Policy (CSP)

Other kinds of injection

- **CWE-90** LDAP Query Injection
- **CWE-95** Injection into Dynamic Evaluation Call
- **CWE-97** Injection of Server-Side Includes into Web Page
- **CWE-470** Injection into Class/Code Selector
- **CWE-643** Injection into XPath Expression
- **CWE-652** Injection into XQuery Expression

Insecure Design

#4 -- OWASP Top 10 2021

Avg. IR: 3.00% of tested applications

Define Trust Boundaries

- Given data can be *trustworthy* or *untrusted*
  - User input is *untrusted*
  - After validation, it might become *trustworthy*

- Clearly delineate between these states
  - Specific classes for untrusted data?
  - Naming conventions for untrusted variables?
  - User input only accessible through specific methods?

Segment Critical Components

- Identify which parts of your program are security critical
  - Authentication?
  - Cryptographic operations?
  - User input processing?
- Keep critical parts separate & compact
  - Easy to verify
  - Easy to test

What Would Attackers Do?

- Constantly reflect on possible attacks against your system
  - This requires you to understand attacks!

- What inputs might an attacker provide?
  - Be specific!
- How does your system protect against them?
  - How might an attacker work around the protections? Iterate!

Want To Know More?

- 705.022 Secure Software Development
  - https://www.iaik.tugraz.at/ssd
  - offered in winter semester

- Subjects covered include:
  - Threat Modeling
  - Static Analysis Methods
  - Defensive Programming Methods
  - and many more…

Security Misconfiguration

#5 -- OWASP Top 10 2021

Avg. IR: 4.51% of tested applications

CWE-219 Sensitive Data Stored Under Web Root

- **`GET /database.config HTTP/1.1`**

- Web servers will gladly serve *any* file that is in their document root
  - If users should not see it, don't put it in the document root!

- Try to avoid relying on **`.htaccess`**, extension-specific handlers, etc.
  - Server upgrade breaks **`mod_php`**, hardcoded passwords served in plain?

CWE-209 Sensitive Information In Error Messages

- Sensitive information is unintentionally shown to a user
- Examples:
  - Exception messages including SQL query structure
  - Stack traces showing source code with configuration file path or password
- Some Solutions:
  - Disable user-visible error logging in production environments
    - Log internally and expose only an opaque reference number to the user
  - Consider what information to include in error messaging

CWE-614 & CWE-1004 Inappropriate Cookie Permissions

- HTTP cookies are commonly used for re-authentication
- Cookies permit a number of *flags* to be set

- These flags should *always* be enabled for session cookies:
  - **`HttpOnly`**: not accessible to JavaScript
  - **`Secure`**: only sent via HTTPS
  - **`SameSite`**: not sent for requests originated by external websites

- Check that your session framework sets them!

Vulnerable & Outdated Components

#6 -- OWASP Top 10 2021

Avg. IR: 8.77% of tested applications

WIRED

LILY HAY NEWMAN SECURITY 12.10.2021 02:54 PM

'The Internet Is on Fire'

A vulnerability in the Log4j logging framework has security teams scrambling to put in a fix.

The Washington Post
Democracy Dies in Darkness

Technology

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

MENU LAWFARE

CYBERSECURITY

What's the Deal with the Log4Shell Security Nightmare?

By **Nicholas Weaver** Friday, December 10, 2021, 4:38 PM

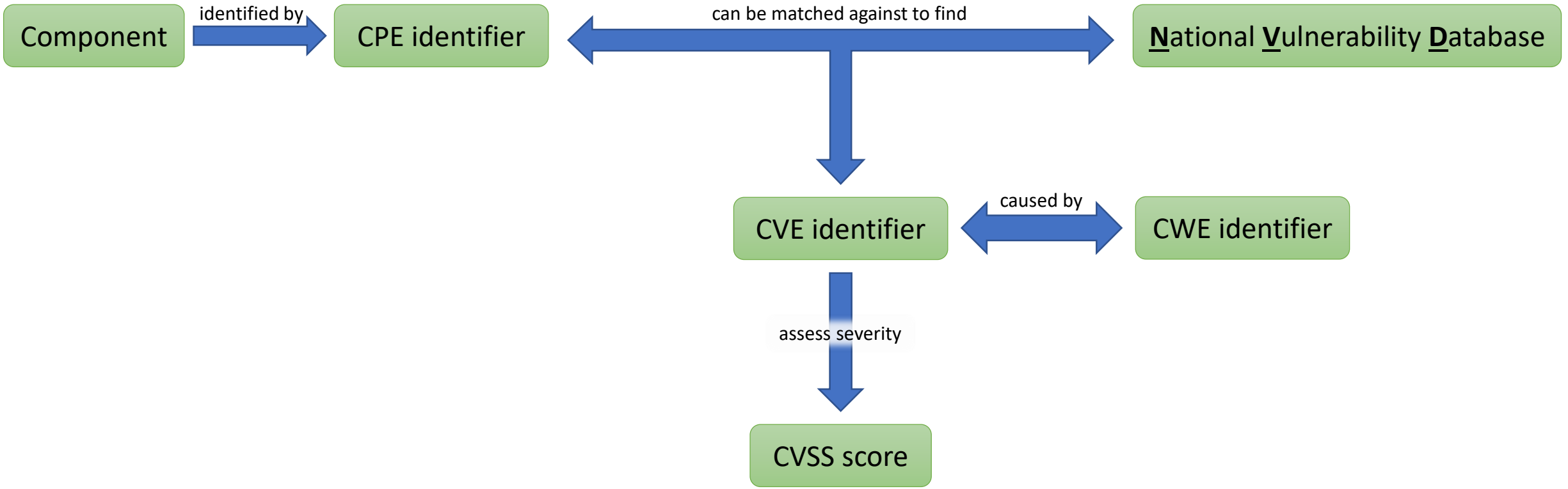
How were we doing three years later?

- In 2024, **17%** of log4j downloads were vulnerable versions!

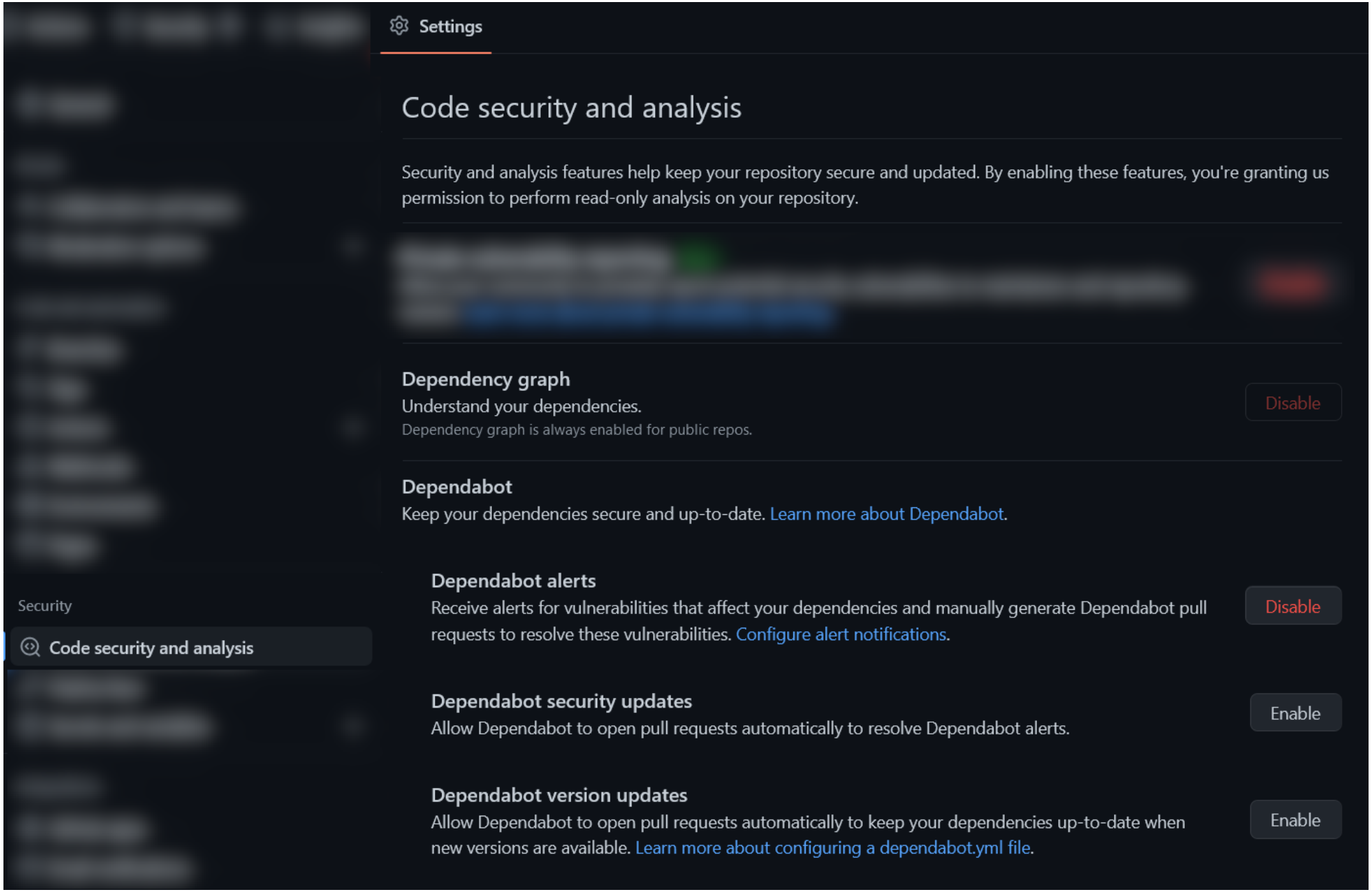
- Dependencies …
  - … of dependencies …
    - … of dependencies …
      - … of dependencies …
        - … of dependencies …
          - … of dependencies …
            - … of dependencies …
              - … of dependencies …
                - … of dependencies …
                  - … of dependencies …

- Do you know whether your software is vulnerable?

Automated Dependency Checking



- Do this for every (recursive) dependency


Settings
Code security and analysis
Security and analysis features help keep your repository secure and updated. By enabling these features, you're granting us permission to perform read-only analysis on your repository.
Dependency graph
Understand your dependencies.
Dependency graph is always enabled for public repos.
Disable
Dependabot
Keep your dependencies secure and up-to-date. Learn more about Dependabot.
Dependabot alerts
Receive alerts for vulnerabilities that affect your dependencies and manually generate Dependabot pull requests to resolve these vulnerabilities. Configure alert notifications.
Disable
Dependabot security updates
Allow Dependabot to open pull requests automatically to resolve Dependabot alerts.
Enable
Dependabot version updates
Allow Dependabot to open pull requests automatically to keep your dependencies up-to-date when new versions are available. Learn more about configuring a dependabot.yml file.
Enable
Security
Code security and analysis

Identification & Authentication

#7 -- OWASP Top 10 2021

Avg. IR: 2.55% of tested applications

CWE-307 Excessive Authentication Attempts Possible

- Users generally won't fail hundreds of logins
- Common brute-force attacks require $2^{something}$ attempts

- You probably shouldn't allow $2^{something}$ attempts

- Possible implementations:
  - Lock or timeout account after some number of failed logins
  - Require computationally intensive task from user

CWE-620 Unverified Password Change

- An attacker might gain access to a user's session
  - (This is bad)
- An attacker might want to use this access to lock the user out
  - (This would be worse)
- Changes to authentication factors must require re-authentication!
- Examples:
  - Require re-entry of current password to change password
  - Require entry of TOTP codes to remove TOTP

CWE-640 Weak Recovery Factors

- Authentication is only as secure as its *weakest* permitted combination
- Who knows your mother's maiden name?
  - Your first pet's name? Your elementary school? Your favorite food?
- Account recovery factors need to actually be secure!
- Additional considerations:
  - Highly visible out-of-band notifications to the genuine user
  - Time delays to allow the genuine user to notice and intervene

Software & Data Integrity Failures

#8 -- OWASP Top 10 2021

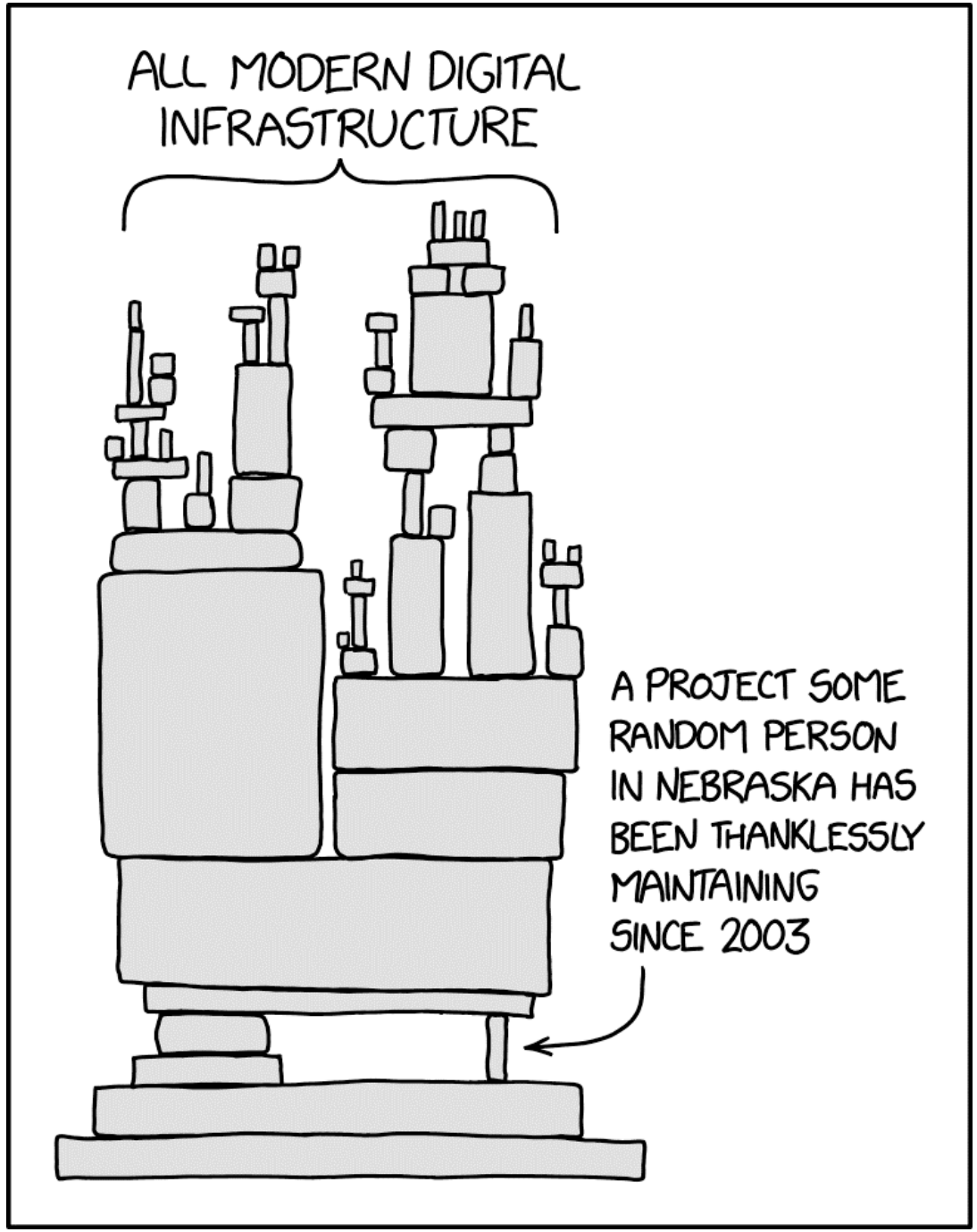
Avg. IR: 2.05% of tested applications

Dependencies Are Not Simple

- **`pip install awesome-package`**

- What just happened?
  - **`pip`** looks in a public list of packages for “awesome-package” and installs it
  - The installation process runs code on your local machine
  - The installed sources will ship with your packaged software

- … sounds scary, right?



Dependencies Are Not Simple

- Here's a few things that can go wrong:
  - The dependency author might've made a genuine mistake
  - The dependency maintainer can be compromised
  - Control of the dependency might transfer to a new maintainer
  - Packages might not come from the repository you expect

- Some solutions:
  - Only use local repository of approved dependency packages
  - Pin specific versions in specific repositories where possible

Logging & Monitoring Failures

#9 -- OWASP Top 10 2021

3rd place Community Survey result

Application Monitoring

- Are we under attack?
  - Attacks commonly come with non-standard usage patterns
  - Monitoring might be able to alert you!

- Examples:
  - There are repeated authentication failures across different accounts
  - A single admin starts resetting hundreds of account passwords
  - Someone starts editing thousands of documents
  - Microsoft Word spawns a network-enabled subprocess

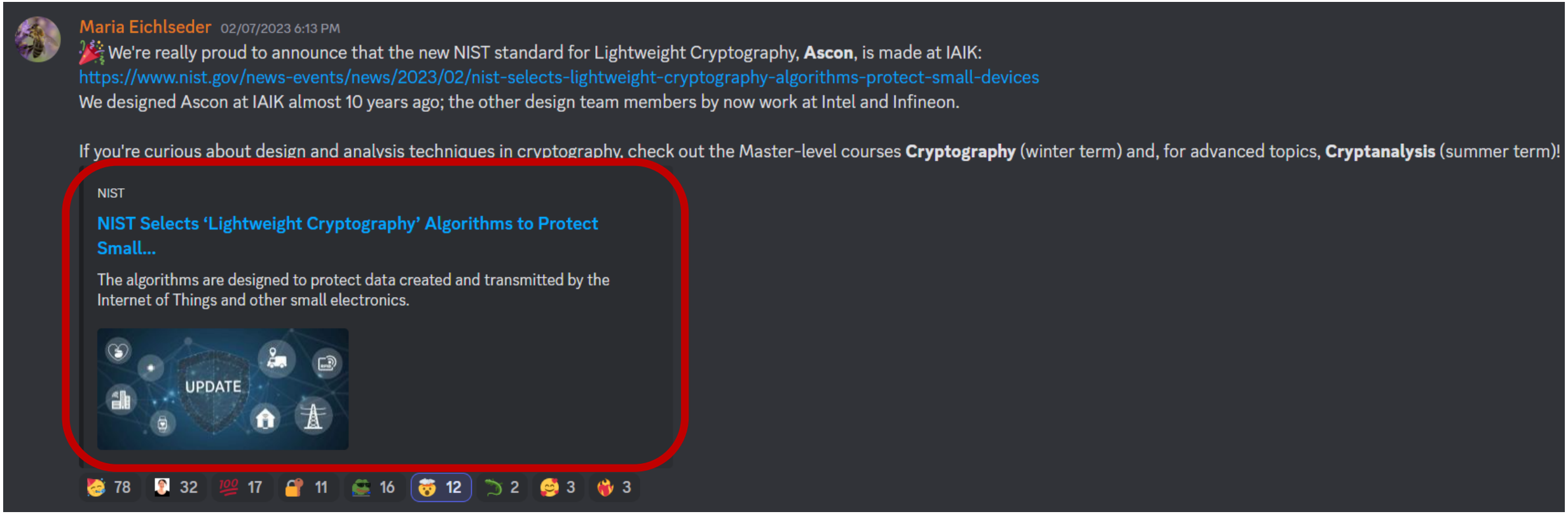
Logging & Forensics

- We've been hacked, now what?
  - Find out what happened so it can't happen again!
- Worst case: we've been hacked, and we don't know how
- Aggressive logging may cause attackers to leave traces
  - Make sure the attackers can't modify the logs: append-only logging

Server-Side Request Forgery

#10 -- OWASP Top 10 2021

1st place Community Survey result



- We can make Discord's servers send a query to any URL!

A Few Other Relevant Contexts

- Server sending email for account verification
- Upload custom avatar via URL
- Callback URLs for APIs
- XML/SVG parsers
- HTTP redirects
- DNS lookups triggered by the above!

- Do we get to see the response? It depends!

Some Interesting Things One Might Do

- Unmask hidden back-end servers
  - Cloudflare-proxied websites
  - Tor Hidden Services

- Employ more esoteric URL schemes
  - **`file://`** can retrieve local files
  - **`gopher://`** can send arbitrary bytes to arbitrary ports
  - …

- These requests originate from the application server, not the attacker!

SSRF countermeasures

- Avoid access to user-controlled URLs wherever possible
  - e.g., make the user's browser download & re-upload images

- If you can't, then:
  - Whitelist permissible protocols
  - Blacklist private IP address ranges
  - Filter DNS queries
  - Don't follow HTTP redirects
  - Isolate servers from each other

Recap

- **CVE numbers** identify individual vulnerabilities or exposures
- **CVSS scores** try to quantify the “badness” of a vulnerability
- **CPE identifiers** identify a particular component version
- The **NVD** is one widely-used database of CVEs and affected CPEs
  - **Automated tools** can perform cross-referencing of this data!

Recap – OWASP Top Ten 2021

1. Broken Access Control
2. Cryptographic Failures
3. Injection
4. Insecure Design
5. Security Misconfiguration
6. Vulnerable and Outdated Components
7. Identification and Authentication Failures
8. Software and Data Integrity Failures
9. Security Logging and Monitoring Failures
10. Server Side Request Forgery