#1 - Vulnerabilities

Vulnerabilities

Is a weakness in a system (software, hardware…)

- It's a broad concept as a vulnerability can derive from many things

A vulnerability allows an attacker to violate a reasonable security policy for that system

- Policies define how a system should behave.
- Examples:
  - Wheels will turn left only when steering wheel turns left
  - Phones will only allow access to its owner
  - Programs will only run code inserted by its original developer

- User roles
- access levels

Vulnerabilities

Vulnerability number always increases as software grows

- It’s inherent to the increased complexity, interactions, development process
- Also, they do not disappear
- Software is updated with fixes, but older software is still vulnerable

Vulnerabilities

Vulnerabilities are states in a computing system that either allows an attacker to:

- execute commands as another user
- access data that is contrary to the specified access restrictions for that data
- pose as another entity
- conduct a denial of service (DoS) (affect availability)

Threats


Malware
Threats against data
Social Engineering threats
Disinformation Misinformation
FAKE
ENISA THREAT LANDSCAPE
Supply-chain attacks
Ransomware
Threats against availability: Denial of Service
Threats against availability: Internet threats

CIA triad → Model for security systems

Confidentiality

- Whether information is disclosed to others
  - ↳ 2 factor autentication

Integrity

- Whether data contents and formats are kept safe from modifications
  - ↳ Encryption

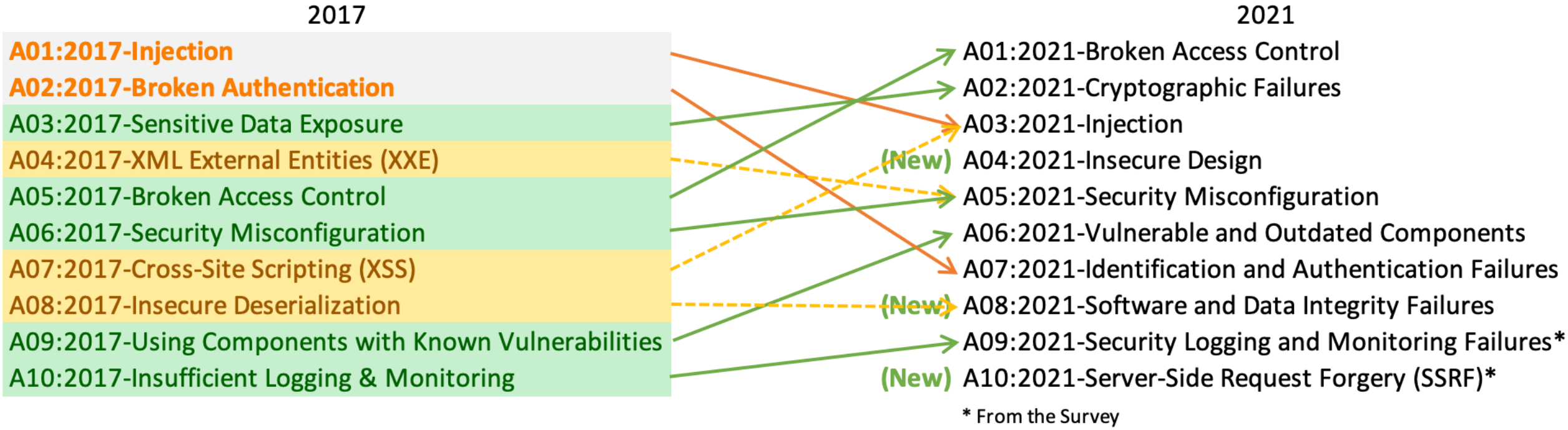
Availability

- Whether system performance is degraded
  - ↳ safe against DOS

Vulnerability sources – OWASP Top 10 (Web)

1. **Injection**
2. **Broken Authentication**
3. Sensitive Data Exposure
4. **XML External Entities (XXE)**
5. **Broken Access control**
6. Security misconfigurations
7. **Cross Site Scripting (XSS)**
8. **Insecure Deserialization**
9. Using Components with known vulns.
10. **Insufficient logging and monitoring**

Vulnerability sources – OWASP Top 10 (Web)



Vulnerability sources – 7 Pernicious Kingdoms

1. **Input Validation and Representation**
2. **API Abuse**
3. **Security Features**
4. **Time and State**
5. **Errors**
6. **Code Quality**
7. **Encapsulation**

*. **Environment**

K. Tsipenyuk, B. Chess and G. McGraw, "Seven pernicious kingdoms: a taxonomy of software security errors," in IEEE Security & Privacy, vol. 3, no. 6, pp. 81-84, Nov.-Dec. 2005, doi: 10.1109/MSP.2005.159.

Vulnerability sources - CWE

Vulnerabilities may exist due to Bugs or Faults

- Bug is an error in the implementation of a software
- Fault is a design or architectural error

CWE - Common Weaknesses Enumeration

- Extensive (891) list of anti-patterns that may lead to insecure systems
- Arranged in a tree, with examples in multiple languages

CWE-348: Use of Less Trusted Source

The software has two different sources of the same data or information, but it uses the source that has less support for verification, is less trusted, or is less resistant to attack.

Details at: https://cwe.mitre.org/data/definitions/348.html

- Describes pattern, provides examples, provides list of related CVEs

É mais abstrato que o CVE. Contem os padrões de uma vulnerabilidade.

CWE-348: Use of Less Trusted Source

```php
$requestingIP = '0.0.0.0';
if (array_key_exists('HTTP_X_FORWARDED_FOR', $_SERVER)) {
        $requestingIP = $_SERVER['HTTP_X_FORWARDED_FOR'];
else{
        $requestingIP = $_SERVER['REMOTE_ADDR'];
}

if(in_array($requestingIP,$ipAllowlist)){
        generatePage();
        return;
}
else{
        echo "You are not authorized to view this page";
        return;
}
```

Set by Web Server or Client

Set by Web Server

Vulnerability Tracking by vendors

During the development cycle, vulnerabilities are handled as bugs

- May have a dedicated security team or not

When software is available, vulnerabilities are also tracked globally

- For every system and software publicly available

Public tracking helps…

- focusing the discussion around the same issue
  - Ex: a library that is used in multiple applications, distributions
- defenders to easily test their systems, enhancing the security
- attackers to easily know what vulnerability can be used

Vulnerability Tracking

Vulnerabilities are privately tracked

- Constitute an arsenal for future attacks against targets
- Exploits are weapons

Knowledge about vulnerabilities and exploits is publicly traded

- From 0 to 2-3M€ (more?) through direct markets, or acquisition programs
- Up to 2.5M€ for bug hunting programs or direct acquisition (Google, Zerodium)
  - 2.5M€: 1 click Android exploit
  - 2M€: 1 click iPhone exploit
  - 1.5M€: WhatsApp or iMessage exploit
  - ~2K for a XSS at HackerOne (although there are records of $1M payouts)

…and privately traded at unknown prices

- Private Companies, Organized Crime, APTs

Vulnerability Tracking

Most well-known trackers systems: CVE and NVD

- CVE: Common Vulnerabilities and Exposures, managed by MITRE
- NVD: National Vulnerability Database, managed by NIST
  - Fed by CVE@MITRE but provides enhanced information

Others

- CERT Vulnerability Notes Database (VNDB)
  - Maintained by CERTs, may provide additional information regarding a CVE
- VulnDB
  - Focus on APIs and providing information to companies
- DISA IAVA and STIGS
  - Information Assurance Vulnerability Alerts: includes MIL and GOV systems
  - Security Technical Implementation Guides
- Industry Sharing and Analysis Centers (ISAC)
  - Industry driven, thematic (AUTO, FINANTIAL, IT, etc… groups)

CVE: Common Vulnerabilities and Exposures

Dictionary of publicly known information security vulnerabilities and exposures

- For vulnerability management
- For patch management
- For vulnerability alerting
- For intrusion detection

vulnerabilitie tracking by version, OS, tool, ... less abstract than CWE

Uses common identifiers for the same CVE's

- Enable data exchange between security products
- Provide a baseline index point for evaluating coverage of tools and services.

Details about a vulnerability can be kept private

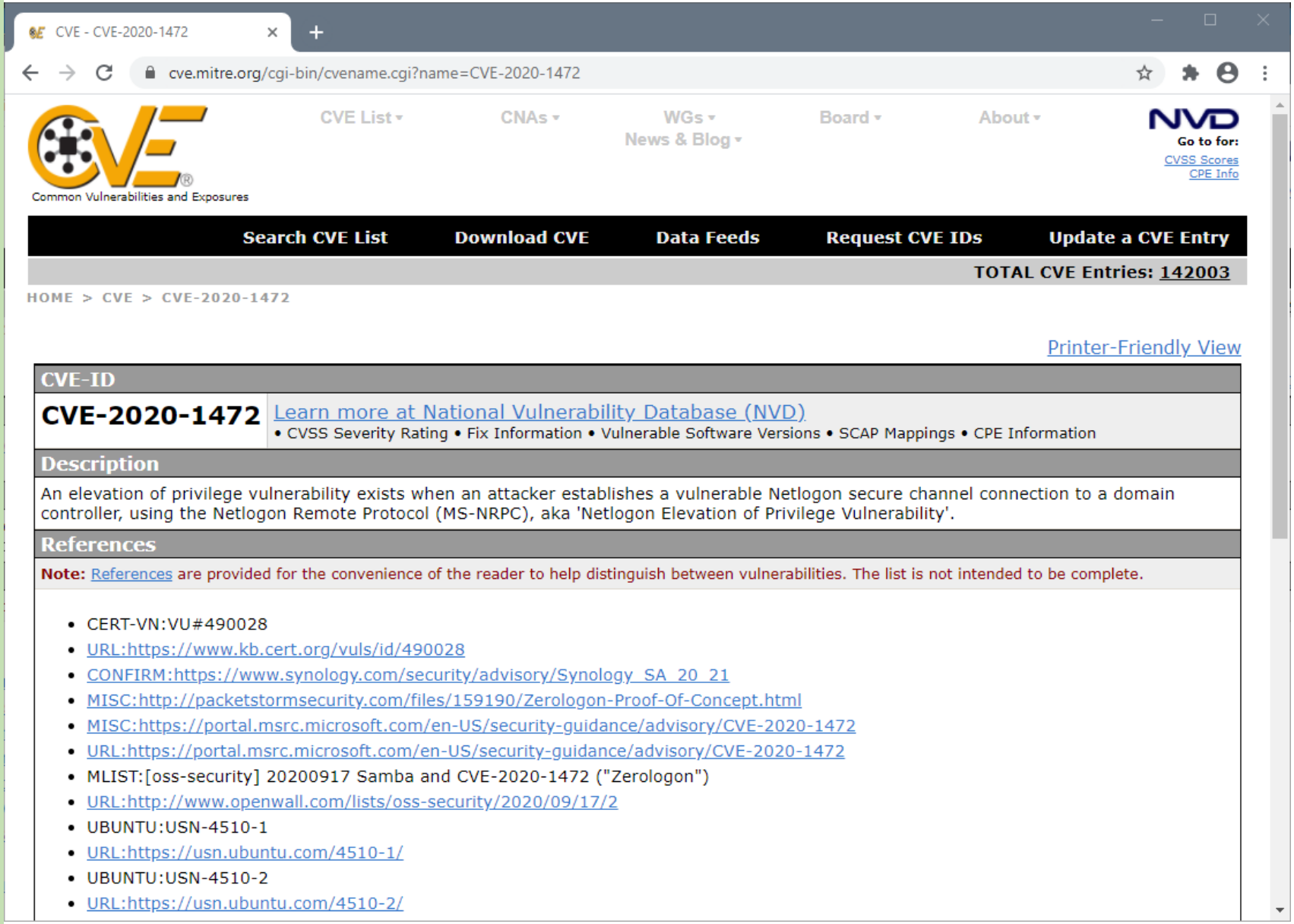
- Part of responsible disclosure: Until owner provides a fix

CVE-2020-1472

@MITRE

Basic information about the CVE

References to other trackers (provided for convenience)



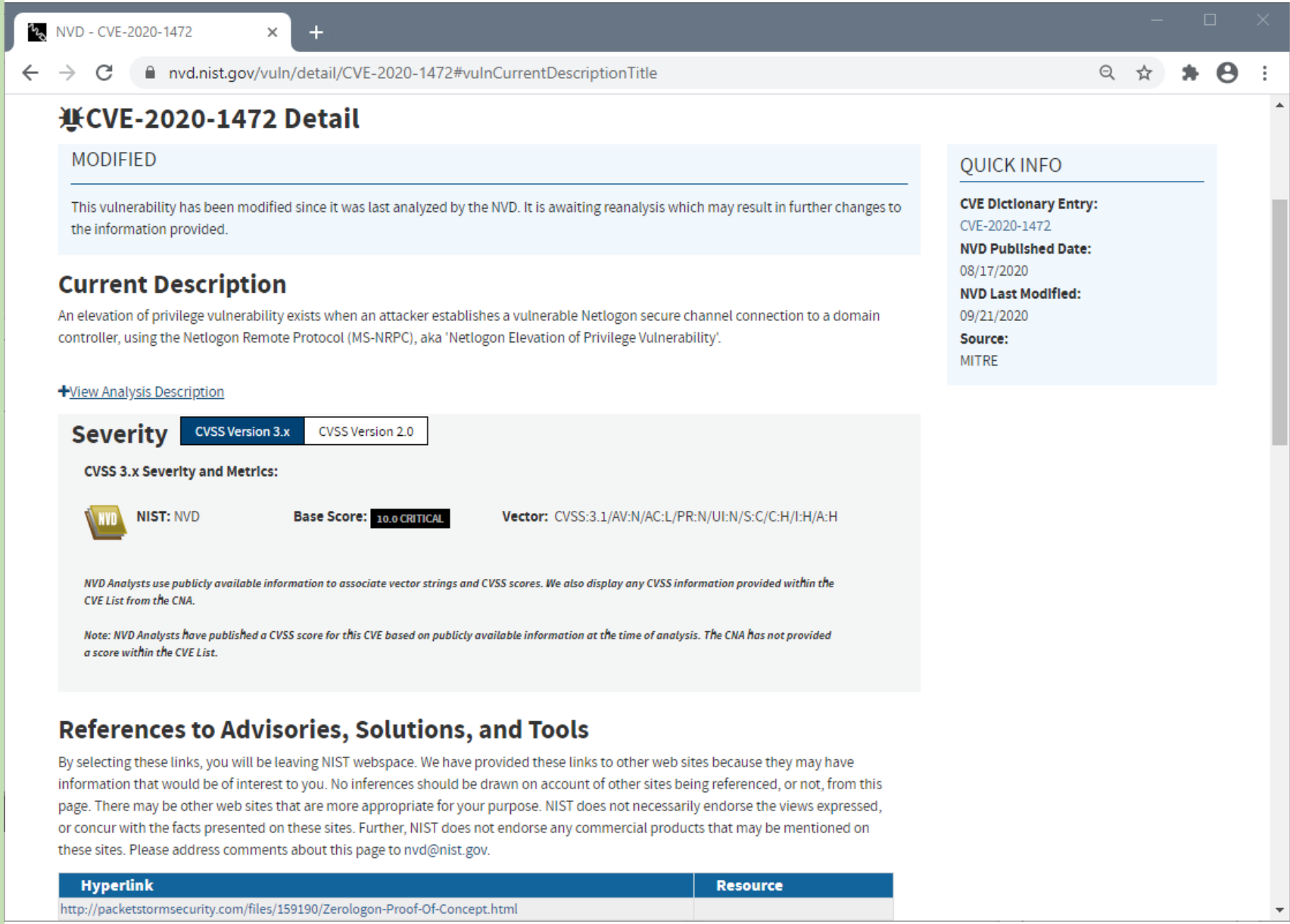
CVE-2020-1472

@NVD

Basic information about the CVE and a small analysis of it

The CVE Severity Score

Links to advisories, solutions



CVE-2020-1472

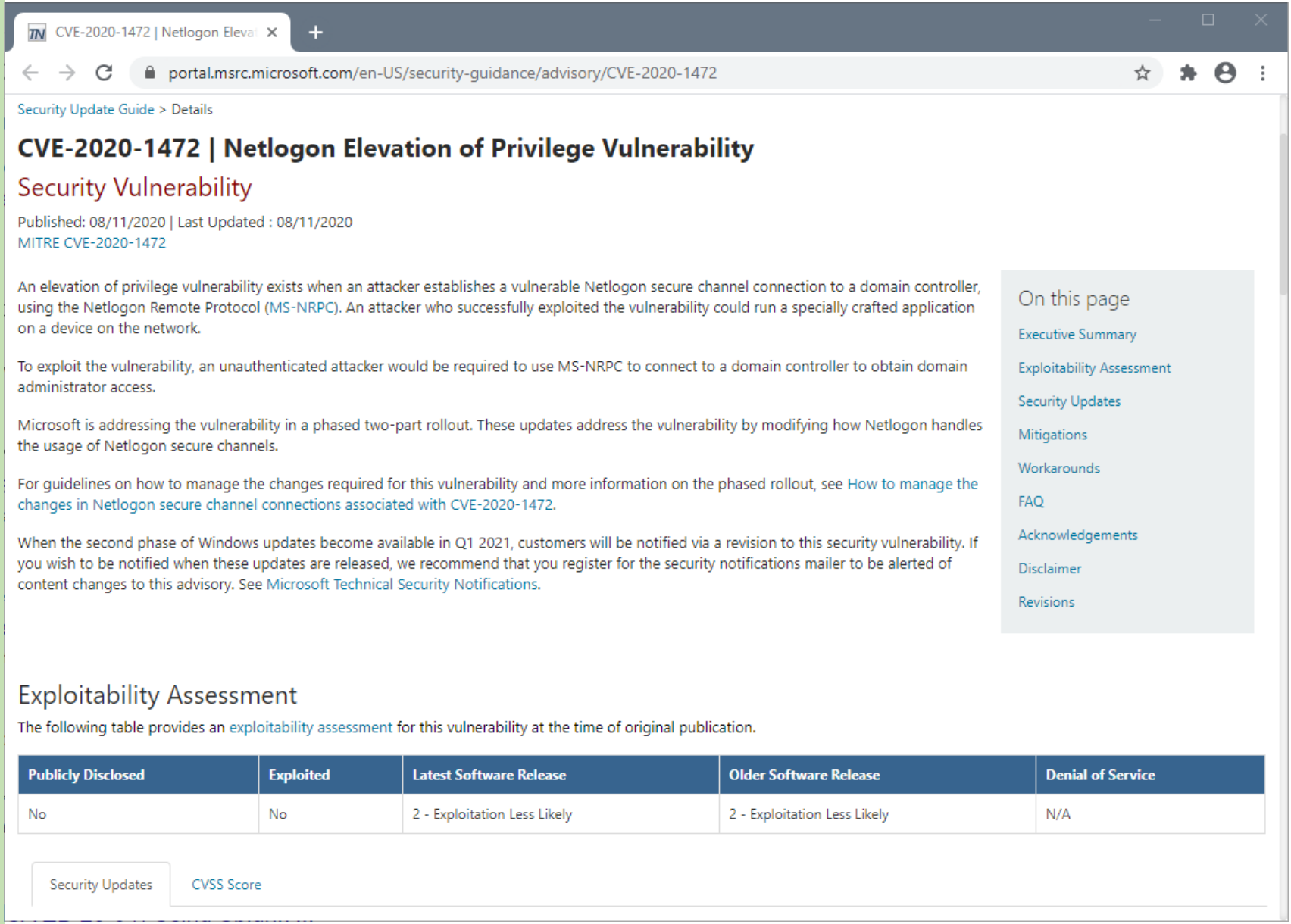
@Product Owner

More detail, why it happens, and how it can be mitigated

Information about patches/updates available to help IT staff and users

Information about it's exploitability.

Format is vendor dependent. Each vendor defines what/how to show information

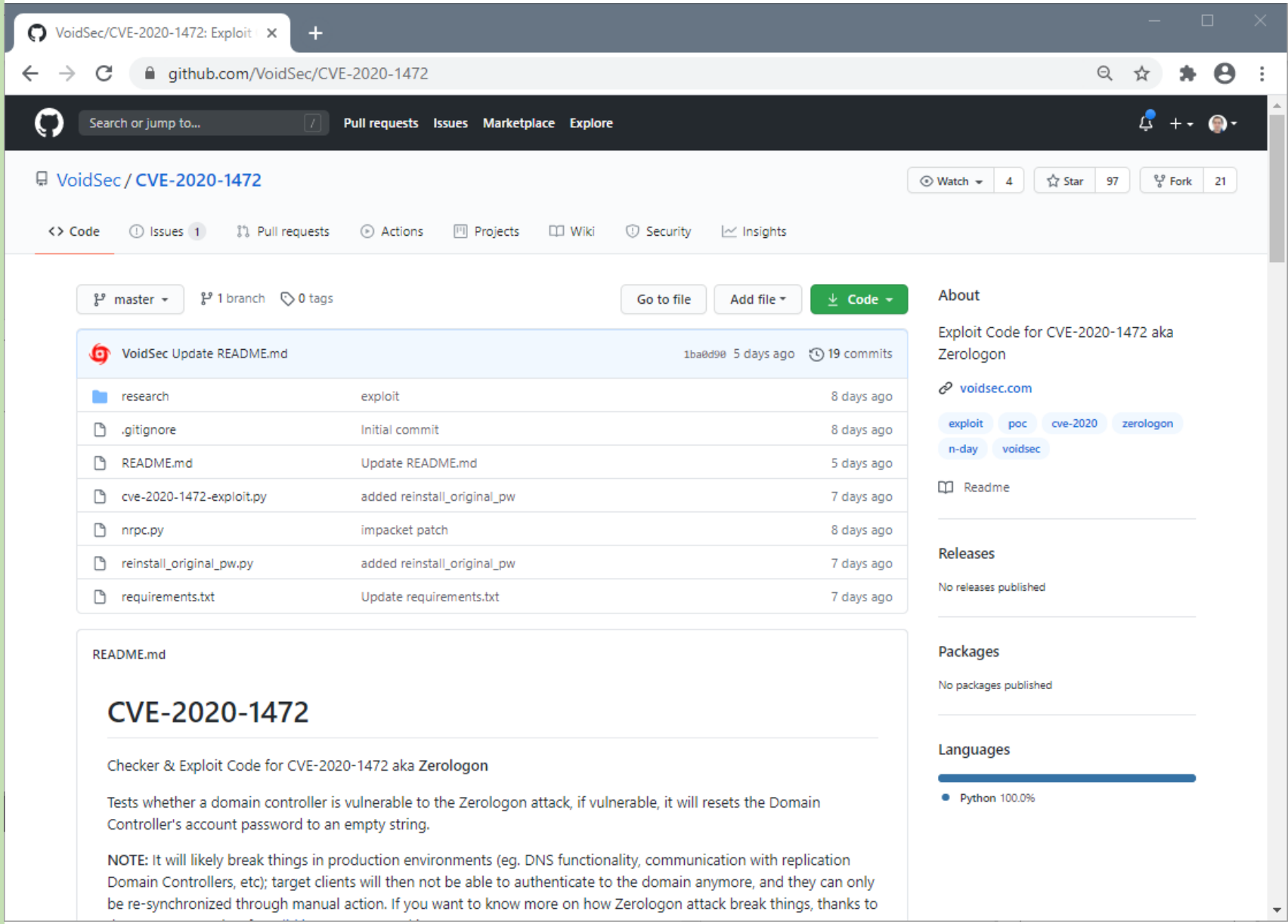


CVE-2020-1472

@Other places

Independent researchers may publish validation tools or exploits

Very dynamic community with public and private facets

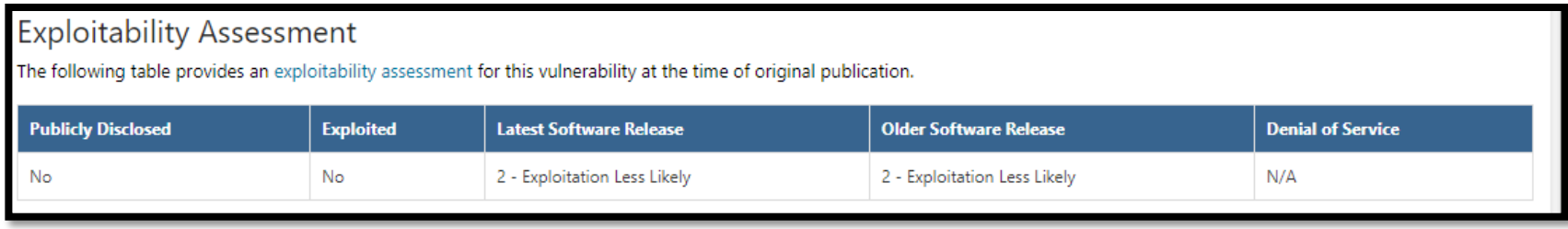


Vulnerability tracking

Not an easy task

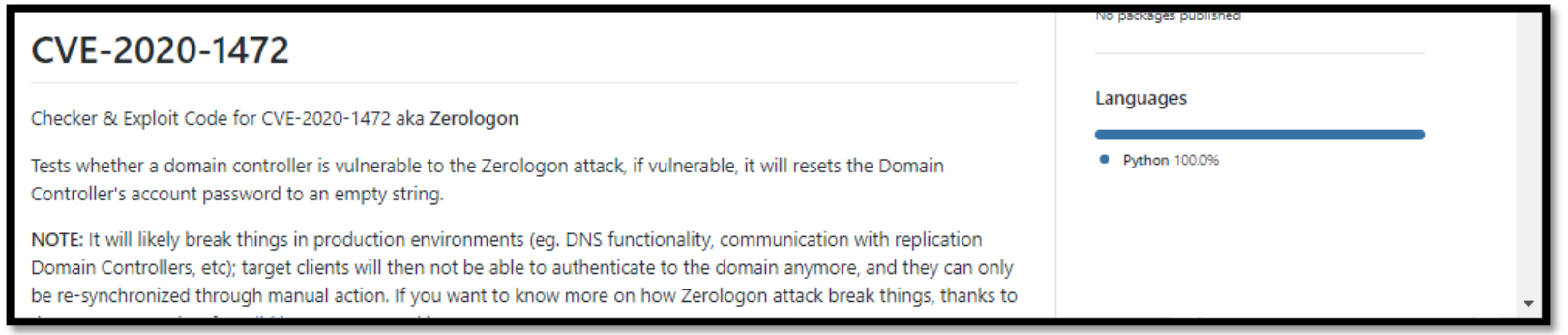
- Exploits are not always known
- Impact and Value may be underestimated

Old feeds may create a false sense of security

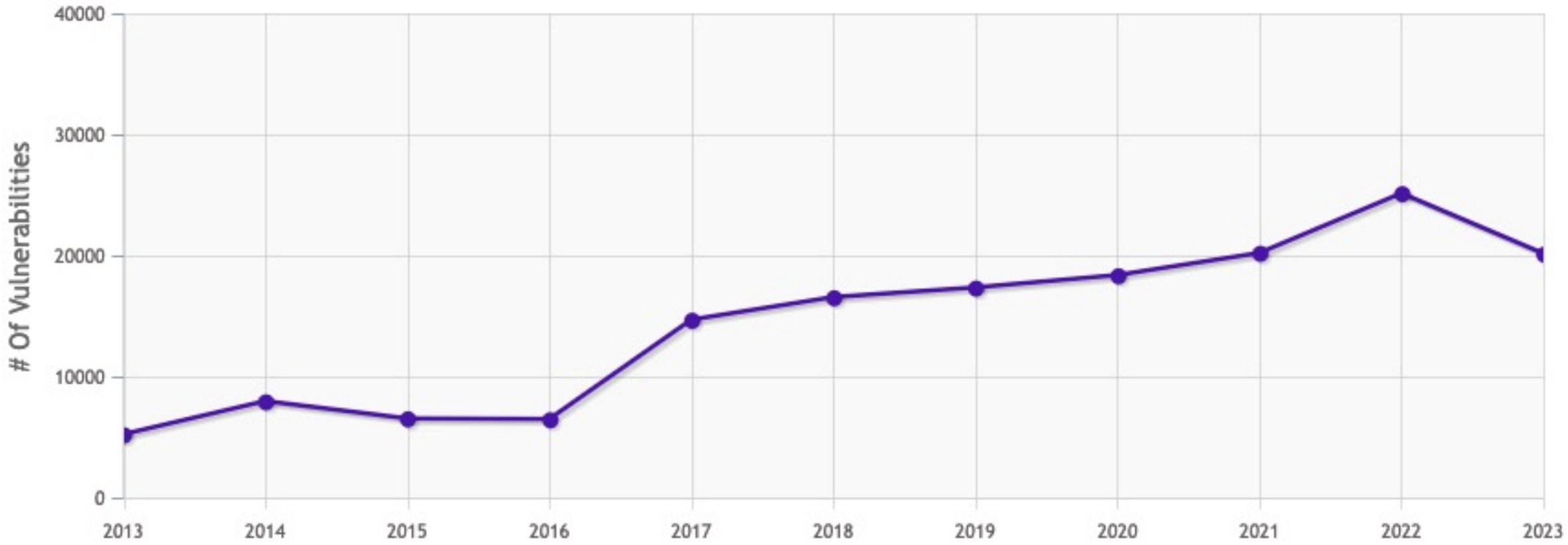


A highly dynamic community is great…

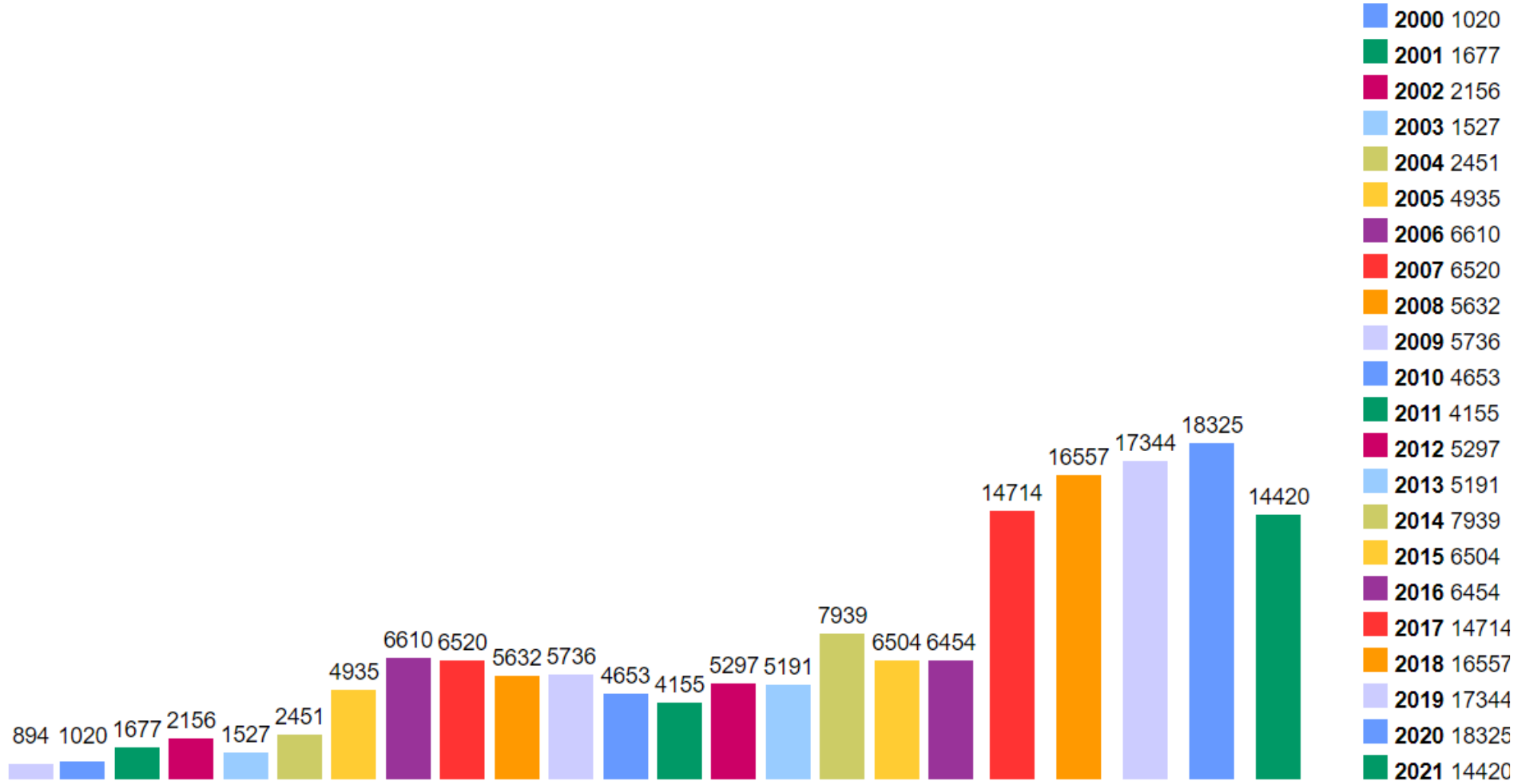
- To defenders as they can test and implement defenses
- To attackers as they can incorporate exploits



CVE per year – cvedetails.com (as of Sep 2023)



CVE per year – cvedetails.com (as of Sep 2021)



CVSS – Common Vulnerability Scoring System

Provides a quick way to determine the severity of a vulnerability (0-10 score)

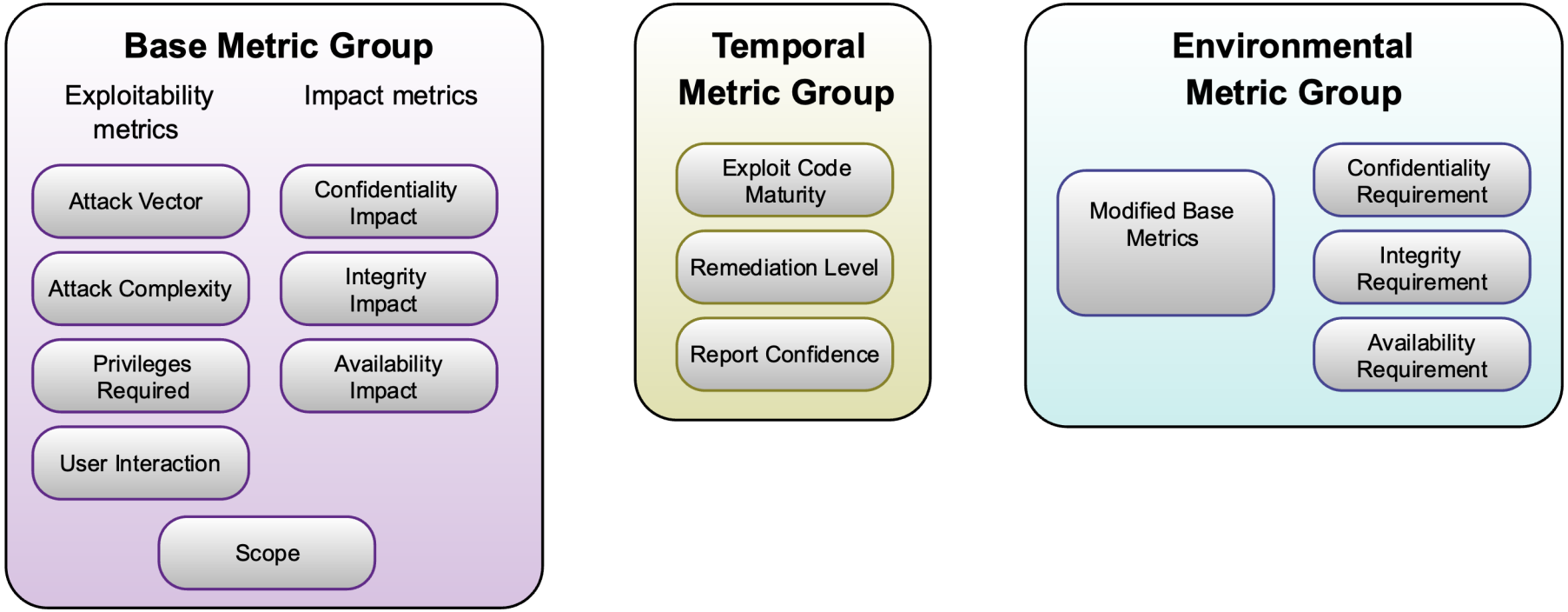
- Helps defenders prioritizing the deployment of mitigations
- Helps attackers selecting the most convenient vulnerability to explore
- Tends to be pessimistic (higher values)

Example: CVSS 3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:L/I:L/A:N

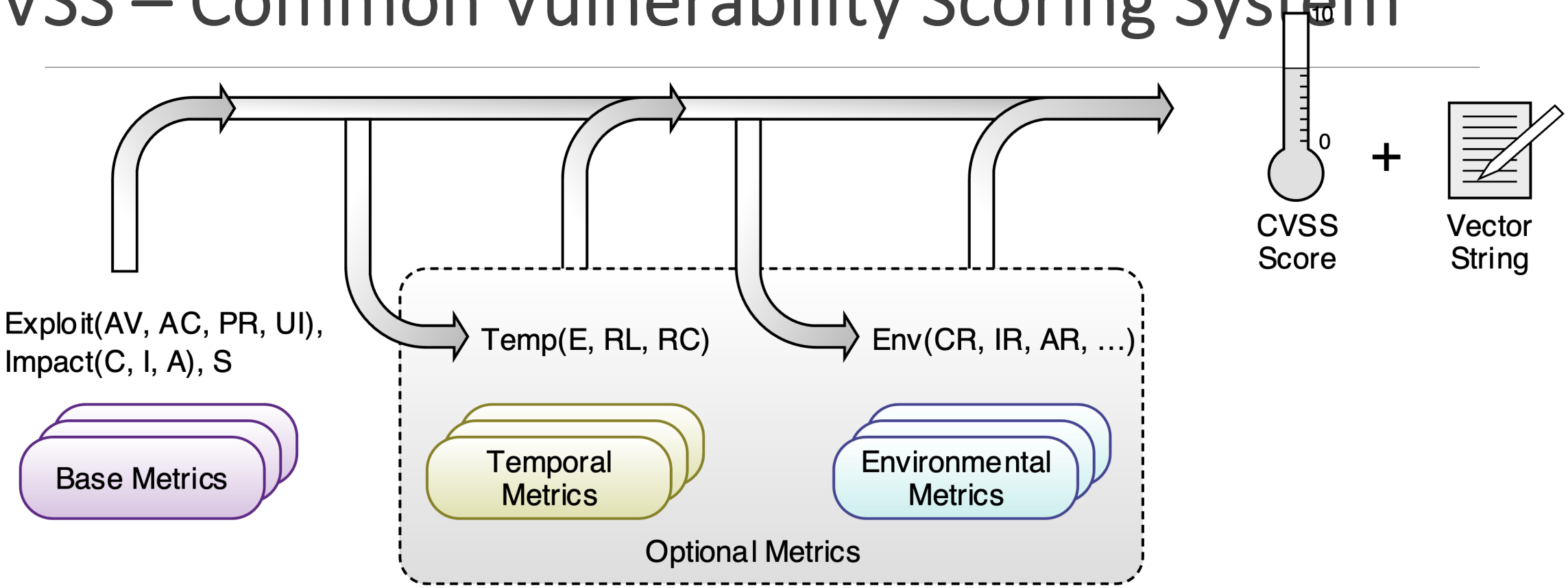
- Final Score: 3.1 (LOW)
- Attack Vector: Network
- Attack Complexity: Low
- Privileges Required: High
- User Interaction: None
- Scope: Unchanged
- Confidentiality: Low
- Integrity: Low
- Exploit Availability: None

CVSS helps decide which vulnerabilities to prioritize.

CVSS – Common Vulnerability Scoring System



CVSS – Common Vulnerability Scoring System



Equations available at: https://www.first.org/cvss/specification-document

Calculator available at: https://www.first.org/cvss/calculator/3.1

Example: Base Metrics

The Base Score formula depends on sub-formulas for **Impact Sub-Score** (ISS), **Impact**, and **Exploitability**

ISS =	**1 - [(1 - Confidentiality) × (1 - Integrity) × (1 - Availability)]**
Impact =	
If Scope is Unchanged	6.42 × ISS
If Scope is Changed	$7.52 \times (ISS - 0.029) - 3.25 \times (ISS - 0.02)^{15}$
Exploitability =	8.22 × AttackVector × AttackComplexity × PrivilegesRequired × UserInteraction
BaseScore =	
If Impact \<= 0	0, *else*
If Scope is Unchanged	Roundup (Minimum [(Impact + Exploitability), 10])
If Scope is Changed	Roundup (Minimum [1.08 × (Impact + Exploitability), 10])

Vulnerability Disclosure

How should a research proceed when a vulnerability is found?

If the engagement is private: deliver to contracting entity

- May negotiate the public release the information…

What about other cases?

Vulnerability Disclosure: None

Researcher doesn't notify vendor about vulnerability

- Doesn't care
- Uses it as part of an arsenal or trades the information

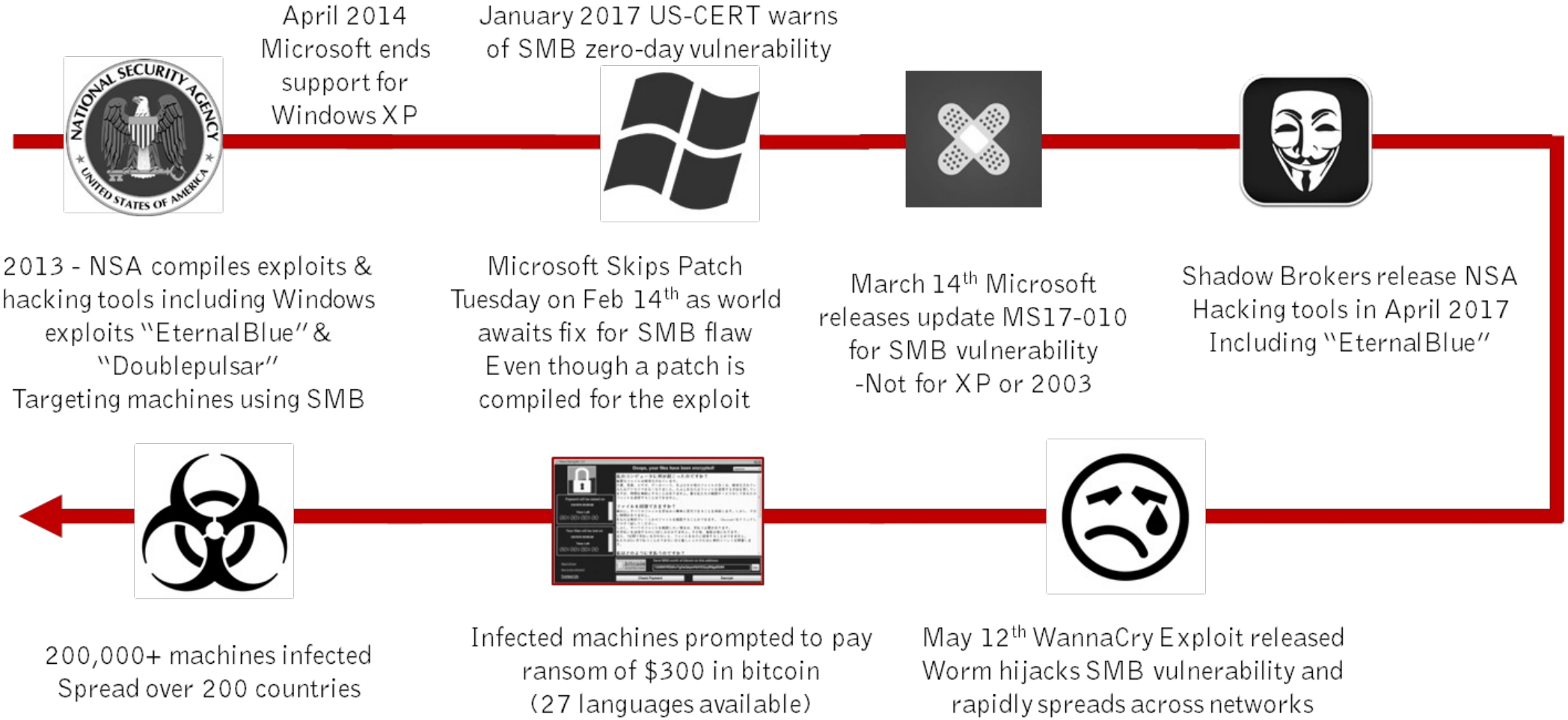
Leads to 0-day vulnerabilities

- Vulnerability is not known to the public and there is no direct remediation
- Some other third parties may also know about the vulnerability and exploit it

If impact is high, it creates major disruption when publicly known

- Quick adoption in malware and dissemination
  - Remember: Systems take at least one month to be patched

CVE-2017-0144 EternalBlue



Source undetermined

Vulnerability Disclosure: Coordinated

1. **Researcher informs vendor about vulnerability and impact**
   - Usually through a form of report with estimation of impact and/or demonstration
2. **Vendor implements and distributes a correction**
   - But not always!
3. **Vulnerability is mostly fixed in supported systems**

→ most common

Optional: CVE entry is requested: https://cveform.mitre.org/

Optional: A website with a fancy name is created for public awareness

CVE-2020-15802 – Sep 9 2020

https://hexhive.epfl.ch/BLURtooth/

Researcher:

- "We discovered the vulnerability in March 2020 and responsibly disclosed our findings along with suggested countermeasures to the Bluetooth SIG in May 2020. We kept our findings private and the Bluetooth SIG publicly disclosed them, without informing us, on the 10th of September of 2020. Our work is assigned CVE-2020-15802."

Bluetooth SIG:

- At the time of writing, there are no deployed patches to address the BLUR attacks on actual devices. The Bluetooth SIG suggested that version 5.1 of the standard will contain guidelines to mitigate the BLUR attacks (e.g., disable key overwrites in certain circumstances as proposed in our countermeasures), but such guidelines are not (yet) public and we cannot comment on them. The Bluetooth SIG provides a public statement about BLURtooth and the BLUR attacks.

Vulnerability Disclosure: Full

Researcher discloses the vulnerability without warning

- As a CVE
- In a public mailing list
- As a blog entry, webpage or news item
- As an exploit

Vendor is pressured to issue a fix as soon as possible

- But not always
  - It doesn't!
  - It considers the product not supported
  - It under reports the issue

Some mayhem may occur until a fix is applied

- Remember all those phones/TVs/etc… without frequent updates

Exercises

This task proposes that a group of 2 students analyze one CVE from the following list and identify:

- what it affected
- what was the vulnerability
- how/when it was discovered
- when was it fixed
- how it was exploited by attackers
- what was the impact of its exploitation
- what was the timeline of major events

CVE-2017-18017 - xt_TCPMSS

CVE-2017-17510 - DLINK Devices

CVE-2017-5754 - Meltdown

CVE-2017-5753 - Spectre

CVE-2017-13077 - KRACK

CVE-2017-0144 - Eternalblue

CVE-2016-10229 - UDP

CVE-2015-1538 - Stagefright

CVE-2014-6271 - Shellshock

CVE-2014-3566 - Poodle

CVE-2014-0160 - Heartbleed

CVE-2013-3183 - Ping6 of Death

CVE-2009-3677 - MSCHAP

CVE-2008-1447 - Kaminsky DNS

CVE-2023-27524 - Superset

CVE-2023-1748 - Nexx

CVE-2023-1424 - Mitsubishi

CVE-2022-36958 - Print Spooler

CVE-2021-44228 - Log4j

CVE-2021-26855 - Proxylogon

CVE-2020-9478 - Rubrik CDM

CVE-2020-15802 - BLURtooth

CVE-2020-1472 - Zerologon

CVE-2020-0796 - SMBGhost

CVE-2019-17510 - DLINK Devices

CVE-2019-15846 - Exim Backslash

CVE-2019-15926 - Linux Out of Bounds

CVE-2017-15846 - Exim Backslash

CVE-2017-0144 - Eternalblue

Exercises

1. **Visit https://threatpost.com or https://www.securityweek.com**

2. **For any article:**
   - Summarize (or speculate) what went wrong
   - Was it an application problem or an external element?
   - How could it have been avoided? Left right
   - How would your team have reacted?

3. **Discuss the point with the class**

Both exercises will not be graded

- Only to share some thoughts