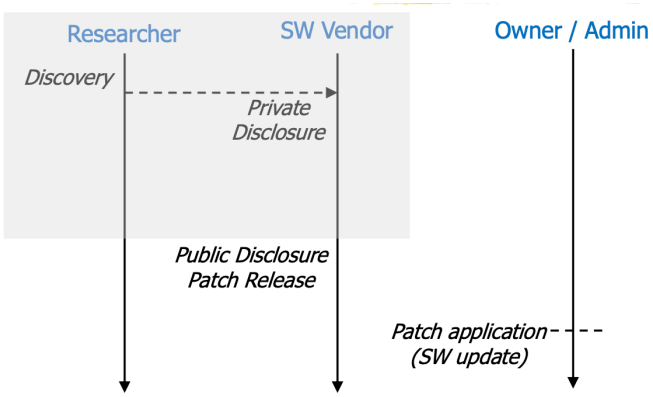
Vulnerability lifecycle

- Ideal case:



-Slide sul Microsoft "patch Tuesday"
-Vulnerability Knowledge: idealmente si vuole che nessuno a parte il ricercatore e il SW vendor sappiano nulla della vulnerabilità, finché non viene resa nota pubblicamente insieme al patch, il fatto che nessuno dovrebbe saperne nulla include ovviamente potenziali attaccanti
-Esempio Log4j
-Takeaway (ideal case): bisogna applicare i patches e farlo velocemente, questo è più facile a dirsi che a farsi
-Questo perché in generale, una persona non esperta che da consigli sulla sicurezza online non include mai l'installazione degli aggiornamenti software, mentre, dal punto di vista di un esperto in merito, sarebbe la prima pratica da mettere in atto in qualsiasi contesto

- In reality, in many cases one or more of these events do not occur:
  - The researcher may not notify SW vendor, SW vendor may not develop a patch, it is possible a public disclosure without any patch available, or owner/admin may not apply the patch

Patch development

- Form SW vendors point of view: the patch development is expensive, may occasionally be very difficult and, in most cases, there is no contact obligations for patch development
  - The software may have ended the security support for the vulnerable software → EOL (end of life)
  - If a vuln is an EOL SW, all the copies of that software are and remain vulnerable forever
- EOL examples
  - Struts1: Apache Struts 1 End of life Press Release to inform that Struts 1.x web framework has reached its end of life and is no longer officially supported in 2013, but the last release is the version 1.3.10 - December 2008. In the meantime the Structs community has focused on pushing the Struts 2 framework forward, with as many as 23 releases as of this writing
  - Windows EOL slide con le versioni di windows e corrispettive EOL
- EOL consequences: Wannacry → 300000 infections, infected systems are EOL or had not applied a 2 months old patch released by microsoft

NB never ever depend on EOL systems →plan ahead for migration

- If you have to depend on EOL systems → limit their use and network exposure as much as possible, monitor any anomaly and insist for their discussion with top management

Responsible disclosure

- Pratical and very important issue: if you discover a vuln and report if to the SW vendor, the vendor can downplay the relevance and does not act
- What to do? All the instances are vulnerable, but even their owner do not know → an attacker might discover and exploit that vuln
- Vulnerability discovery: key approach
  1. Make vuln info public → defenders are alerted, public pressure on vendors may be the thing that can work (ovviamente prima li avverto in privato, poi rendo tutto pubblico se il vendor non fa nulla), in this case attackers are alerted as well
  2. The researcher can also make a proof of concept exploit to make stronger public pressure on vendors
  - This is potentially dangerous for community
  3. Give a reasonable deadline to the SW vendor for acting before going public → responsible disclosure
- An example si Google project zero that discover bugs, gives a deadline of 90 days + 14 days of grace period before going public
  4. Keep vuln secret and forget about It →do not lose time and do not fear legal actions

Zero days

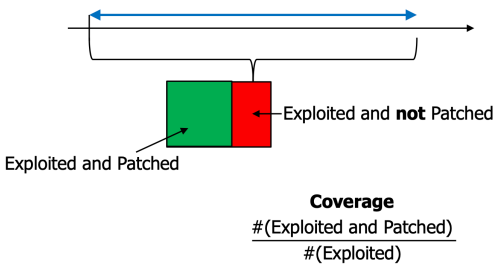
- Another thing that can happen, when you discover a vuln, you can keep the vuln secret and sell the exploit
  - Why report the vuln info to the SW vendor, if you can sell vuln info to organizations interested in exploiting it
  - This can be done in legal or illegal markets
- Zero days vulnerability: vulns that are exploited while unknown to the manufacturer (and everyone else)
- These zero days vulns can be bought in legal markets, discovered by attackers (criminals, intelligence services), bought in illegal markets
- They are a real problem, the magnitude is unclear
- Example of legal market: Zerodium buys zero days exploits from researchers and sells them to selected organizations
- Some 0-days stats on legal market:
  - 2004-2016 data from selected 0-days vendors
    - 207 zero days with exploits, 50% publicly unknown, average life expectancy 6.9 years
    - Only 25% live less than 1.51 years and only 25% live more than 9.5 years
  - 2014-today maintained by google project zero
    - Public spreadsheet tracks cases of zero day exploits that were detected in real attacks, vulns not known to the public or the vendor at the time of detection
    - April 5 2022: 211 august 9 2023: 284
- Attackers attempt to avoid usage of 0 days whenever possible
  - 0 days have high effectiveness, high economic value, and if discovered by defenders → significant loss (they are not zero day anymore)
  - Minimizing the risk of discovery is crucial
- Population of experts incentivized in discovery of exploitable vuln, keep them hidden to vendors, diffuse knowledge in restricted circles
- Users/organizations have more and more vulns exploitable for a long time, they do not even know about, but several restricted circles do
- If you are a national intelligence service, you discover or are proposed to acquire an high impact vuln used by many organization or by critical infrastructure → you can use it for attacking enemy organizations, but you may leave your organizations vulnerable

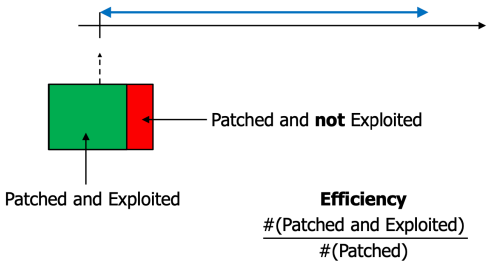
Patch management

- We told that in some cases the owner/admin of the software does not apply the patch, this is because the patch management is a very difficult problem with many facets
- Facts about patch application:
  - Applying a patch may be complex and time consuming
  - Apply a patch usually requires device downtime
  - Upgrading a system X to version N+1 may create compatibility problems with systems that interact with X
  - System owner/admin often is not even aware of the specific vulnerability, which system are affected and who is in charge of them
  - There are just too many vulns that might need patching
  - It is a fundamental problem
- Every week in NDV are published a lot of new CVEs (così, per curiosità)
- Basic fact: very few CVEs are actually exploited (about 5% of all the CVEs), the administrators have to focus on those, the problem is that predicting which CVEs will be indeed exploited is very difficult

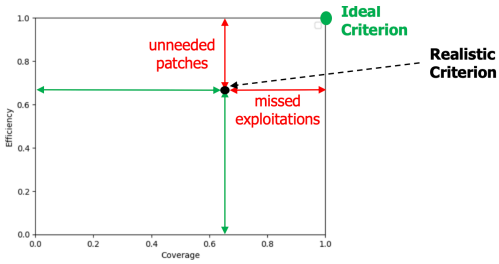
Exploit prediction: problem definition

- We have to define a criterion for choosing which vulnerabilities to patch
- We observe which vulnerabilities have been actually exploited worldwide, it is an approximation made by collecting many intelligence feeds
- Covarage Efficiency





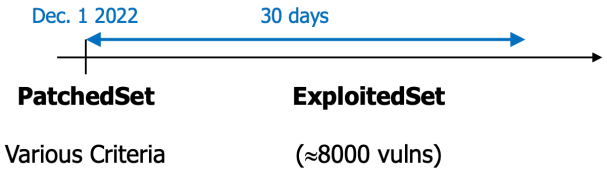
Vulnerabilities: management



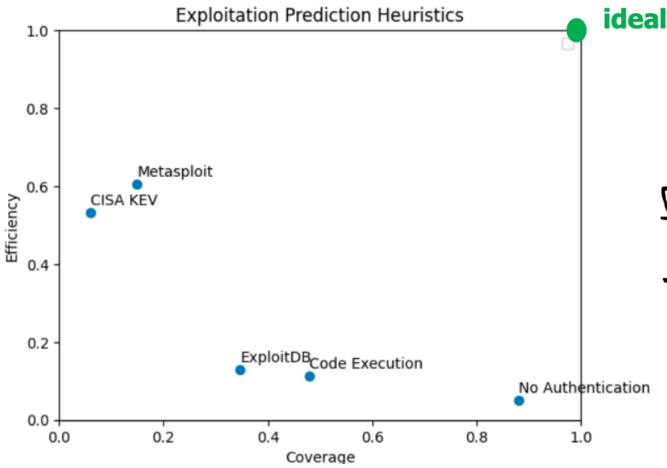
- High efficiency and low coverage: you have patched only vulns that matter but have missed many vulns
- Low efficiency and high coverage: you have wasted a lot of patching effort but you have covered nearly all vulns
- Patching effort: efficiency and coverage are relative indexes, one may have different criteria with identical efficiency and coverage but widely different patching effort (#patched)
- Summary of the problem definition:
  - We have to choose a criterion for choosing which vulnerabilities to patch, the assessment indexes are how good in defense (coverage), how efficient and how costly
  - Many favor are not assessed
  - Given a certain vuln, we have to understand how many systems do I have with that vuln, how costly is an incident based on that vuln and how likely is to be attacked with that vuln

---

Exploit prediction: example criteria

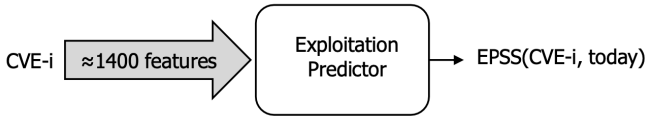


- scenario →
- Selection based on CVSS:
  - We choose to patch all the CVEs with CVSS >= 9.1 (circa 15% of all the vulns)
  - Patching effort of about 28000 vulns
  - CVSS is not a good predictor of exploitation, has an high rate of unneeded patches and missed exploitations (very low efficiency and low coverage)
- Selection based on CISA-KEV:
  - CISA: cybersecurity & infrastructure security agency
  - KEV: known exploited vulnerability catalog
  - The CVEs that are in CISA-KEV are about 0.5% of all the vulns, the patching effort is of 900 vulns
  - CISA-KEV is not a good predictor of exploitation, it has not much unneeded patches (ovvio perch sto patchando solo quello che viene effettivamente exploited) but it has a very high rate of missed exploitations (about 50% efficiency and very low coverage)
- There are other selection criteria →
- Nessuno di questi criteri è particolarmente degno di nota



EPSS: Exploit prediction scoring system

- The EPSS (CVE-i, d) is the probability estimate that the CVE-i will be exploited in [d, d+30]
- It changes every day, the probability definition are (non molto chiaro cosa voglia dire)
  - The number of CVE-i exploitation attempts worldwide
  - The number of all CVE exploitation attempts worldwide
- Pseudocodice:
  - Repeat everyday:
    - For each CVE-i
      - Compute features of CVE-i
      - Estimate its probability of exploitation in the next 30 days



- The exploitation predictor is a regressor (XGBoost): it is a data driven model trained on 1 year of observed exploitation activity (march 2023 3rd model)
- The vuln is represented in an array with 1400 elements (numerical features and categorical features (one-hot representation)

Vulnerabilities: management

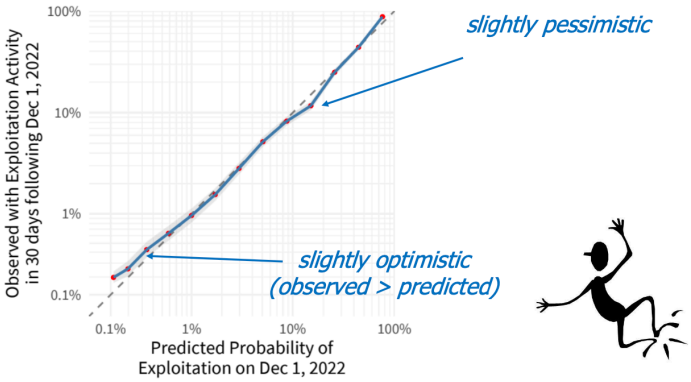
Description	Sources
Exploitation activity in the wild (labels)	Fortinet, AlienVault, Shadowserver, GreyNoise
Publicly available exploit code	Exploit-DB, GitHub, MetaSploit
CVE mentioned on list or website	CISA KEV, Google Project Zero, Trend Micro ZDI
Social media	Mentions/discussion on Twitter
Offensive security tools and scanners	Intrigue, sn1per, jaeles, nuclei
References with labels	MITRE CVE List, NVD
Keyword description of vulnerability	Text description in MITRE CVE List
CVSS metrics	National Vulnerability Database (NVD)
CWE	National Vulnerability Database (NVD)
Vendor labels	National Vulnerability Database (NVD)
Age of the vulnerability	Days since CVE published in MITRE CVE list

-Praticamente la rappresentazione della vulnerabilità raccoglie più o meno tutte le sue proprietà intrinseche + un riassunto di quello che la gente dice/pensa della vulnerabilità

-Il tutto viene aggiornato ogni giorno

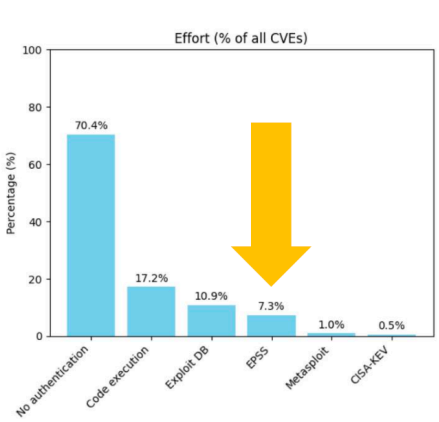
-Esempio: se una CVE ha un EPSS=0.01, significa che dagli exploitation attempts osservati, 0.01 (1%) tentativi su tutti riguardano questa CVE

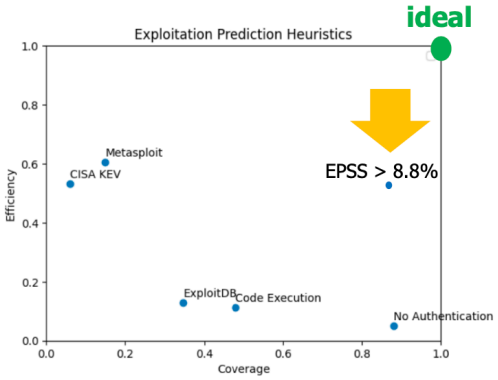
- $P_{observed}(CVE_i) \approx P_{predicted}(CVE_i) \rightarrow$
- EPSS evolution examples
  - In can occur a significant growth after 1 day, again on the next day and heavily exploited for more than 6 months
  - In can occur that a vuln is immediately exploited heavily for several weeks
  - The temporal evolution of the EPSS (and the exploitation of the vuln) may often be very irregular and hard to predict (even in the short term)
- EPSS is a predictor, it actually acts retrospectively and it may have delays of several days



EPSS vs CVSS selection criteria

- Same patching effort:
  - The set taken is: CVEs that has EPSS >= 0.022, CVSS >= 9.1 (15% of all the CVEs)
  - EPSS has a much better coverage that CVSS (actually has an excellent coverage), and it is more efficient than CVSS (but is not very efficient)
- Same coverage
  - we take all the CVEs that has CVSS >= 7 and EPSS >= 0.088 (both have 82% of coverage)
  - In this case EPSS is much more efficient than CVSS (but not very efficient) and EPSS requires much less patching effort than CVSS





Vulnerability management in organizations

- Fact of life: there are too many vulns that might need patching → we cannot eliminate all the risks associated with vulnerabilities
- Common and wrong approach:
  - Give up and cross the fingers
  - Patch without any hurry extremely critical vulns that emerge every now and then → many cases of total catastrophe are caused by evident and easy to patch vulns
- Correct approach is aim for systematic and constant reduction of risk, to the extent feasible with the defensive budget available. Establish a structured process for vuln management (structured process → non una cosa fatta solo ogni tanto quando capita, management → gestire, non eliminare)
  - Basic requirement: sufficient human resources allocated to this process

NB no organization patches a significant fraction of its vulns, and no organization applies patches quickly

Vulnerabilities: management

- Typically only about 15% of internal vulns are patched within a month from the available patch
- Vulnerability management:
  - Establish who is in charge of collecting guessing material, how much effort to allocate, how to coordinate effort (planning, scheduling), how to plan and manage downtime (out of scope)
  - Fundamental components: (che a noi interessano)
    - Asset management: which systems and who is in charge
    - Vulnerability prioritization: how to allocate defensive efforts to CVEs

---

Asset management

- Asset management is an accurate description of systems with all the software versions, known vulnerabilities, if they are exposed to the internet and by who are they managed
- A continuous and comprehensive asset visibility is a basic pre condition for any organization to effectively manage cybersecurity risks
  - È una lista in cui c'è scritto tutto si ogni sistema dell'organizzazione, non si può pensare alla sicurezza informatica senza sapere prima che sistemi bisogna proteggere (di base non so neanche che vulnerabilità devo affrontare se non so cosa ho in azienda)
- Asset management means creating and maintaining authoritative and accurate information: day by day operations, efficient decision making when you need them (it is not about making lists or databases that never get used)
- Defensive cornerstone:
  - Asset management is not only for vuln management: how can you defend effectively if you do not even know what are you defending
  - There are many tool families: asset discovery/management, vulnerability scanners (probe devices agains a subset of known vulns), internet exposure monitor, attack surface discovery/management

NB Any defensive tool is almost pointless without an effective asset management

- Maturity test for organization: se chiedi al direttore tecnico di procurarti la appena descritta lista, non dovrebbe metterci più di mezz'ora a tirarla fuori (che ridere, simpatico)
  - Much easier said than done: in October 2022 CISA emana a binding operational directive, a compulsory direction to federal, executive branch, departments and agencies
    - By April 3, 2023 they have to perform automated asset discovery every 7 days, initiate vulnerability enumeration across all discovered assets, including all discovered normadic/roaming devices, every 14 days…
  - National Cyber Security Center (penso inglese) ha reso pubblica una guida per l'asset management

---

Vulnerability prioritization

- It is divided in two:
  - Threat assessment: how likely is that our org will be attacked with a certain CVE
  - Environmental assessment: if the org is attacked with a certain CVE, what is the risk?
- Threat assessment + environmental assessment → defensive action (patching, making injection more difficult, enhancing monitoring/ detection, or if forget about a certain CVE (for a motivated reason))
- We will see two systematic methods fro obtaining numerical scores (for vulnerability prioritization), we will not establish a methodology for combining these scores or for choosing the defensive actions

---

Environmental assessment

- Consiste nel capire se ci sono patch esistenti per una certa vulnerabilità, se è difficile l'injection e nel caso come fare a rilevare/contenere, capire che sistemi sono infettati, and potential IT/business impact
- CVSS FULL score: standardized procedure for obtaining a numerical score (does not take business impact into account, but does consider all the other)
- CVSS
  - 8 categorical features (impact, exploit injection) → base score

Vulnerabilities: management

- 3 categorical features (difficulty of writing exploit) → temporal score
- 11 categorical features (peculiarities of my environment) → environmental score

- NVD (national vulnerabilities database) does not consider the temporal and the environmental scores, it provides only a base score and a CVSS calculator than anyone can use → praticamente tu fai le tue considerazioni sul cosa una vulnerabilità può fare nella tua azienda, vai sul database, selezioni la CVE che ti interessa e calcoli (con il calculator che ti mette a disposizione il sito) il tuo full score "compilando" tutte le categorie per il temporal e l'environmental score, il calcolatore in base a quello che metti dentro ti da il full score
- CVSS versions:
  - Version 4 in November 2023, it is very complex (input assignment and understanding the meaning of the score components)
  - NVD in January 2024: gives the base score for CVSS 2/3 and full score calculators for CVSS 2/3
  - Opinione del prof: se vuoi capire il full score meglio, bisogna leggere le descrizioni di CVSS versione 2, the full score details of CVSS 3/4 are for specialists

---

Threat assessment

- What should be done:
  - We should assess how likely is that our org will be attacked with a certain vulnerability exploitation? Or with a certain CVE?
  - It is hardly feasible in practice (non possono mettermi a capire quanto è probabile essere attaccati per ogni vulnerabilità possibile)
- Instead can be used
  - Threat intelligence: alerts at the level of country/sector (not our org)
  - EPSS (CVE-i): remember that it evolves daily
  - We assume that the global prediction can also predict my events → it is the only practical approach in absence of any specific threat intelligence
- Vulnerability prioritization outline:
  - Every day in should be done:
    1. Download EPSS for all CVEs
    2. Discard those that are irrelevant (no system affected)
    3. Discard those that you consider irrelevant (environmental)
    4. Choose an EPSS threshold based on you budget
    5. Analyze key novelties

Software dependencies

- Vulnerability risk: practical case
  - A vulnerability for a library has been just made public, do I have to worry? Which systems makes use of this library? The asset management has coarser granularity
- Example Log4j, Twig
- Open source projects have an average of 180 package dependencies → quando scarico qualcosa, ci sono talmente tante dipendenze e librerie usate che probabilmente neanche conosco, non so quale vulnerabilità potrei avere
- Fact: huge problem
  - Every software depends on many modules developed by different manufacturers (third-party dependencies)
  - It is extremely difficult to
    - List third party dependencies
    - Define the perimeter of trust
    - Assess the potential reach of a given vulnerability (we don't know which modules depend on a certain vuln)
- SBOM (software bill of materials): has emerged as a key building block in software security and software supply chain risk management
  - Is a nested inventory, a list of ingredients that make up software components
  - It is a work in progress (there are lot of technical difficult problems)