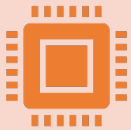
Network Security

AA 2020/2021

Vulnerabilities

Software bugs



A bug is a problem in the execution of the software that leads to unexpected behaviour

- Software crashes
- Wrong entries are displayed/stored in a backend database
- Execution loops infinitely
- ..

Characteristics of a bug

- Replicability
- Logic/configuration/design/implementation
- Fix priority
- If it's documented, it's a feature

An example of a sw bug (pseudocode)

.....

```
        gets(password);
        correct_pwd=lookup(username, database);
        if (correct_pwd!=password)
                printf('Login failed');
                return err;
        else{
                printf('login succeeded');
                exec(context);
        }
        return x;
}
```

An example of a sw bug (pseudocode)

This is a vulnerability!

```
……….

        gets(password);
        correct_pwd=lookup(username, database);
        if (correct_pwd=password)
                printf('login succeeded');
                exec(context)
        else{
                printf('Login failed');
                return err;
        }
        return x;
}
```

Vulnerabilities

- *A flaw or weakness in system security procedures, design, implementation, or internal controls that could be exercised (accidentally triggered or intentionally exploited) and result in a security breach or a violation of the system's security policy*

Definition from NIST SP 800-30

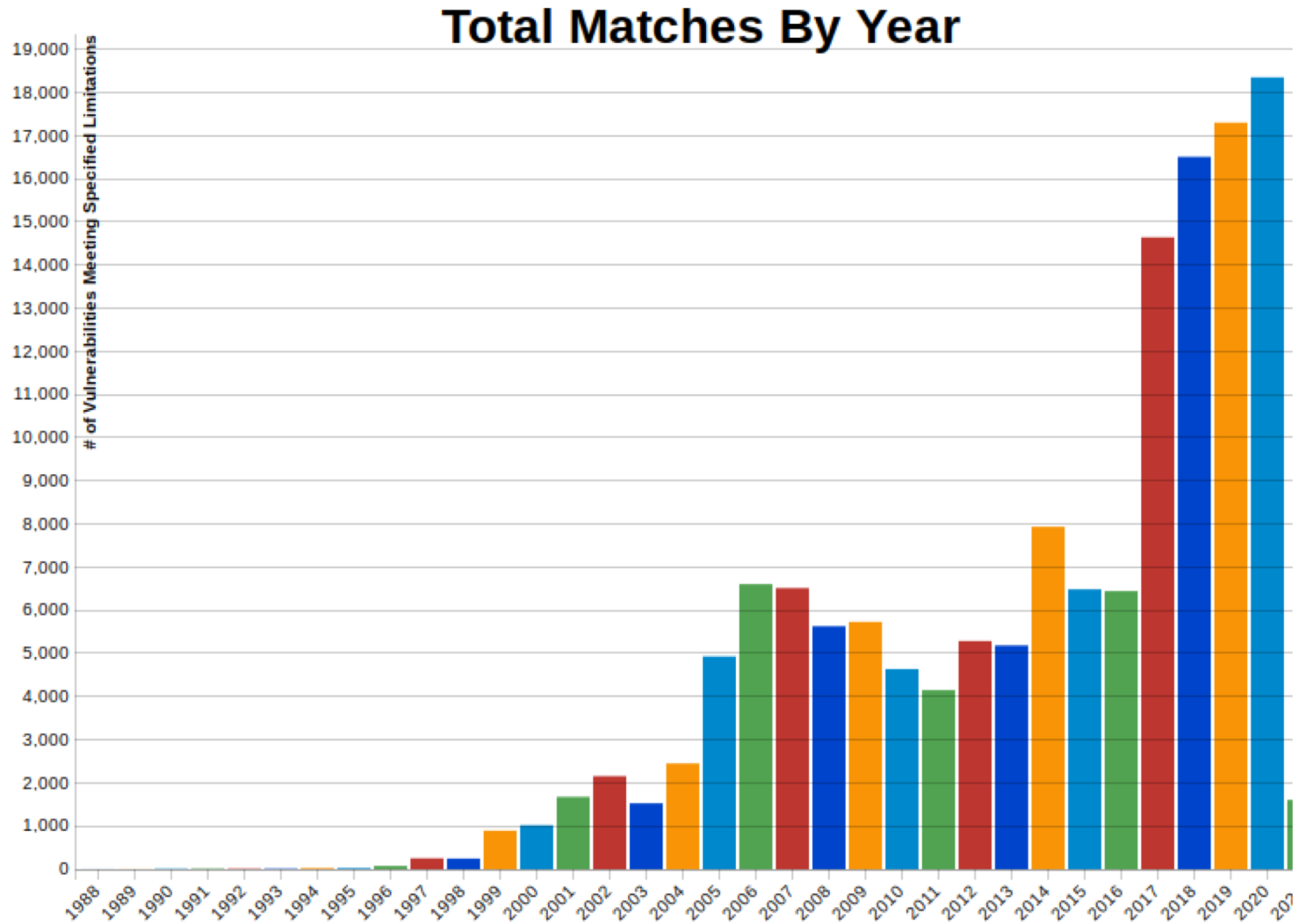
Types of vulnerabilities

- Vulnerabilities can be found at any level in an information system
  - Configuration vulnerabilities
  - Infrastructure vulnerabilities
  - Software vulnerabilities
- Configuration vulnerabilities
  - Software or system configuration does not correctly implement security policy
    - e.g. accept SSH root connections from any IP
- Infrastructure vulnerabilities
  - Design or implementation problems that directly or indirectly affect the security of a system
    - e.g. a sensitive database in a network's DMZ
- Software vulnerabilities
  - Design or implementation of a software module can be exploited to bypass security policy
    - e.g. authorisation mechanism can be bypassed

Vulnerabilities

- Thousands of vulnerabilities are discovered each year
  - Some are publicly disclosed
  - Some are not
- MITRE → non-profit organisation (Massachusetts, U.S.A.)
  - Supports, among others, activities from
    - Department of Homeland Security (DHS)
    - Department of Defense (DoD)
    - National Institute for Standards and Technology (NIST)
  - Maintaines standard for vulnerability identification
    - Common Vulnerabilities and Exposures (CVE)

Known Software Vulnerabilities

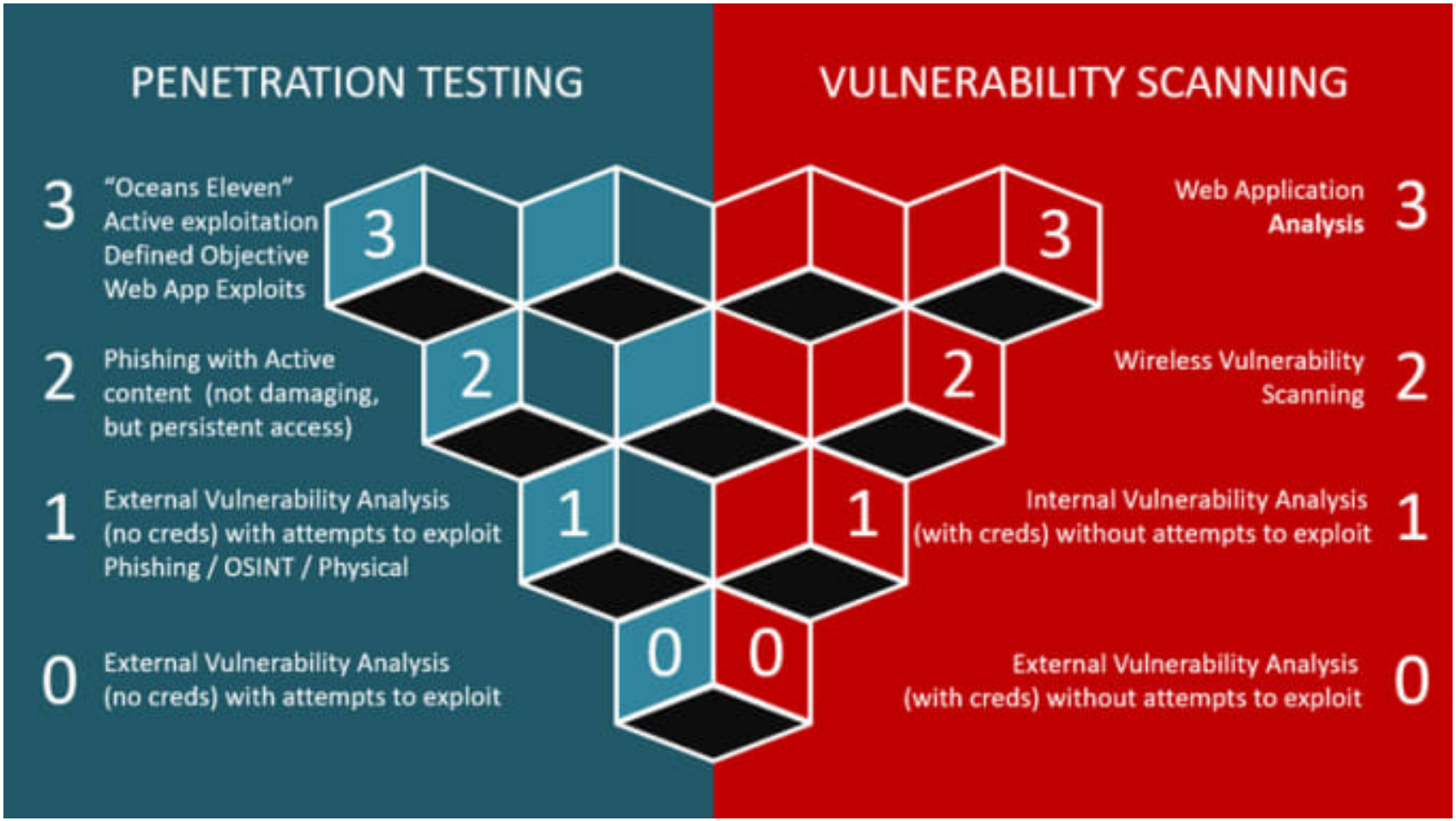


Source: NIST' National Vulnerability Database

Vulnerability discovery

Matasploit

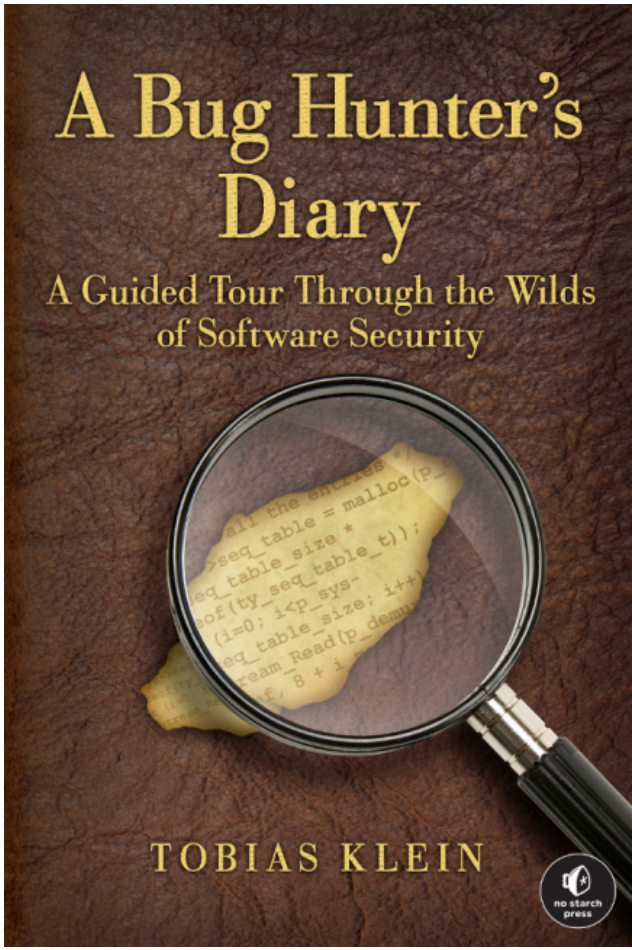
OpenVAS



Vulnerability discovery

- Vulnerabilities are widely different in nature
  - Often implementation-dependent
  - May require deep understanding of sw module interaction
  - Necessary in-depth knowledge of system design
    - e.g. kernel structure, memory allocation,..
- Two main discovery techniques
  - Code lookups (you need source)
    - Manual/semi-automatic search in codebase for known patterns
  - Fuzzing
    - Semi-automatic random input generation--> try to crash program
  - Bonus technique: “Google hacking”
    - Look for known vulnerable functions in google → returns vulnerable webpages

- *Modeling and Discovering Vulnerabilities with Code Property Graphs* F. Yamaguchi, N. Golde, D. Arp, and K. Rieck IEEE Symposium on Security and Privacy (Oakland), 2014



How one teenager is making millions by hacking legally

This is 19-year-old Santiago Lopez from Argentina.

He's the first millionaire bug-bounty hacker, which means he gets paid to find glitches in the software of some of the world's biggest companies.

Mr Lopez made his money on the world's biggest ethical hacking platform: HackerOne.

BBC News' Joe Tidy has been to see how he spends the money.

01 Mar 2019

Vulnerability discovery and disclosure

- Can be found either internally or externally to a company
  - **Internally** → managed within the company (part of Q&A process)
    - Patch (fixing) prioritisation
    - Communication to customers
  - **Externally** → found by an external security researcher
    - Disclosure to vendor
      - Payment
    - Patching prioritisation
    - Disclosure to public

Vulnerability handling

- Internal process must
  - Accept information about new vulnerabilities
    - Internal or external sources
  - Verify vulnerability report
  - If vulnerability exists
    - Develop resolution
    - Post-resolution activities
- ISO 30111

Vulnerability handling – verification phases

- Initial investigation
  1. The reported problem is a security vulnerability
     - Must have repercussions over security policy
  2. The vulnerability affects a supported version of the software the vendor maintains (e.g. not caused by 3rd party modules).
     - Else, exit process
  3. The vulnerability is exploitable with currently known techniques
     - Else, exit process
  4. Root cause analysis
     - Underlying causes of vulnerability and look for similar problems in the code
  5. Prioritisation
     - Evaluate potential threat posed by the vulnerability

Vulnerability handling – resolution and release phases

Resolution decision

- Vendor must decide how to resolve the vulnerability
- Different decisions for different types of vulnerabilities
  - Configuration vulnerabilities → advisory may be enough
  - Code vulnerabilities → patch
  - Critical vulnerabilities → release a mitigation before full patch

Remediation development

- Every resolution must be tested before being delivered to clients
  - minimize negative impacts caused by software change

Release

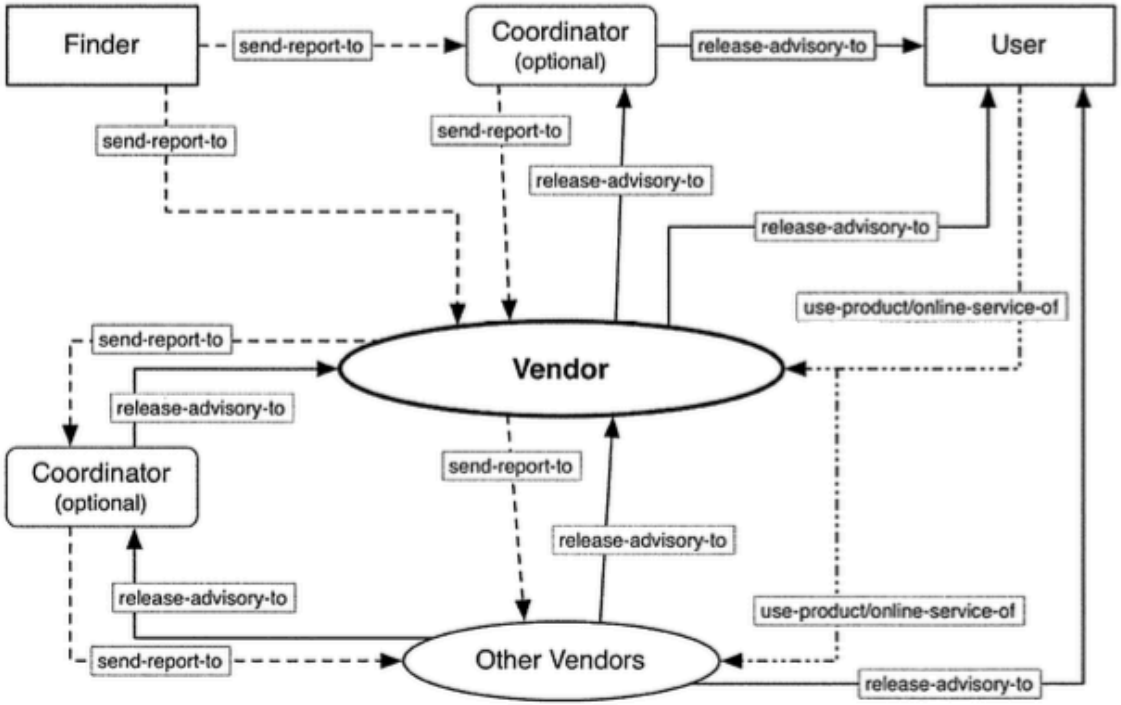
- Web services → vendor deploys patch itself
- Stand-alone product → patch release (see ISO 29147)

Post-release

- Monitor situation (e.g. patch may not be always effective)
  - Support to final client

Vulnerability disclosure

- Vulnerabilities are information sets
- The vulnerability disclosure process is about information exchange – ISO 29147
  - Finder → vendor
  - Vendor → user



Picture from ISO 29147

- Standards related to vulnerability exchange
  - STIX and TAXII

Confidentiality of vulnerability information

Vulnerability information is considered sensitive and confidential by vendors

Pose a threat to end users

May affect vendor's reputation

Build secure communication channels to preserve confidentiality and integrity of information

Vulnerability advisories are typically published after patching

Internal policies determine whether a vulnerability will be published or not

- Typically a function of vulnerability severity

Issues with vulnerability disclosure – the case of external finders

- Security researcher that finds vulnerability may expect
  - Economic return
  - Credit (to mention on curriculum)
- Issue → how to communicate vulnerability to vendor?
  - Say too little → vulnerability not reproducible → no $$$
  - Say too much → vulnerability fully known → thanks for the info → no $$$
- Agreement between sec researcher and vendor
  - Third party mediates (e.g. ZDI)
  - Bug bounty programs (e.g. Microsoft, Google) Zerodium to know the quotation.
  - Credit assured (e.g. Apple?)
- Often involves development of Proof-of-Concept exploit that shows the vulnerability is exploitable
- For more on vuln. disclosure issues see “Miller, Charlie. "The legitimate vulnerability market: Inside the secretive world of 0-day exploit sales." *In Sixth Workshop on the Economics of Information Security*. 2007.”

Third party mediators

- Several on the market, act as proxy between security researcher and vendor
  - Communicate vulnerability to vendor
  - Hold vulnerability information for a certain amount of time (typically 60-90 days)
  - When hold period expires they disclose the vulnerability
    - Mechanism to push vendors to patch
  - Secunia, ZDI, SecurityFocus, Zerodium, …
- If vulnerability is known before vendor releases patch → "zero day vulnerability"
- Google Zero Day Project
  - Discover vulnerabilities (often in competitors' software)
  - Aggressively release vuln info after deadline expires

ZERODIUM Payouts for Desktops/Servers*

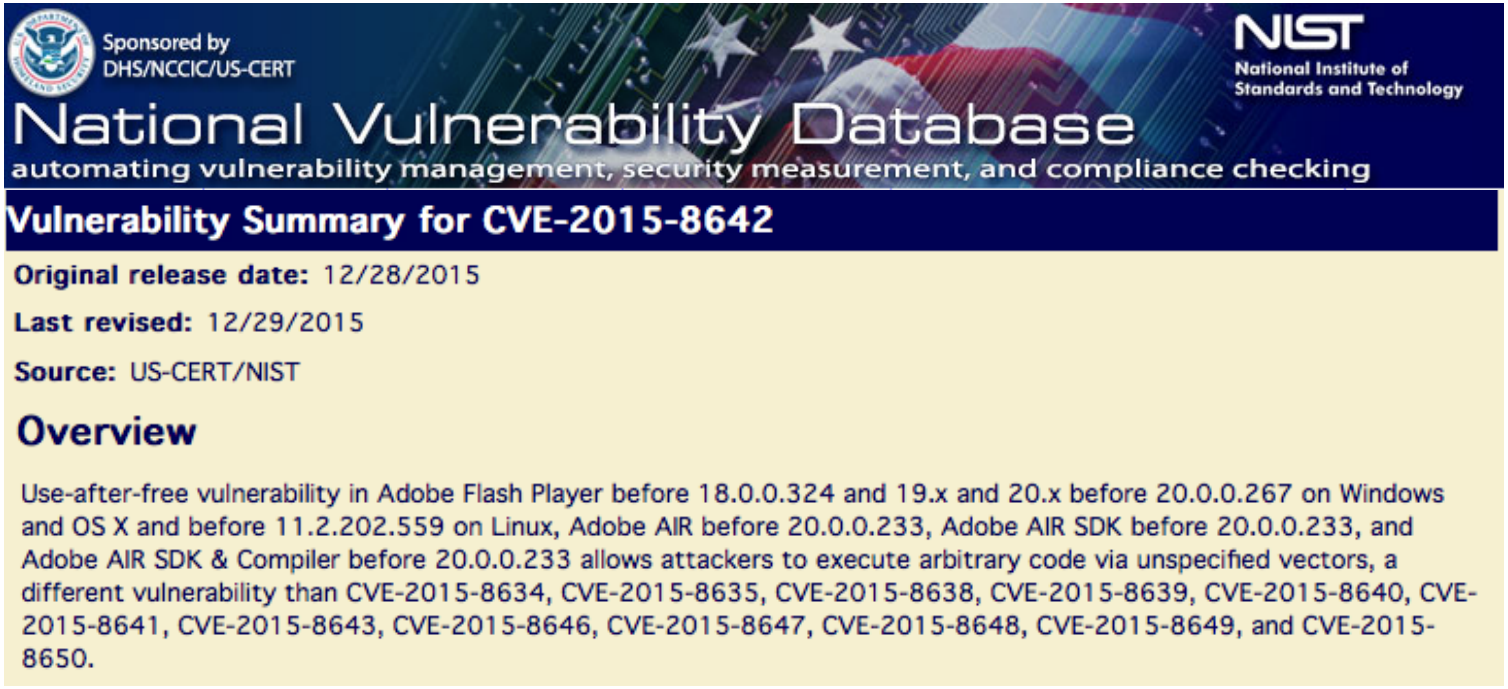
Windows
macOS
Linux/BSD
Any OS

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		

National vulnerability database

- NVD for short → NIST-maintained database of disclosed vulnerabilities
  - The “universe” of vulnerabilities



National Vulnerability Database (2)

Common Platform Enumeration (CPE)

+ Configuration 1
    + AND
        + OR
            * cpe:/a:adobe:air_sdk:20.0.0.204 and previous versions
            * cpe:/a:adobe:air_sdk_%26_compiler:20.0.0.204 and previous versions
        + OR
            cpe:/o:apple:mac_os_x
            cpe:/o:apple:iphone_os
            cpe:/o:google:android
            cpe:/o:microsoft:windows
+ Configuration 2
    + AND
        + OR
            * cpe:/a:adobe:flash_player:20.0.0.235
            * cpe:/a:adobe:flash_player:20.0.0.228
            * cpe:/a:adobe:flash_player:19.0.0.245
            * cpe:/a:adobe:flash_player:19.0.0.226

Vulnerability feeds

- Vulnerabilities are disclosed by publication in the NVD and other vulnerability feeds
  - Public and private
- Private feeds release information earlier
  - “early advisories”
  - Secunia, SecurityFocus, ZDI
- Public feeds typically release weekly or monthly updates
  - SANS@RISK
  - https://www.sans.org/newsletters/at-risk

Types of vulnerabilities

- Different types of vulnerabilities
- “The Open Web Application Security Project (OWASP) is a 501(c)(3) worldwide not-for-profit charitable organization focused on improving the security of software”
  - https://www.owasp.org/index.php/Main_Page
- Good resource for information security resources
- “Top 10 vulnerability threats”
  - Good overview of most common vulnerability types with examples

OWASP Top 10

A1	Injection
A2	Broken Authentication
A3	Sensitive Data Exposure
A4	XML External Entities
A5	Broken Access Control
A6	Security Misconfiguration
A7	Cross Site Scripting (XSS)
A8	Insecure Deserialization
A9	Using Components with Known Vulnerabilities
A10	Insufficient Logging and Monitoring

https://owasp.org/www-project-top-ten/

Common Weakness Enumeration
A Community-Developed List of Software & Hardware Weakness Types

CWE TOP 25 Most Dangerous Software Weaknesses

Home > CWE List > CWE- Individual Dictionary Definition (4.4) ID Lookup: Go

Home | About | CWE List | Scoring | Community | News | Guidance | Search

CWE VIEW: Software Development

View ID: 699 **Status:** Draft
Type: Graph

Downloads: Booklet | CSV | XML

Objective

This view organizes weaknesses around concepts that are frequently used or encountered in software development. This includes all aspects of the software development lifecycle including both architecture and implementation. Accordingly, this view can align closely with the perspectives of architects, developers, educators, and assessment vendors. It provides a variety of categories that are intended to simplify navigation, browsing, and mapping.

Audience

Stakeholder	Description
Software Developers	Software developers (including architects, designers, coders, and testers) use this view to better understand potential mistakes that can be made in specific areas of their software application. The use of concepts that developers are familiar with makes it easier to navigate this view, and filtering by Modes of Introduction can enable focus on a specific phase of the development lifecycle.
Educators	Educators use this view to teach future developers about the types of mistakes that are commonly made within specific parts of a codebase.

Relationships

The following graph shows the tree-like relationships between weaknesses that exist at different levels of abstraction. At the highest level, categories and pillars exist to group weaknesses. Categories (which are not technically weaknesses) are special CWE entries used to group weaknesses that share a common characteristic. Pillars are weaknesses that are described in the most abstract fashion. Below these top-level entries are weaknesses are varying levels of abstraction. Classes are still very abstract, typically independent of any specific language or technology. Base level weaknesses are used to present a more specific type of weakness. A variant is a weakness that is described at a very low level of detail, typically limited to a specific language or technology. A chain is a set of weaknesses that must be reachable consecutively in order to produce an exploitable vulnerability. While a composite is a set of weaknesses that must all be present simultaneously in order to produce an exploitable vulnerability.

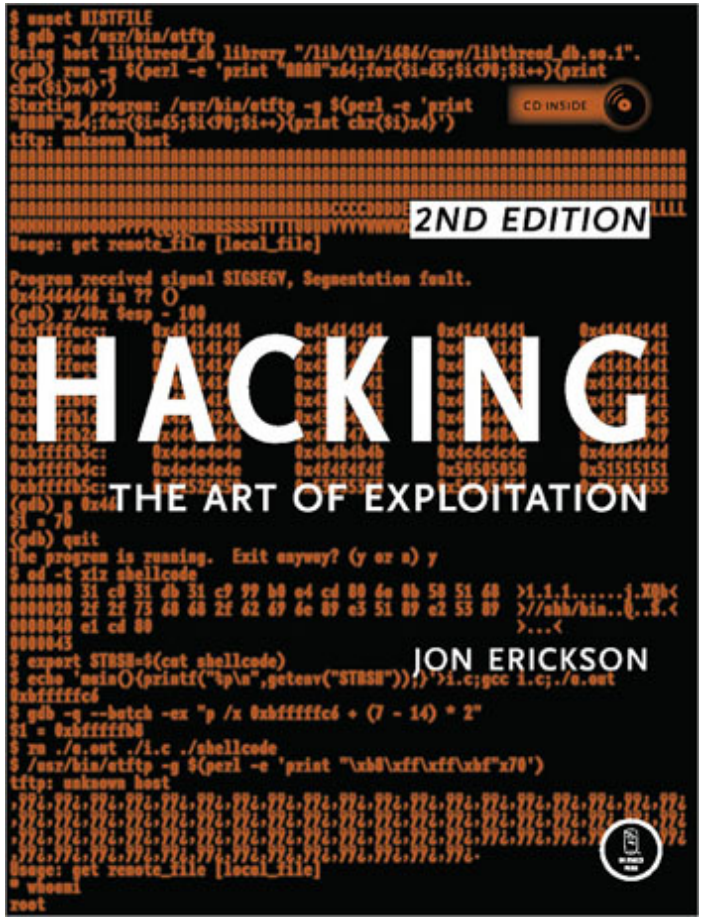
Show Details:

Expand All | Collapse All | Filter View

699 - Software Development

- C API / Function Errors - *(1228)*
- C Audit / Logging Errors - *(1210)*
- C Authentication Errors - *(1211)*
- C Authorization Errors - *(1212)*
- C Bad Coding Practices - *(1006)*
- C Behavioral Problems - *(438)*
- C Business Logic Errors - *(840)*
- C Communication Channel Errors - *(417)*
- C Complexity Issues - *(1226)*
- C Concurrency Issues - *(557)*
- C Credentials Management Errors - *(255)*

Reference to technical details



ACM Code of Ethics

GENERAL ETHICAL PRINCIPLES.

1.1 Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing.

1.2 Avoid harm.

1.3 Be honest and trustworthy.

1.4 Be fair and take action not to discriminate.

1.5 Respect the work required to produce new ideas, inventions, creative works, and computing artifacts.

1.6 Respect privacy.

1.7 Honor confidentiality.

PROFESSIONAL RESPONSIBILITIES.

2.7 Foster public awareness and understanding of computing, related technologies, and their consequences.

2.8 Access computing and communication resources only when authorized or when compelled by the public good.

2.9 Design and implement systems that are robustly and usably secure.

PROFESSIONAL LEADERSHIP PRINCIPLES.

3.7 Recognize and take special care of systems that become integrated into the infrastructure of society.

Association for Computing Machinery

Why to grade vulnerabilities?

- Central question:

  → How severe are the security problems affecting my software configuration?

- Not all vulnerabilities are the same
  - XSS vs BoF vs SQLi vs Privilege escalation vs …
  - Vulnerability counting can NOT be a measure of severity

  → What is the threat level of your systems?

  → Clients and users should be informed too

    → Not all users are “security experts”

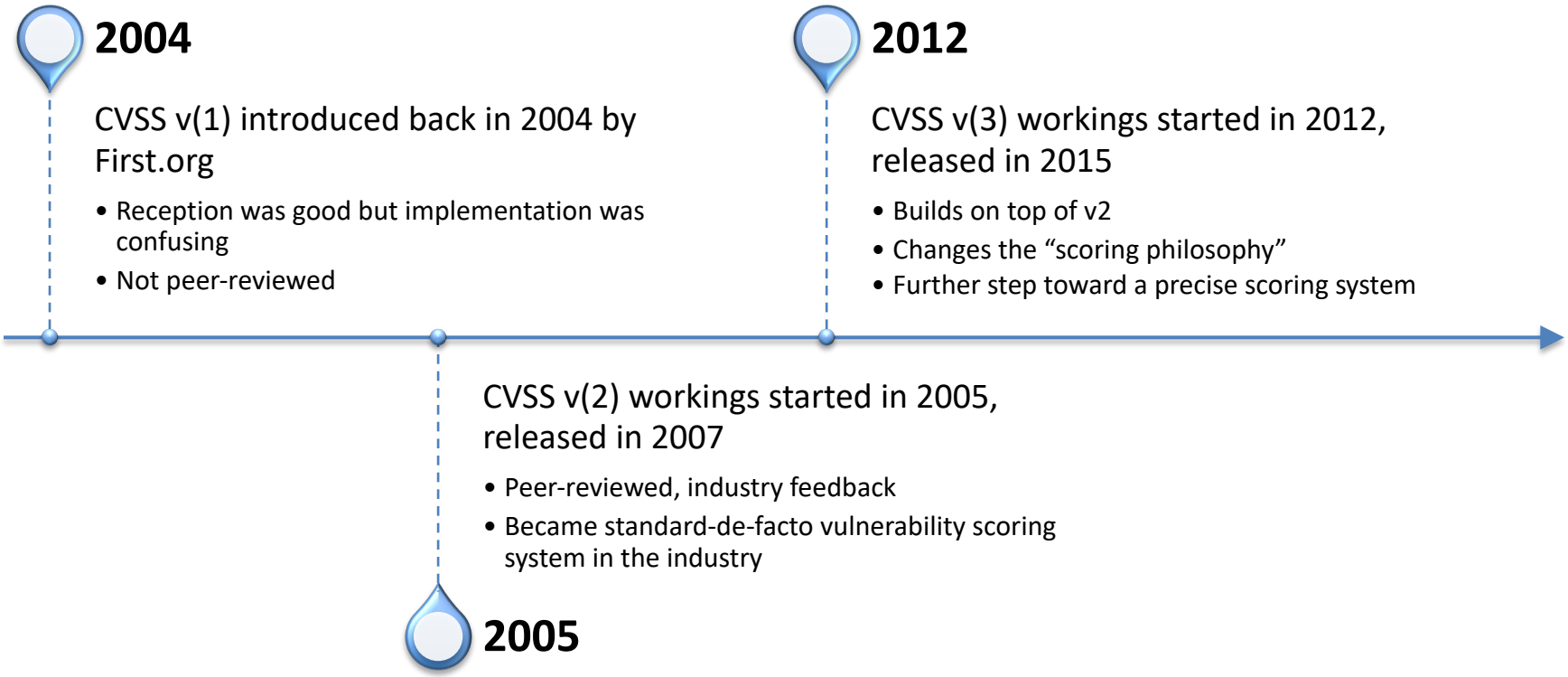
    → “IT knowledge” can be assumed

  → How to measure/communicate a security issue?

The Common Vulnerability Scoring System

- CVSS is an open framework for communicating the characteristics and severity of software vulnerabilities.
- Goal is to have a **shared system of metrics** to **analyze and measure vulnerabilities**
  - Different users score the same vuln in the same way → severity assessment
  - Different people "read" the same vuln and understand the same thing → severity communication

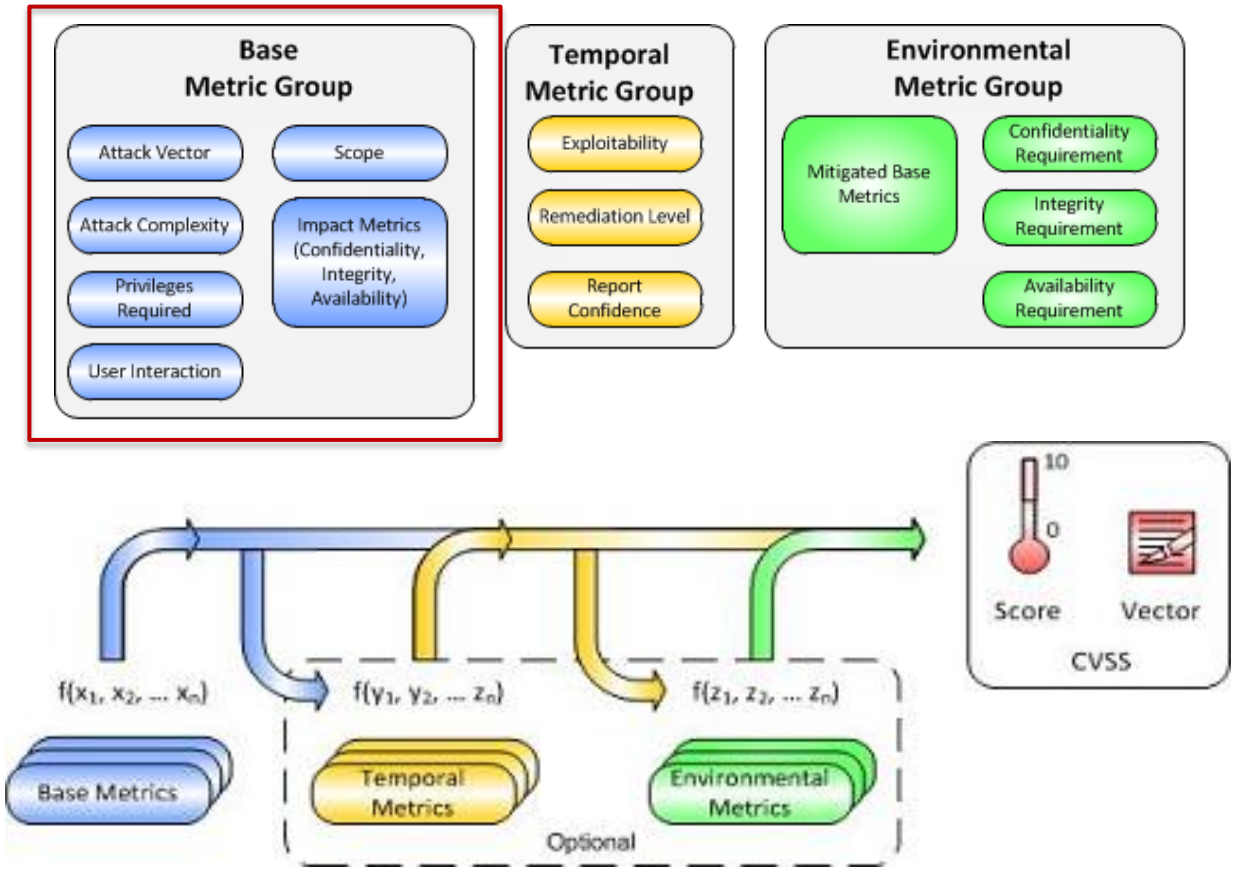
CVSS v(x) walkthrough



CVSS v3

http://www.first.org/cvss/v3/development

- CVSS is based on three metric groups



CVSS Base metric overview

- Exploitability metrics
  - Attack Vector
  - Attack Complexity
  - User Interaction
  - Privileges Required

Measured over the vulnerable component

- Scope metric

Security Authority of Vulnerable Component = Security Authority of Impacted Component?

- Impact metrics
  - Confidentiality
  - Integrity
  - Availability

Measured over the impacted component

Expl. Metrics: Attack Vector

- This metric reflects the context in which the vulnerability exploitation occurs.
- The more remote an attacker (or the attack) can be from the target, the greater the vulnerability score.
- Possible values:
  1. **Network**: exploitation is bound to the network stack
  2. **Adjacent Network**: attacker needs to be in same subnet
  3. **Local**: attack is not bound to network stack, but rather to I/O on system. In some cases, the attacker may be logged in locally in order to exploit the vulnerability, otherwise, she may rely on User Interaction to execute a malicious file.
  4. **Physical**: attacker must be physically operating over the vulnerable component

Expl. Metrics: Attack Complexity

- This metric describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability.
- Possible values:
  1. **High**: A successful attack depends on conditions outside the attacker's control. That is, **a successful attack** cannot be accomplished , but **requires the attacker to invest in some measurable amount of effort in preparation or execution against the vulnerable component** before a successful attack can be expected.
  2. **Low:** Specialized access conditions or extenuating circumstances do not exist. An attacker can expect repeatable exploit success against a vulnerable target

Examples for Attack Complexity: High

- For example, a successful attack may depend on an attacker overcoming any of the following conditions:
  1. The attacker must conduct **target-specific reconnaissance**. For example, on target configuration settings, sequence numbers, shared secrets, etc.
  2. The attacker must **prepare the target environment** to improve exploit reliability. For example, repeated exploitation to win a race condition, or overcoming advanced exploit mitigation techniques.
  3. The attacker **injects herself into the logical network path** between the target and the resource requested by the victim in order to read and/or modify network communications (e.g. man in the middle attack).

Expl. Metrics: Privileges Required

- This metric describes the level of privileges an attacker must possess before successfully exploiting the vulnerability.
- Possible values:
  1. High: The attacker is authorized with (i.e. requires) privileges that provide significant (e.g. administrative) control over the vulnerable component that could affect component-wide settings and files.
  2. Low: The attacker is authorized with (i.e. requires) privileges that provide basic user capabilities that could normally affect only settings and files owned by a user. Alternatively, an attacker with Low privileges may have the ability to cause an impact only to non-sensitive resources.
  3. None: The attacker is unauthorized prior to attack, and therefore does not require any access to settings or files to carry out an attack.

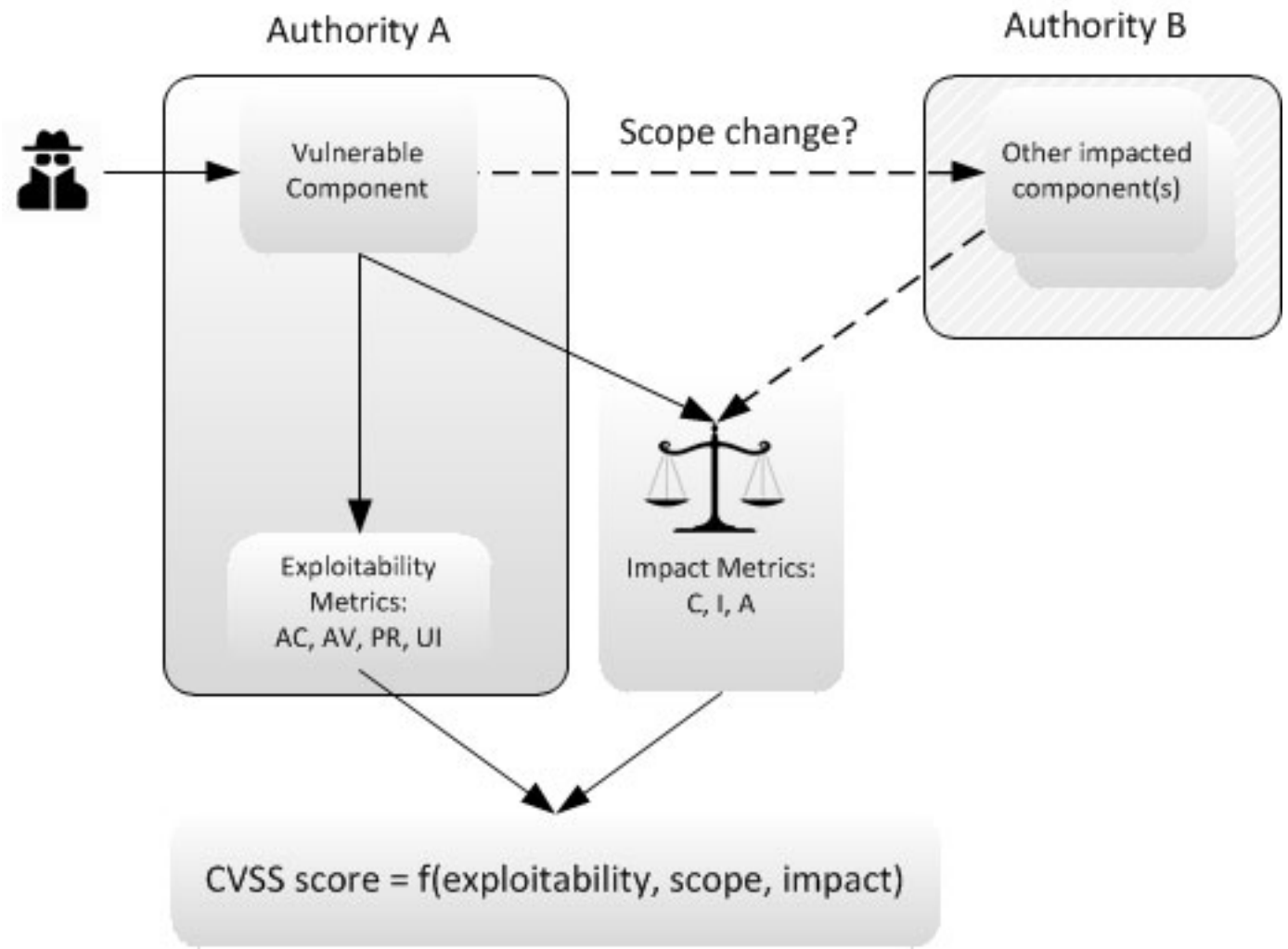
Expl. Metrics: User Interaction

- This metric captures the requirement for a user, other than the attacker, to participate in the successful compromise the vulnerable component.
- This metric determines whether the vulnerability can be exploited solely at the will of the attacker, or whether a separate user (or user-initiated process) must participate in some manner.
- Possible values:
  1. Required: Successful exploitation of this vulnerability requires a user to take some action before the vulnerability can be exploited. For example, a successful exploit may only be possible during the installation of an application by a system administrator.
  2. None: The vulnerable system can be exploited without any interaction from any user.

Scope (1)

- Scope refers to the collection of privileges defined by a security authority (e.g. an application, an operating system, or a sandbox environment) when granting access to computing resources (e.g. files, CPU, memory, etc). These privileges are assigned based on some method of identification and authorization.
- When the vulnerability of a software component governed by one security authority is able to affect resources governed by another security authority, a Scope change has occurred.

Scope (2)


Authority A
Authority B
Vulnerable Component
Scope change?
Other impacted component(s)
Exploitability Metrics: AC, AV, PR, UI
Impact Metrics: C, I, A
CVSS score = f(exploitability, scope, impact)

Scope (3)

- Possible values:
  - Unchanged: An exploited vulnerability can only affect resources managed by the same authority. In this case the vulnerable component and the impacted component are the same.
  - Changed: An exploited vulnerability can affect resources beyond the authorization privileges intended by the vulnerable component. In this case the vulnerable component and the impacted component are different.

Impact metrics

- Measures the losses on
  - Confidentiality, → impact on confidentiality of **data**
    - *property that information is not made available or disclosed to unauthorized individuals, entities, or processes*
  - Integrity, → impact on integrity of **data**
    - *the "property of accuracy" of information*
  - Availability → impact on availability of **the component**
    - is the "property of being accessible and usable upon demand by an unauthorized entity"
- Each metric measures the losses **suffered by the impacted component**
- Possible values:
  1. High → total loss
  2. Low → partial loss
  3. None → no loss

Scoring Guide/Philosophy

- Access Vector → is the attack bound to the network stack?
- Attack Complexity → can the attacker control all factors relevant to the exploitation?
- Privileges Required → does the attacker need be authenticated?
- User Interaction → does the victim user need to interact with the attack?
- Scope → is the authorisation authority under which the vulnerable component is the same as the impacted component?
- Impact
  - Confidentiality, Integrity → Data
  - Availability → Service
- Scoring rule: When more than one assessment is possible, go with the more severe one
  - e.g. exploitation can happen both though local I/O and on network stack → go with network

Scoring Exercise (1)

- MS Word Denial-of-Service attack (CVE-2013-6801)
  - Microsoft Word 2003 SP2 and SP3 on Windows XP SP3 allows remote attackers to cause a denial of service (CPU consumption) via a malformed .doc file containing an embedded image, as demonstrated by word2003forkbomb.doc, related to a "fork bomb" issue.

Access Vector	
Access Complexity	
Privileges Required	
User Interaction	
Scope	
Confidentiality	
Integrity	
Availability	

Scoring Exercise (2)

- CISCO host crash (CVE-2011-0355)
  - Cisco Nexus 1000V Virtual Ethernet Module (VEM) 4.0(4) SV1(1) through SV1(3b), as used in VMware ESX 4.0 and 4.1 and ESXi 4.0 and 4.1, does not properly handle dropped packets, which allows guest OS users to cause a denial of service (ESX or ESXi host OS crash) by sending an 802.1Q tagged packet over an access vEthernet port, aka Cisco Bug ID CSCtj17451.

Access Vector	
Access Complexity	
Privileges Required	
User Interaction	
Scope	
Confidentiality	
Integrity	
Availability	

Scoring Exercise (3)

- CVE-2009-0927
  - Stack-based buffer overflow in Adobe Reader and Adobe Acrobat 9 before 9.1, 8 before 8.1.3 , and 7 before 7.1.1 allows remote attackers to execute arbitrary code via a crafted argument to the getIcon method of a Collab object, a different vulnerability than CVE-2009-0658.

Access Vector	
Access Complexity	
Privileges Required	
User Interaction	
Scope	
Confidentiality	
Integrity	
Availability	

An alternative score for this vuln exists.
If one assumes that the vuln requires some pdf file to be opened by a. Reader, then we have:

- AV:L/UI:R

In this case we went with the one that gives the higher severity (AV:N,UI:N)

Scoring Exercise (4)

- Libvirt USB handling (CVE-2012-2693)
  - libvirt, possibly before 0.9.12, does not properly assign USB devices to virtual machines when multiple devices have the same vendor and product ID, which might cause the wrong device to be associated with a guest and might allow local users to access unintended USB devices.

Access Vector	
Access Complexity	
Privileges Required	
User Interaction	
Scope	
Confidentiality	
Integrity	
Availability	

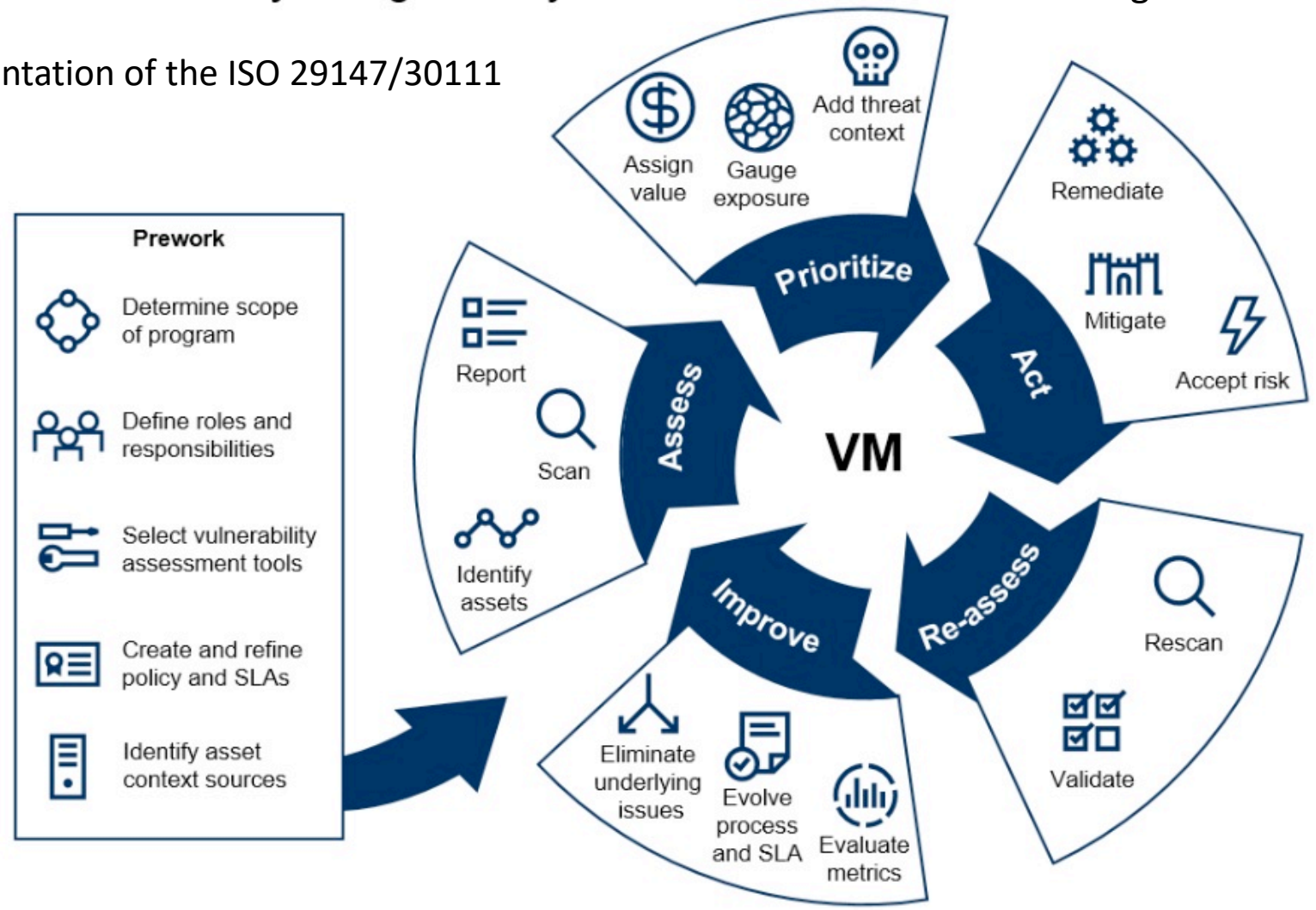
Keys

- Ex1:Local,Low,None,Required,Unchanged,None,None,High
- Ex2:Adjacent Network,Low,None,None,Changed,None,None,High
- Ex3:Network,Low,None,None,Unchanged,High,High,High
- Ex4: Local,High,Low,None,Change,Low,Low,Low

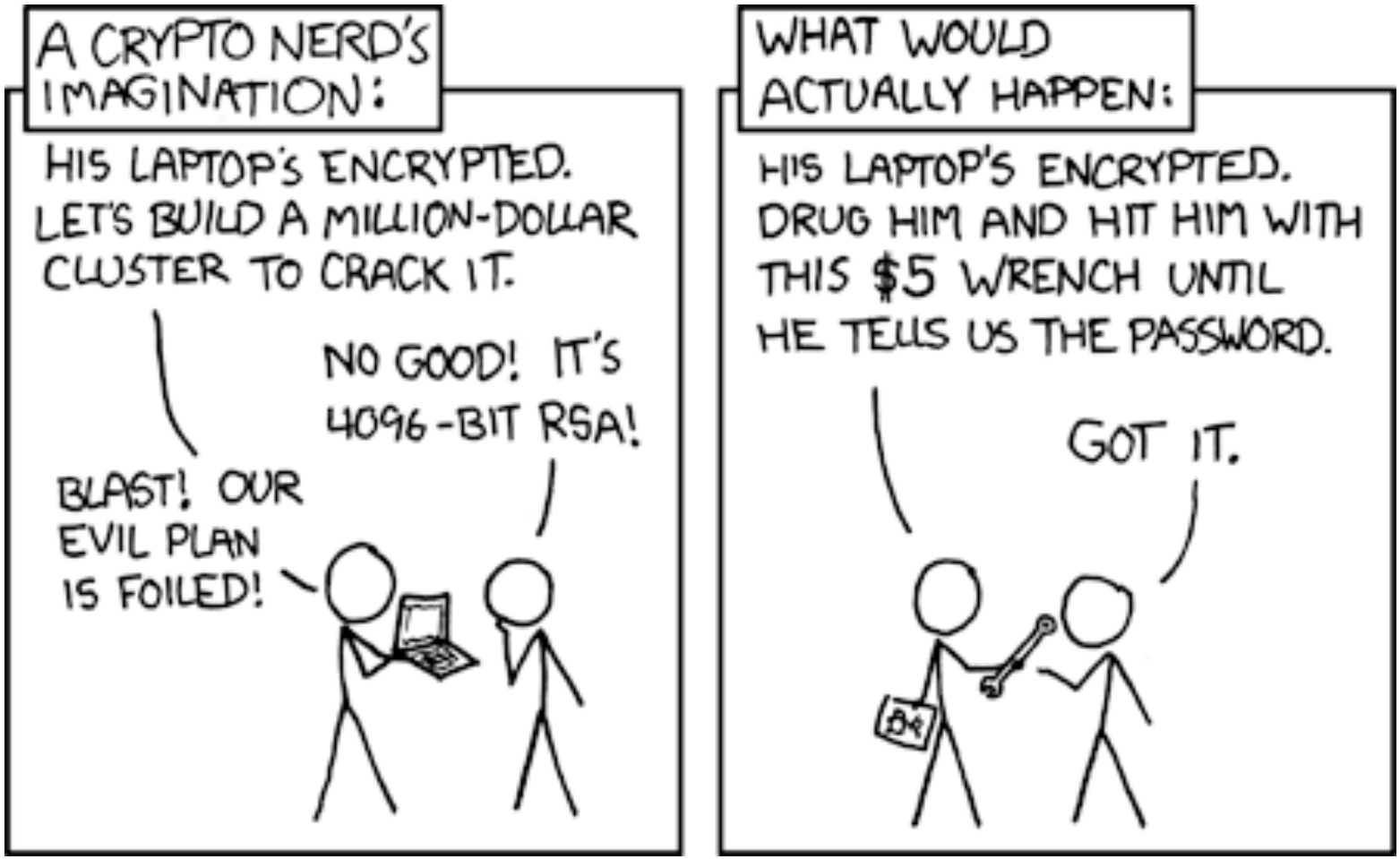
Vulnerability management

Management View

Implementation of the ISO 29147/30111



Security is about people

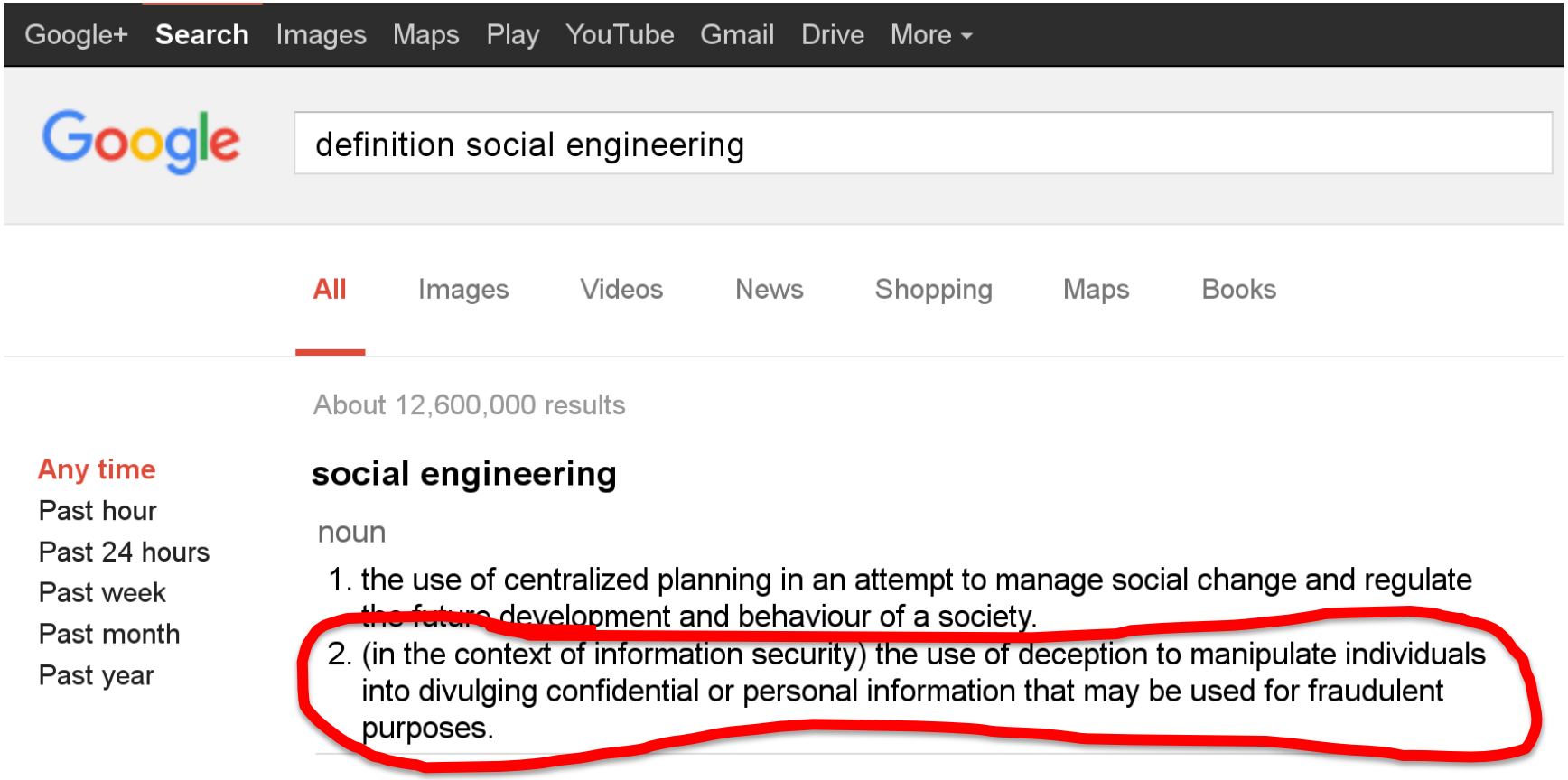


Human vulnerabilities

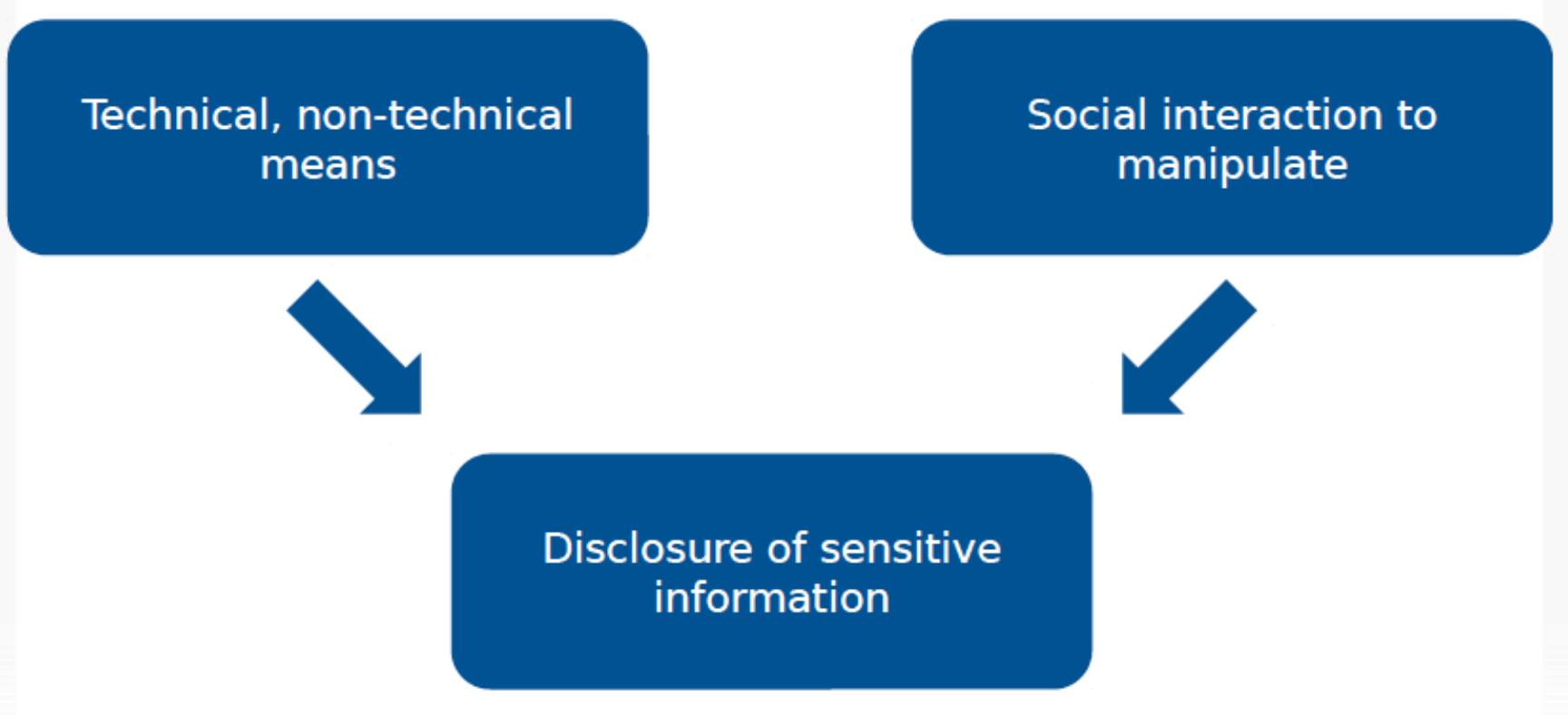
"The biggest threat to the security of a company is not a computer virus, an unpatched hole in a key program or a badly installed firewall. In fact, the biggest threat could be you. What I found personally to be true was that it's easier to manipulate people rather than technology. Most of the time organizations overlook that human element"

Kevin Mitnick

Social engineering



Social engineering



Phishing

- The attacker aims at obtaining the credentials of users of a website/service
  - other types of private information can be gathered too
  - Typically through more sophisticated “spear phishing” attacks
- Attacker creates a *replica* of the original website
  - Replica is published online
  - Link typically sent through spam emails, social networks
  - Recipient may be fooled in opening the link and entering their credentials as in the genuine website
  - Credentials are of course sent to the attacker instead

Phishing – attacker tools

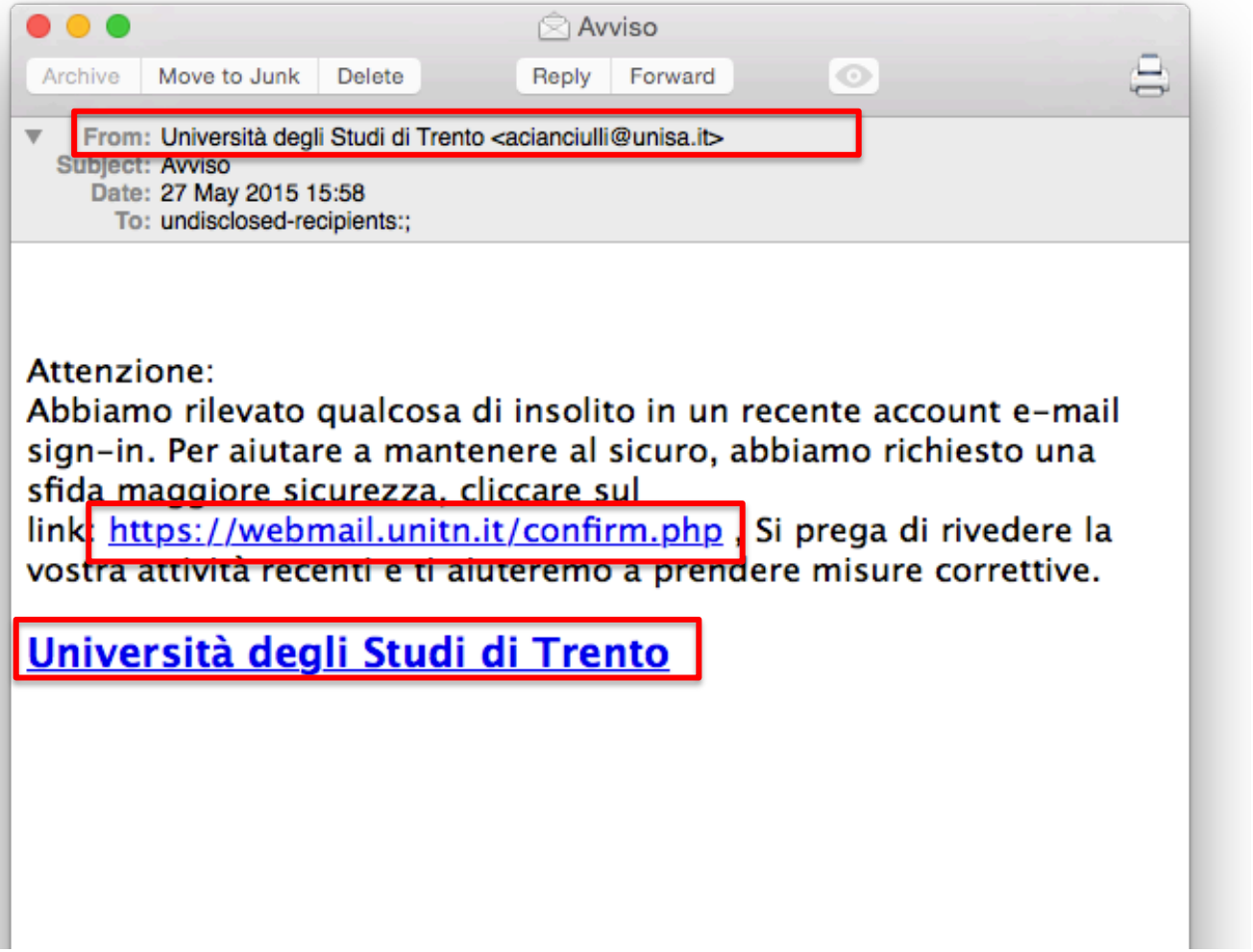
- Creating a working replica of a website is only as hard as creating a copy
  - Attacker needs to modify some of its components
    - e.g. send form HTTP POST to a webserver the attacker controls
  - Advanced attackers may remove JS/third party components to prevent exposing the phishing website
    - Advanced attackers vs script kiddies
- Automated tools exist that do this for the attacker
  - Few hundreds of dollars on black markets
  - Essentially a recursive wget

Phishing in a nutshell

Phishing example

Translation (including English reproduction of lexical and grammatical errors).

Warning:
We noticed something unusual in a recent email account sign-in. To help maintaining secure, we requested a challenge higher security. click the link {link}, We kindly ask to review your activities recent and we will help you taking correcting measures.



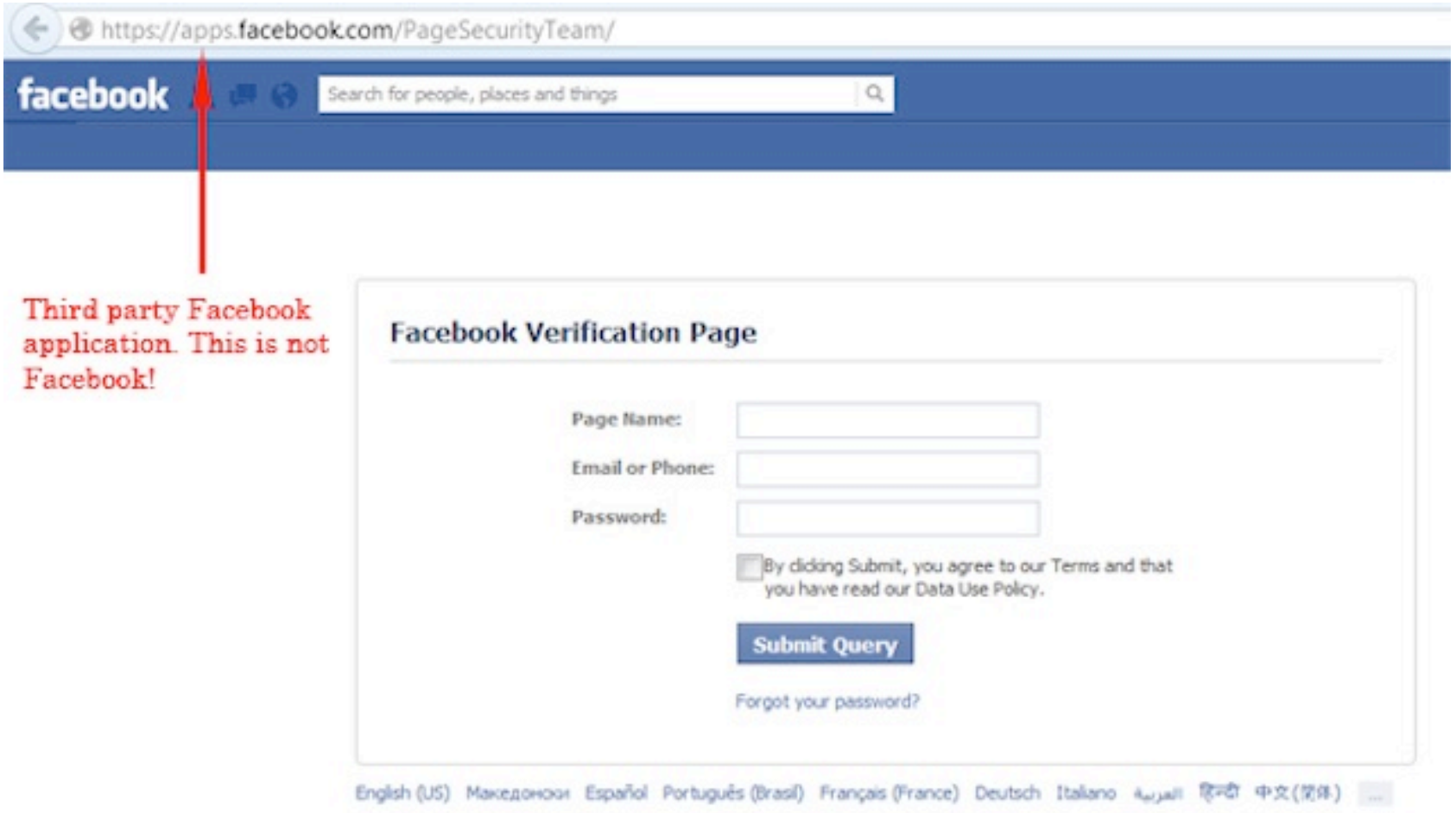
Combining phishing and software vulnerabilities

- In this case it's easy to notice that the domain I'm redirected to is not UniTn's
- However, there exist vulnerabilities in browsers that allow the malicious website to **spoof** the address displayed in the address bar
- Example:
  - The webpage is **gfcv-altervista.org**
  - The browser says it's **webmail.disi.unitn.it**

Example of address spoofing

- Safari 8 vulnerability under OSX < 10.10.5
  - Other similar vulnerabilities exist for IE and Chrome
- If browser is vulnerable, attacker can manipulate address bar's content to his/her liking

Example

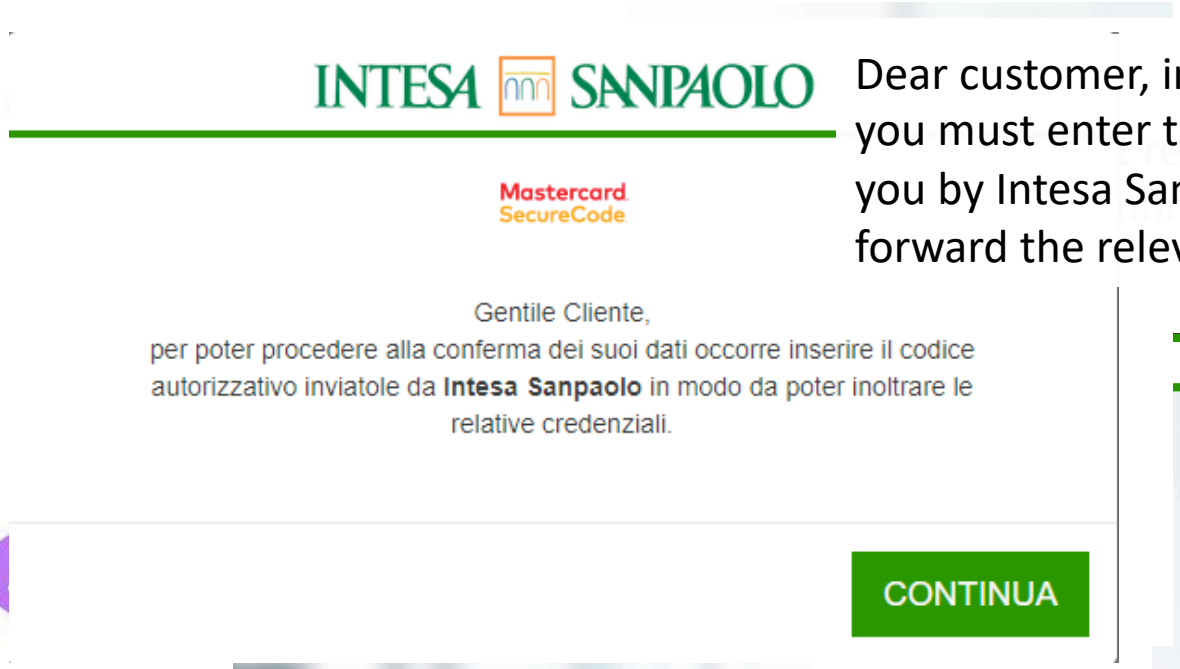

https://apps.facebook.com/PageSecurityTeam/
facebook
Search for people, places and things
Third party Facebook application. This is not Facebook!
Facebook Verification Page
Page Name:
Email or Phone:
Password:
By clicking Submit, you agree to our Terms and that you have read our Data Use Policy.
Submit Query
Forgot your password?
English (US) Español Português (Brasil) Français (France) Deutsch Italiano

Vishing

edgeverve.com

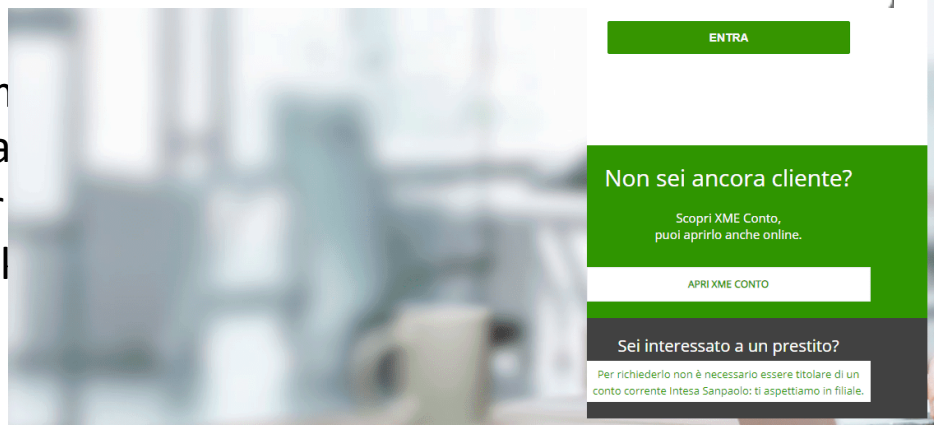
https://www.youtube.com/watch?v=F78UdORll-Q

Smishing



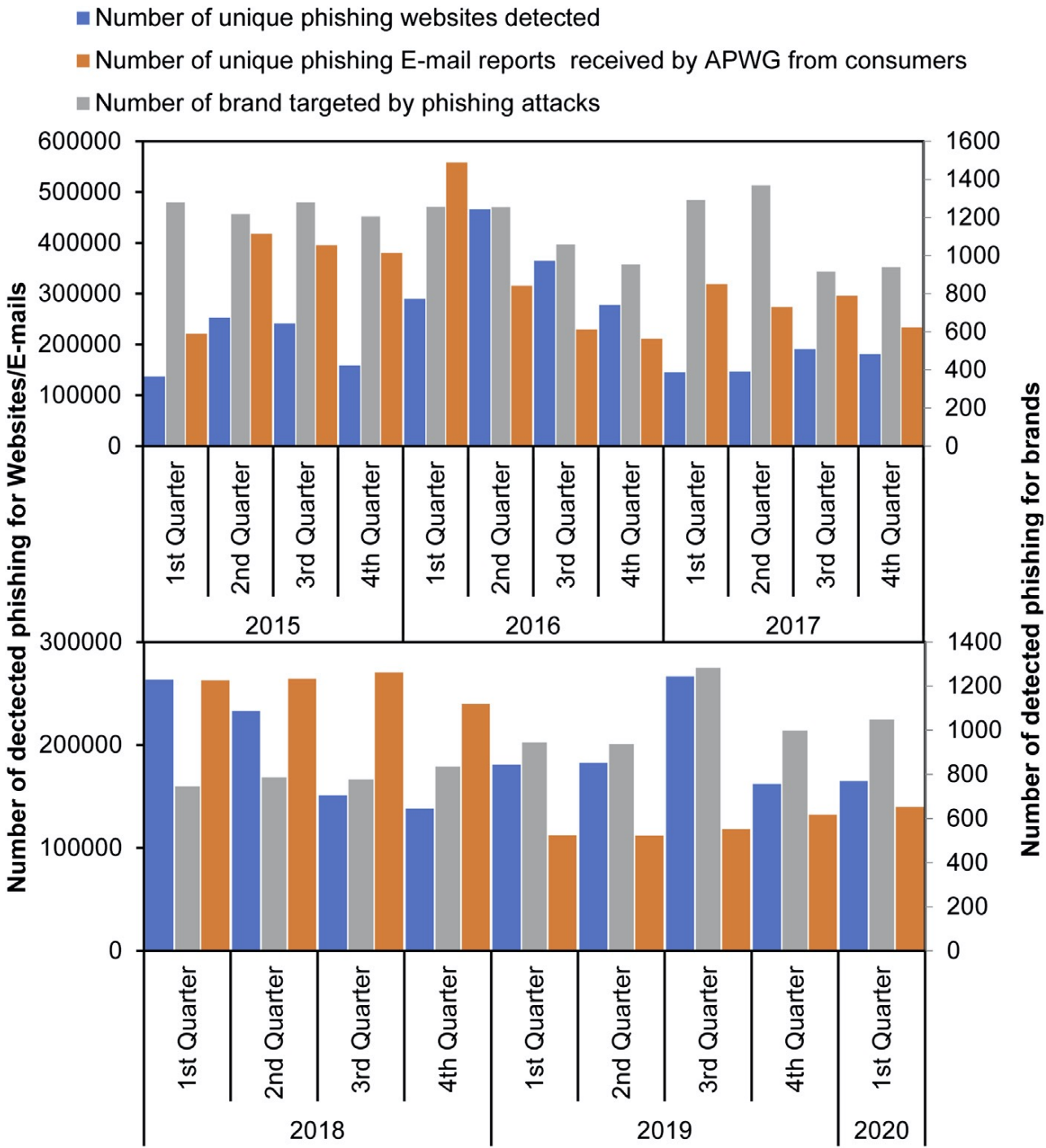
Dear customer, in order to confirm your data, you must enter the authorization code sent to you by Intesa Sanpaolo in order to be able to forward the relevant credentials

Dear Custom
anomalous a
please enter
following linl



"Phishing Attacks: A Recent Comprehensive Study and a New Anatomy"
Frontiers in Computer Science · February 2021

Other source: APWG (Anti-Phishing Working Group) Phishing Activity Trends Report

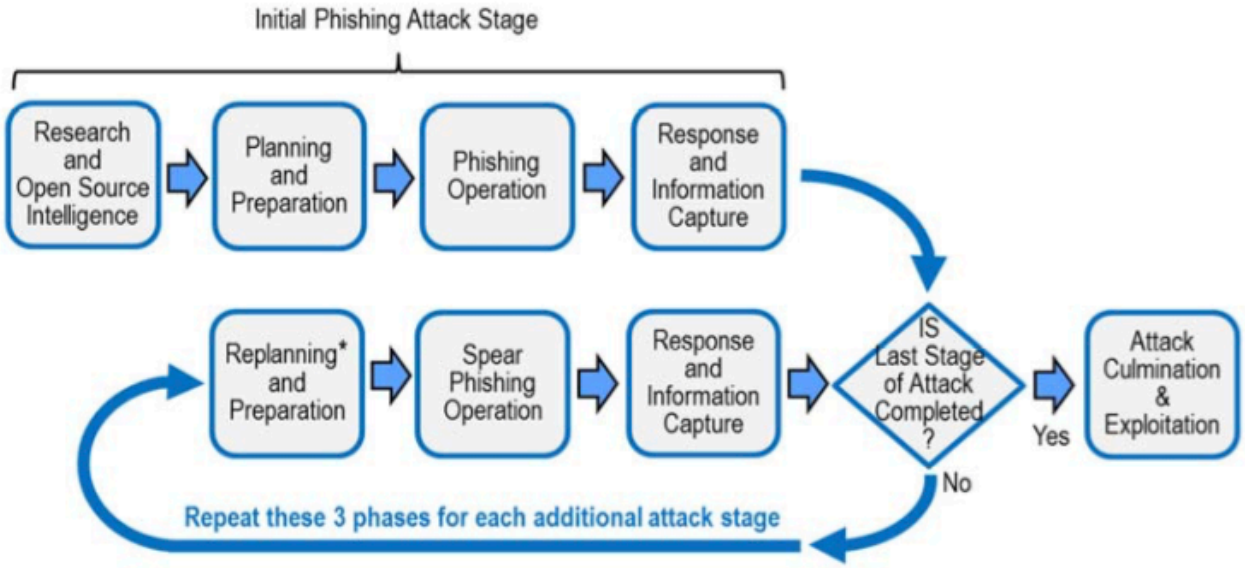


Multi-stage attack

- Can distinguish between single and multiple-stage social engineering attacks
- Single stage attacks usually aim at collecting sensitive information about “general” targets
  - No specificity in the attack
    - e.g. attack all costumers of mybank.com

Spear Phishing

- Two-stage attacks involve an initial reconnaissance that gathers information needed for second stage
  - Used to increase credibility of attack
    - E.g. proper legal references, employee names, correct set of users in CC to phishing email, etc
  - Spearphishing against CEO/director/manager/person of interest



Steps in detail (first stage)

Pattern Phase	Typical Activities	Pattern Interactions
1. Research and Open Source Intelligence	• Search for opensource intelligence • Establish attack objectives • Identify opportune targets	1.1 Attacker researches and strategizes about potential targets and specific objectives.
2. Planning and Preparation	• Develop attack strategy including means to avoid detection and mitigation by UIT organization • Prepare phishing attack artifacts	2.1 Attacker plans phishing attack and creates phishing artifacts (e.g., phishing email, mobile text message, phony website, malware to be implanted).
3. Phishing Operation	• Release phishing artifact via email, cellphone, rogue website, or other means • Wait for a response	3.1 Attacker initiates phishing attack through email, cellphone, rogue website, or other means.
4. Response and Information Capture	•Gain access and/or privileges to obtain greater information reach •Implant malware to achieve information objectives •Identify other opportune UIT targets and internal system information, and capture guarded and sensitive information	4.1 One or more targets unwittingly respond to phishing artifact and become a UIT. 4.2 Attacker detects or is alerted to UIT response and obtains initial information directly from UIT data entry. 4.3 Attacker implants malware on victim's machine or network. 4.4 Attacker obtains desired information via malware.

Unintentional Insider Threats: Social Engineering. CERT Insider Threat Center. January 2014
http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=77455

Steps in detail (second stage)

Pattern Phase	Typical Activities	Pattern Interactions
5. Re-planning and Preparation	•Re-plan attack strategy including means to avoid detection and mitigation by UIT organization •Prepare spear phishing attack artifacts	5.1 Attacker uses information capture in Step 4 above to replan follow-on steps for spear phishing attack. This may entail creation of new artifacts or specific attack approaches.
6. Spear Phishing Operation	• Execute spear-phishing • Wait for a response	6.1 Attacker initiates spear phishing attack.
7. Response and Information Capture	•Gain access and/or privileges to obtain greater information reach •Exploit more specific insider targets: financial system, secure systems, etc.	7.1 One or more high-value targets unwittingly responds to the spear phishing artifact and becomes a UIT. 7.2 Phisher detects or is alerted to UIT response and obtains desired information directly from UIT data entry.
8. Attack Culmination and Exploitation	• Use captured information to directly attack UIT or UIT's organization to steal, manipulate, and/or destroy targeted assets	8.1 Attacker uses desired information in direct attack on UIT or UIT's organization to steal, manipulate, and/or destroy targeted assets.

Unintentional Insider Threats: Social Engineering. CERT Insider Threat Center. January 2014
http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=77455

Example: well engineered, 2-stage social engineering attack

- On 19th of May 2015 I received an email from somebody attaching a "receipt". The email was in good Italian, and had seemingly meaningful law references regulating the emission of the receipt
  - However, I was not expecting a receipt
  - I discarded it right away as an attack → trashed
- The next day, I receive this email:

Dear costumer,

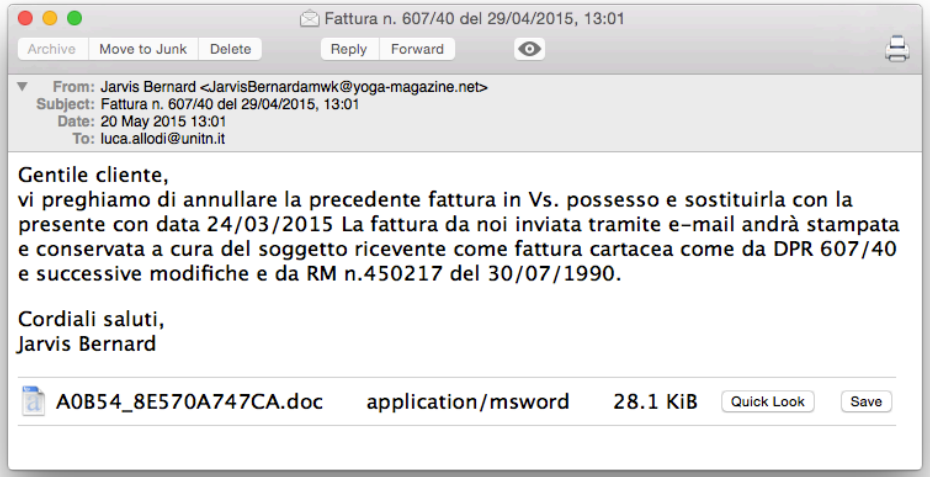
We kindly ask you to ignore the previous receipt and substitute it with the present, dated 24/03/2015 The receipt must be printed and archived by the receiving subject as prescribed by DPR 607/40 and subsequent changes, and by RM no. 450217, emitted on 30/07/1990

Best regards,
Jarvis Bernard

continuance commitment (variation of)

authority

normative commitment



Almost fell for it..

SHA256:	fb4d983c26b0e5d13df260e5da4e9cddf780d2520bb7c4e3440a868b93ad6f94
File name:	99BCA6_B7C8B4025833.doc
Detection ratio:	2 / 57
Analysis date:	2015-05-20 11:09:00 UTC (2 weeks, 5 days ago)

AVware	Trojan.MHT.Agent.a (v)	20150520
VIPRE	Trojan.MHT.Agent.a (v)	20150520
ALYac	✓	20150520
AVG	✓	20150520
Ad-Aware	✓	20150520
AegisLab	✓	20150520
Agnitum	✓	20150519
AhnLab-V3	✓	20150519
Alibaba	✓	20150520
Antiy-AVL	✓	20150520

Reported results are for attachment of first email. Second attachment gave same results.

Social engineering

- Phishing is only an application of a wider set of attacks that exploit human nature to (usually) breach data confidentiality
- “Social engineering” identifies a set of techniques that attack weaknesses in human psychology
  - The final goal is to *persuade a human being* in performing actions elicited by the attacker

Social engineering

- *Situational theory of publics* → This theory explains when people communicate and when communications aimed at people are most likely to be effective
  - **Problem recognition** → subject thinks the problem is relevant to them
  - **Active involvement** → subject thinks they will suffer the consequences of the threat
  - **Constraint recognition** →subject thinks their actions are limited by factors outside of their control
- https://www.social-engineer.com/category/video/

Elaboration Likelihood Model (ELM)

- ELM describes the ways humans change their attitudes or decide to perform actions they would not perform without external *stimuli*
- Two routes to “persuasion”
  - Central route
    - *Stimuli* are weighted by the subject and final decision is carefully elaborated
    - High amount of cognitive effort
      - Associated with “rational perfectly informed decisions” in economics
    - Persuasion happens through careful elaboration of information
  - Peripheral route
    - Communication that typically does not result in careful cognitive effort in understanding the message
    - Subject is convinced by under-analyzing apparently relevant “cues” that are in reality unrelated to the subject matter
    - Persuasion happens through ”adjunct elements” to the communication
      - Likeability of subject, physical attractiveness, trust, …

Uses of the peripheral route

- Vastly used as a “cheap” route to convince people to perform an action
  - Buy a product
  - Subscribe to a service
  - Visit a location
  - …
- Especially effective when physical contact is not a factor
- Marketing strategies often rely on this mechanisms
  - TV ad must convince you to buy a shampoo in 30 seconds
- Social engineering differs from marketing in that attacks typically do not try to sell products
  - Rather, social engineers must *persuade* victims to disclose sensitive or private information

Hacking a human

- Six factors affect likelihood of human persuasion

1. Reciprocation
   - Subjects form implied or explicit obligations towards each other → **Normative commitment**
2. Consistency
   - Subjects tend to be consistent with previous decisions, even if all evidence shows that these were *bad* decisions → **Continuance commitment**
3. Social proof
   - Subjects tend to act similarly to their peers to "fit in" → **Affective commitment**
4. Likeability
   - Subjects tend to **trust** people they like, find convincing, or attractive
5. Authority
   - Subjects **fear** punishment (that an authority can impose) and will comply
6. Scarcity
   - Subjects will **react** quickly and possibly irrationally to stimuli when they believe that their freedom of choice is a function of time or resource availability

Normative commitment

- Subjects will perform an action because that's customary or mandated by law or contract
- Based on the notion of reciprocation of benefits
  - When subjects receives something they value, they feel "**cognitive dissonance**"
    - Essentially a "bug" of human psychology
    - Faced when subject must elaborate two contrasting forces or inputs simultaneously
      - Subject must elaborate evidence in contrast to his previous beliefs
      - E.g. "I do not need sun cream" → "here is a tester for you" → "thank you I should probably buy some"
- Promises count as "something of value"
  - I promise you a valuable good at the sole cost of shipping
- People tend to comply because they feel "gratitude" for the unsolicited proposal

Continuance commitment

- Subjects tend to maintain congruence in their attitudes and decisions even in presence of evidence that these are *bad*
  - Subjects tend to maintain **cognitive consonance** as opposed to face cognitive dissonance
- In economics this is reflected in the concept of “loss aversion and sunk costs”
  - If an initial investment was bad, people will tend to keep on investing because they are convinced it will eventually pay-off
    - Pay (small) escalating costs to win a teddy-bear
- Upfront costs are low w.r.t promised benefit vs cost of taking precautions (or opportunity costs)
  - People are willing to give away personal information for negligible benefits or discounts (even if they claim they are willing to pay a premium to preserve their privacy) [Acquisti 2003]

Affective commitment

- People are influenced by the opinion of those they esteem or like
- Decision of action taken to be part of a clique or a circle of peers
  - Widely used for marketing too
- Emotional bond with interlocutor can be exploited to have the victim communicate personal details or perform certain actions
  - e.g. pretend you are on a vacation with a friend of the victim and ask money to solve an emergency
    - Social networks make these inferences possible for the attacker

Liking and Trust

- Similarly to affective commitment, people are willing to be liked by those whom they like
  - Take action to obtain consent from those they like
- People tend to extend “credibility” of subjects they perceive as successful beyond the reasonable boundaries of these subjects’ actual expertise
  - e.g. famous actor that publicizes biscuits despite having no actual expertise or credibility as a baker, but only as an actor
- When physical/presence attraction is not a factor (e.g. email exchange), the likeability can emerge from a “friendly connection”
  - e.g. appeal or elicit common traits

Authority

- People tend to respond to authority especially when in fear of the outcomes of *not taking action*
  - E.g. Punishment or the cancellation of a privilege
    - *"Your email account is going to be deleted if your password is not confirmed."*
- Obedience to authority is a very powerful tool to persuade people in pertaining actions or behaviors
- In some (occasionally very controversial) cases people will obey to authority even against well-established moral values and ethics

Scarcity

- Similarly to fear, scarcity leads people to take quick, potentially uninformed decisions in fear of losing an opportunity that will either disappear in time or that is scarce in quantity
- Can be used by social engineers to elicit unwise decisions from the victims
  - Threaten that if no decision is taken quickly, the opportunity may fade away
  - Attackers poses a “constraint” in the freedom of choice of the victim

Reading List

- Arora, Ashish, et al. "Impact of vulnerability disclosure and patch availability-an empirical analysis." *Third Workshop on the Economics of Information Security*. Vol. 24. 2004.
- Miller, Charlie. "The legitimate vulnerability market: Inside the secretive world of 0-day exploit sales." *In Sixth Workshop on the Economics of Information Security*. 2007.
- http://phrack.org/issues/49/14.html
- OWASP resources
- Moore, Tyler, and Richard Clayton. "An Empirical Analysis of the Current State of Phishing Attack and Defence." *WEIS*. 2007.
- Workman, Michael. "Wisecrackers: A theory-grounded investigation of phishing and pretext social engineering threats to information security." *Journal of the American Society for Information Science and Technology* 59.4 (2008): 662-674.
- Acquisti, Alessandro, and Jens Grossklags. "Losses, gains, and hyperbolic discounting: An experimental approach to information security attitudes and behavior." *2nd Annual Workshop on*
- *Economics and Information Security-WEIS*. Vol. 3. 2003.

- Michael Workman, “Gaining Access with Social Engineering: An Empirical Study of the Threat”, *Journal of Information Systems Security, 16:315–331, 2007*
- James E. Grunig, A situational theory of publics: Conceptual history, recent challenges, and new research. 1997