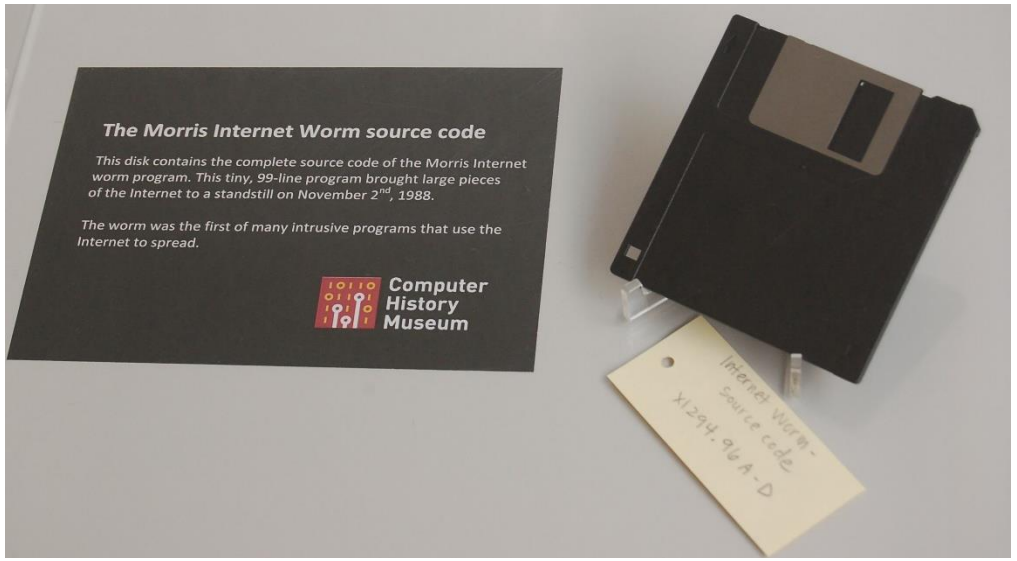


Introduction to software security

ISEL – Instituto Superior de Engenharia de Lisboa
Rua Conselheiro Emídio Navarro, 1 | 1959-007 Lisboa

Conceitos base de segurança

- Confidentiality
  - Absence of unauthorized disclosure of information
  - Guaranteed by cryptographic or access control means
- Integrity
  - Absence of unauthorized changes to the system or information
  - Verified by cryptographic or access control means
- The security policy determines what is allowed and what is not

- Availability
  - System readiness to provide the service or make information available

Vulnerabilities

Classification

- Project
  - Vulnerability during the requirements definition and architecture design phase.
  - E.g.: Not taking into account all the scenarios where communication can be observed
- Codification
  - Code error (bug) with security implications
  - E.g.: insufficient input validation
- Operational
  - Vulnerability caused by configuration error or runtime environment
  - E.g.: accounts without passwords

Vulnerablitites

Ataques e correções

- An attack is a malicious action that activates one or more vulnerabilities.
- A successful attack using exploit software results in an intrusion
  - Attack + Vulnerability -> Intrusion
- Detecting and fixing errors is part of the development cycle
- Some studies report 15 to 50 errors per 1000 lines of code
- The publication and correction of vulnerabilities sometimes leads to the construction of new exploits
- Vulnerabilities that are unknown to the software company or the general public are designated as zero-day (0-day)

Vulnerablitites

Classes of vulnerabilities

- *Common Weakness Enumeration (CWE) is a list of vulnerability classes*
  - *CWE-NNN format, where NNN is the number assigned to the class*
- *CWE 2021 Top10*

Rank	ID	Name	Score	2020 Rank Change
[1]	CWE-787	Out-of-bounds Write	65.93	+1
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	46.84	-1
[3]	CWE-125	Out-of-bounds Read	24.9	+1
[4]	CWE-20	Improper Input Validation	20.47	-1
[5]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	19.55	+5
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	19.54	0
[7]	CWE-416	Use After Free	16.83	+1
[8]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.69	+4
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	14.46	0
[10]	CWE-434	Unrestricted Upload of File with Dangerous Type	8.45	+5

https://cwe.mitre.org/top25/archive/2021/2021_cwe_top25.html

Vulnerablitites

Vulnerable software/libraries catalog

- *Common Vulnerabilities Exposures (CVE) is a catalog of vulnerabilities in commercial or open software*
  - *CVE-YYYY-NNNN format, where YYYY is the year in which it was cataloged and NNNN is the assigned number*
- *Most are design or coding vulnerabilities.*
- *Vulnerabilities are reported by a CVE Numbering Authority (CNA)*
  - *Researchers, companies, emergency response centers (CERT), …*
  - *Currently 102 CNAs distributed across 17 countries*
- *Example*: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-5789
  - *An integer overflow that leads to a use-after-free in WebMIDI in Google Chrome on Windows prior to 73.0.3683.75 allowed a remote attacker who had compromised the renderer process to execute arbitrary code via a crafted HTML page.*

Vulnerabilidades

Grau de gravidade

Common Vulnerability Scoring System (CVSS)

- Attack vector (network, adjacent network, local, physical access)
- Attack complexity (high, low)
- Required privileges (high, low, none)
- User interaction (none, required)
- Scope (same, other)
- Impacts (confidentiality, integrity, availability)
- Exploitability (code not available, proof of concept, functional)
- Remediation Level (Full Solution, Interim Fix, Unofficial)
- https://www.first.org/cvss/calculator/3.0

Attacks

Attack surface

- The interface through which a system can be compromised is called the attack surface.
  - technical attacks
  - Network
  - Application server, execution environments
  - Operating system
  - Hardware
- social engineering attacks
  - Users

Attacks

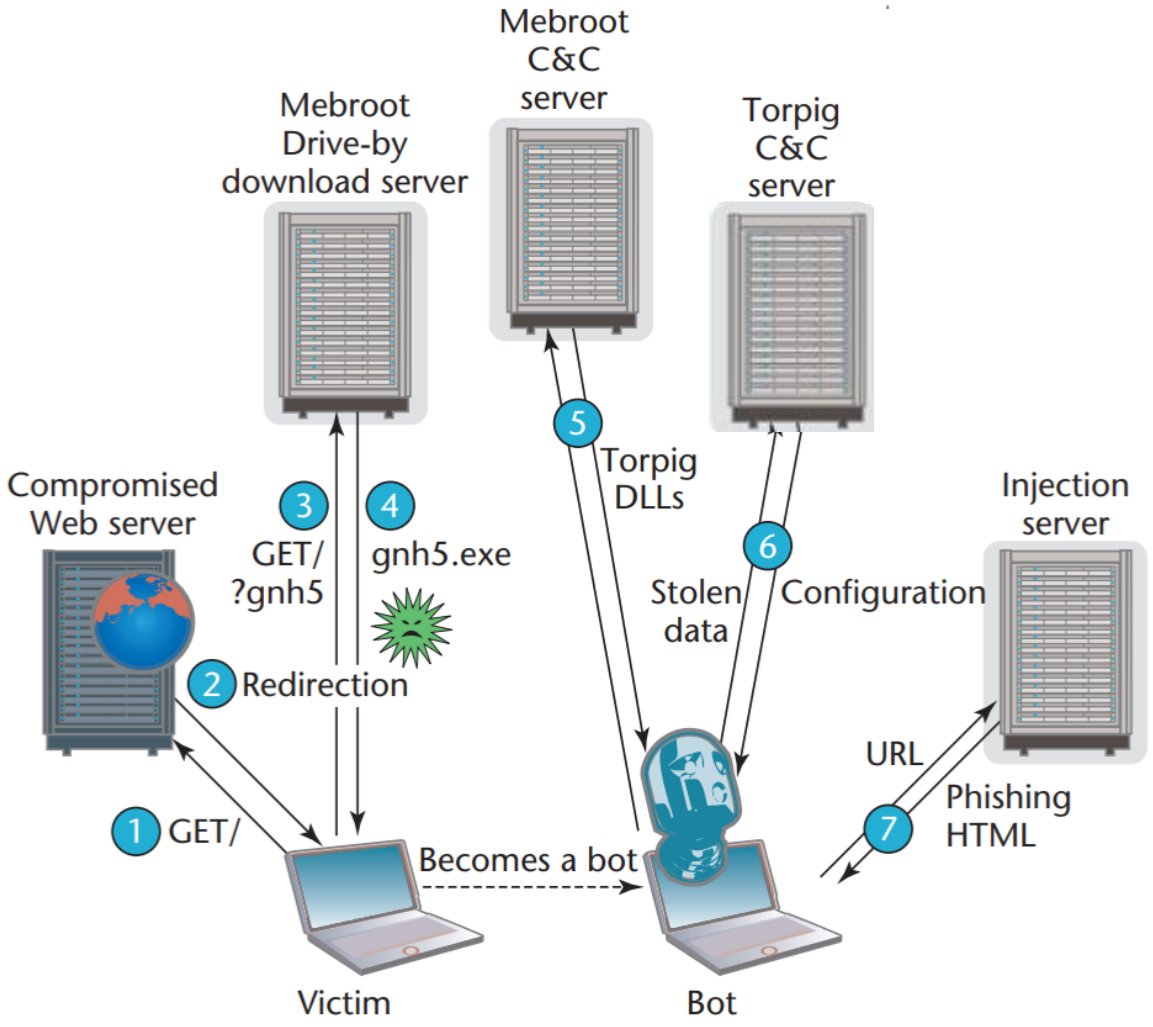
- Malicious programs (https://www.virustotal.com/)
  - Malware/Viruses; Worm/Worm; Trojan Horse
- These programs have different attack strategies…
  - Bot: program to obtain data or execute commands from the outside (backdoor)
  - Botnet: botnet
  - Command and Control Server: botnet remote control
- … and different goals
  - Identity theft or confidential data
  - Cryptocurrencies
    - Ransomware: bot that waits for command to encrypt all computer data, delivering key against payment of cryptocurrencies
    - Cryptominers: bot that consumes CPU cycles by mining cryptocurrencies to deliver to attacker

Lockheed Martin *kill chain*

Ordem	Fase	Descrição
1	*Reconnaissance*	Search, identify and select targets.
2	*Weaponization*	Linking the malware with the payload that allows it to affect the target system. Ex: Put the attack code in PDF or Word files.
3	*Delivery*	Transmission from the “weapon” to the target (Ex: Email attachments, USB stick, websites visited by targets)
4	*Exploitation*	Once delivered, the "weapon" is activated by executing the attacker's code, exploiting the attack surface (operating system, applications, user, …)
5	*Installation*	The "weapon" installs a backdoor on the target system, allowing permanent access to it
6	*Command & Control*	From outside the organization, there is access to systems within the target network
7	*Actions on Objective*	The attacker tries to achieve its objectives, which may include theft and destruction of data or intrusion into other targets

Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains, Lockheed Martin Corporation, 2010
https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf

Example: Torpig



A shaded components controlled by attackers:

1: attackers compromise vulnerable websites

2 and 3: modified pages redirect the victim's browser to a drive-by download

4: the victim downloads and runs Mebroot, becoming a bot

5: the bot gets the Torpig modules

6: the bot uploads stolen data from the victim's computer

7: when the victim visits certain websites, and selected pages, the bot gets phishing pages that it presents to the user (ex: *login*)

https://sites.cs.ucsb.edu/~vigna/publications/2011_SPMagazine_torpig.pdf

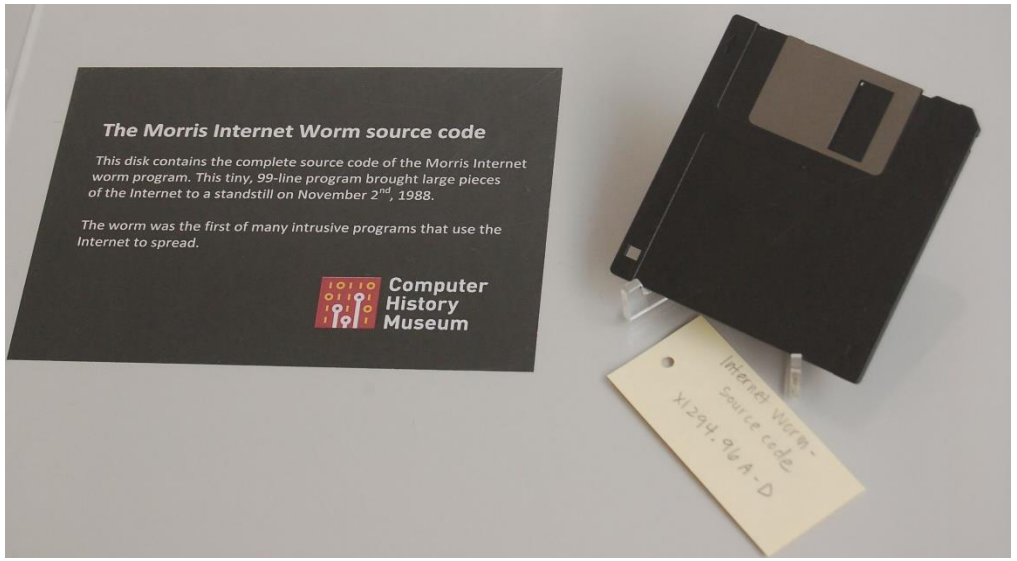
ENISA – Threat report 2021

https://www.enisa.europa.eu/publications/enisa-threat-landscape-2021

Social engineering

Phishing, identity theft, data leakage

- Refers to all techniques designed to speak to a target to reveal specific information or perform a specific illegitimate action.
- Most common techniques
  - Pretexting
    - Obtaining credentials using a false justification but the target finds it credible
  - Baiting
    - Provide the victim with relevant “services” that end up stealing data or affecting the system
  - Quid pro quo
    - Attacker asks for sensitive information offering a reward
  - Tailgating
    - Have physical access to protected places accompanying an authorized person



Risk assessement and Software development

ISEL – Instituto Superior de Engenharia de Lisboa
Rua Conselheiro Emídio Navarro, 1 | 1959-007 Lisboa

Risk assessment

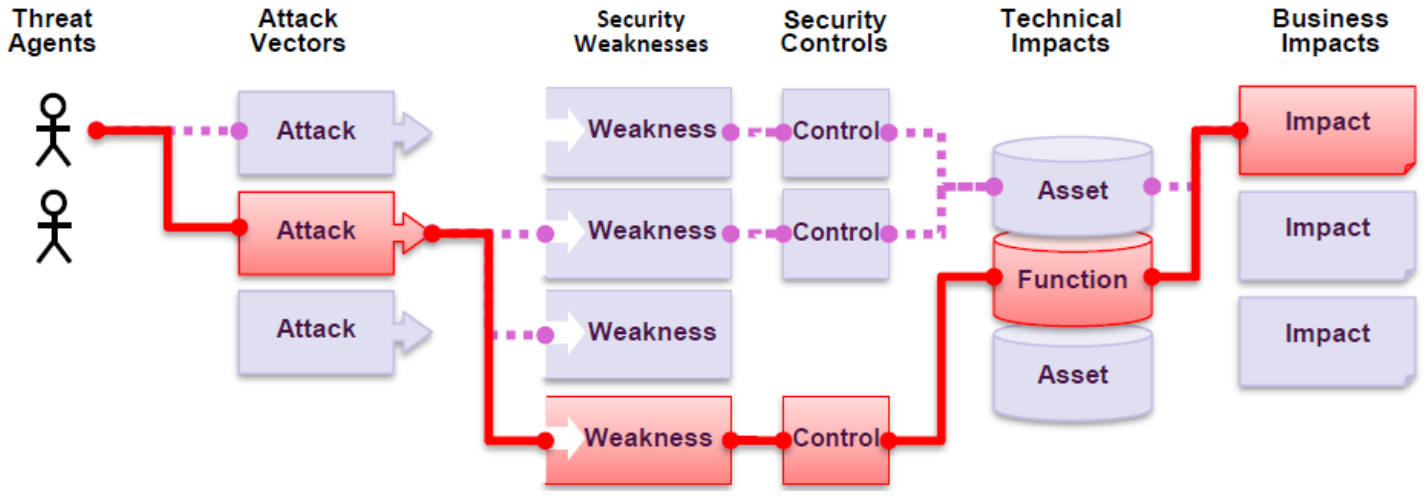
- When planning software, take into account
  - Functionality, Usability, Performance, Simplicity, Performance, Low time-to-market
- Aiming to develop a system without any vulnerability is unrealistic
- It is necessary to assess the **risk**
  - Risk = probability x impact
- Likelihood to exploit the risk
  - Exposure of affected system, type of use
  - Degree of Vulnerability: Design, Code, or Configuration Errors
- Impact
  - Impact on information security properties: confidentiality, integrity, availability
  - Impact on the organization's reputation

OWAP Risk Management Methodology

- Regulatory frameworks such as ISO 27001 or the Open Web Application Security Project (OWASP) propose the following methodology for risk classification and mitigation

1. Identify the situation, stakeholders and assets involved
2. Factors for estimating the probability of exploiting risk
3. Factors to estimate the impact
4. Determine the severity of the risk
5. Decide what to fix
6. Adapt the risk rating model

OWAP Risk Management Methodology



- Agents can take multiple paths to harm the system.
  - Different levels of motivation, degrees of opportunity, number of attackers
- The different paths can be easier or harder
  - Ease of discovery and exploration, ability to detect intruders
- Each attack vector has a different impact on the business.
  - Loss: confidentiality/integrity, personal data, financial, reputation

https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology#Step_1:_Identifying_a_Risk

https://www.portaldodpo.pt/blog/service/risco/

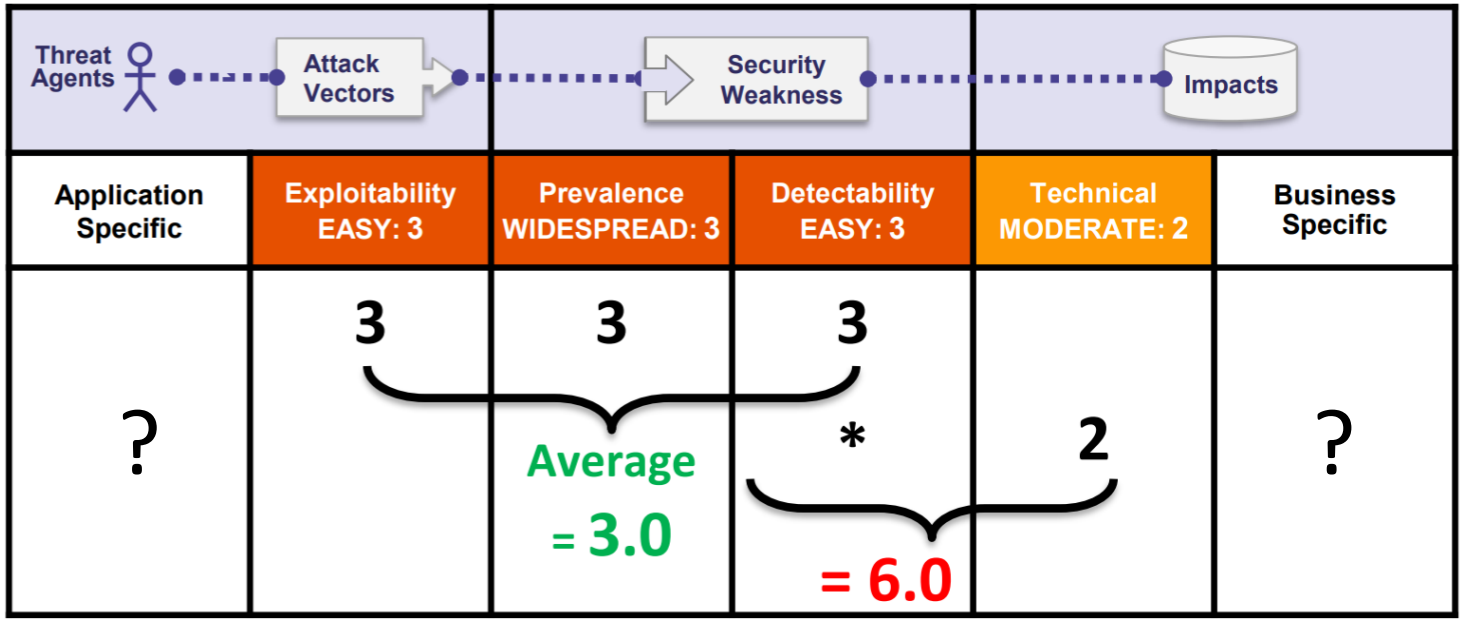
OWAP Risk Management Methodology

Threat Agents	Exploitability	Weakness Prevalence	Weakness Detectability	Technical Impacts	Business Impacts
Appli-cation Specific	Easy: 3	Widespread: 3	Easy: 3	Severe: 3	Business Specific
	Average: 2	Common: 2	Average: 2	Moderate: 2	
	Difficult: 1	Uncommon: 1	Difficult: 1	Minor: 1	

- Levels for estimating probability and impact
- Higher levels will give rise to a higher level of risk

Example of risk calculation for "Bad security configuration" vulnerability

e.g. the latest available updates are not installed correctly



https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology#Step_1:_Identifying_a_Risk

Less potential for vulnerabilities, less risk

- Programming languages have implications for system security
  - Native languages with manual memory management have higher performance but higher risks
  - High-level languages (Java, C#, Phyton) run in a controlled (managed) environment with regulated access to native resources
  - The Perl language interpreter can check if the input data reaches critical components (tainted mode)
- Closed or open source
  - More robust/professional closed source (proprietary)? Free (open) source made by committed creators?
  - Many eyeballs argument: open source has the greatest potential for scrutiny. But does it happen? Age-old bugs like ShellSock or Heartbleed show that this is not always the case
  - security by obscurity argument: proprietary code is more secure because it is unknown. But the frequency of errors discovered in these systems is high

Avoid the most common design vulnerabilities

https://ieeecs-media.computer.org/media/technical-activities/CYBSI/docs/Top-10-Flaws.pdf

1. Never assume or trust
   - The user interface alone does not prevent access to protected resources
   - Client-side key storage can lead to their discovery
2. Use non-bypassable authentication mechanisms
   - Authenticators cannot be easy to forge
   - Use more than one authentication element
3. Authorize after authenticating
   - Access control based on identity or role in the organization is essential to ensure the confidentiality, integrity and availability of information

Avoid the most common design vulnerabilities

4. Separating data from control instructions
   - Mixing data and control instructions is at the base of the most common vulnerabilities in web and native applications
5. Explicitly Validate All Data
   - It is related to the above.
   - Applications usually make assumptions about inputs, they need to be validated.
6. Correctly use encryption
   - Do not use "homemade" algorithms
   - Key mismanagement
   - Weak randomness sources

Avoid the most common design vulnerabilities

7. Identify sensitive data and how it should be handled
   - Correctly identifying sensitive data is essential for its protection
8. Always consider users
   - Sufficiently expressive but not excessive security controls
9. Understand the impact of exterior components on the attack surface
   - The inevitable use of external components results in inheriting their weaknesses and limitations
   - Validate the provenance and integrity of the external component
10. Design for future changes
   - Safe modification of application parts and keys

Security in development processes

SDL Agile

- Development processes known as Agile (e.g. Extreme Programming, Scrum) are increasingly used
- Organized in sprints (period 2 to 4 weeks) where stories (set of tasks) are developed, stored in the product backlog
- The SAFEcode (Software Assurance Forum for Excellence in Code) consortium defined a set of stories and security tasks
  - Examples of stories to embed: output coding is done correctly; checking of buffer indexes; synchronization on concurrent access to resources
  - Examples of operational tasks: Use the latest compiled versions; apply available patches; carefully review higher risk code
  - Expert tasks: training coding security and testing; perform penetration tests; perform fuzzing tests

Security in development processes

http://safecode.org/wp-content/uploads/2018/01/SAFECode_Agile_Section2a-tables.pdf

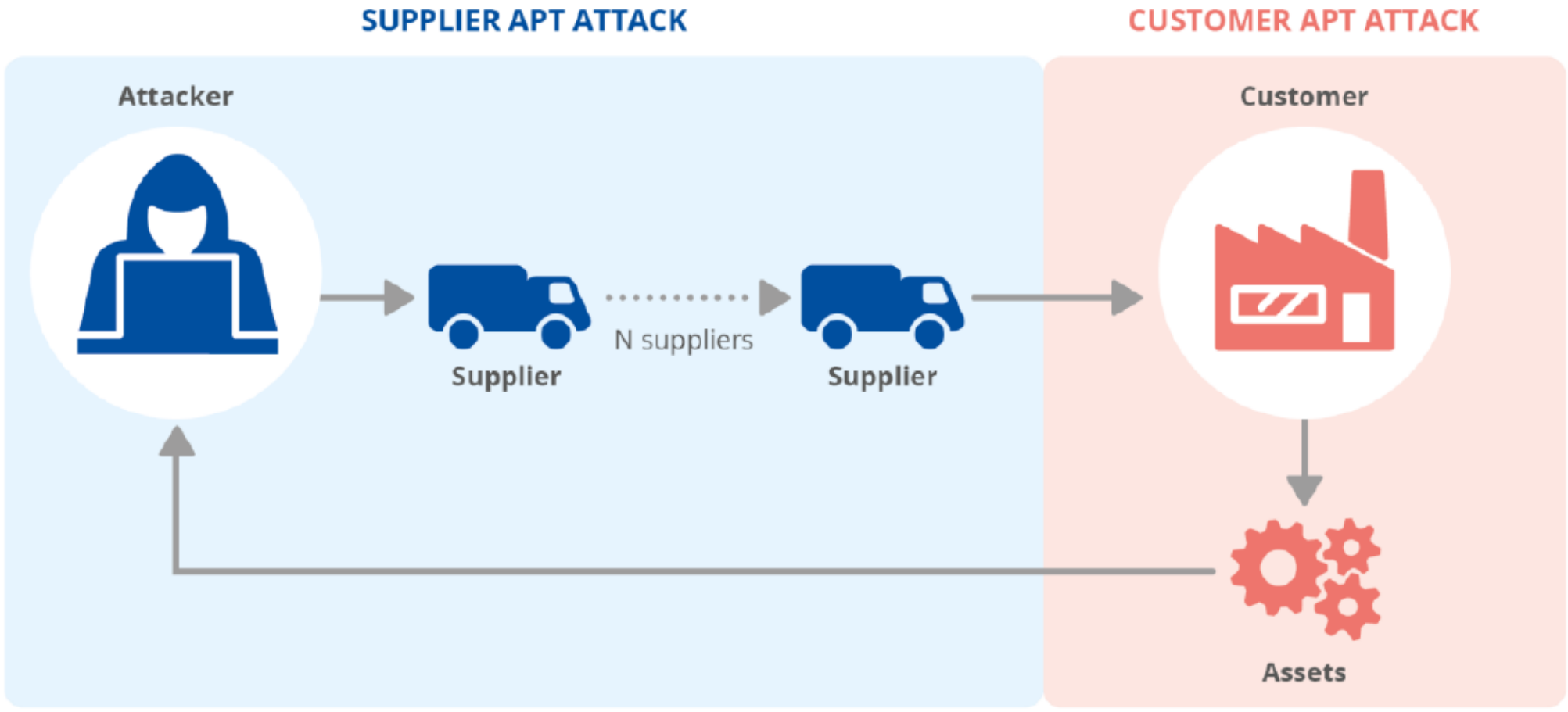
3	As a(n) architect/ developer, I want to ensure **AND** as QA, I want to verify application of or access within index boundaries of buffers and arrays	**[A/D]** Define where buffer operations (on dynamic buffers) occur. Define data types and bounds for buffer operations. **[D]** Adhere to SAFECode's Fundamental Practices for Secure Software Development for prevention of buffer overflows. **[D]** Scan source code for such violations using static code analyzer tools, e.g., Coverity. **[A/D]** Conduct false positive analysis of flagged issues. **[D]** Fix buffer overflow issues analyzed as confirmed. **[T]** Use fuzz testing tools to verify that no process/system crashes/hangs exist. If they do, fix them and re-run the tool.	• Minimize Use of Unsafe String and Buffer Functions • Use a Current Compiler Toolset • Use Static Analysis Tools	CWE-120 CWE-131 CWE-805

SAFECode guidelines

- Comprehensive set of design and coding recommendations (link)
- Architecture
  - Principles seen above; Threat Modeling
  - Strategy for encrypting information (eg key management)
  - Use standard authentication mechanisms/protocols; Establish policies for logs
- Code
  - Use coding conventions and best practices (OWASP - Secure Coding Practices, Secure Coding Guidelines for Java SE)
  - Use secure functions
  - Use tools to detect security issues early in development
  - Validate inputs; Handle errors, giving only detailed information in internal logs

Security in the software supply chain

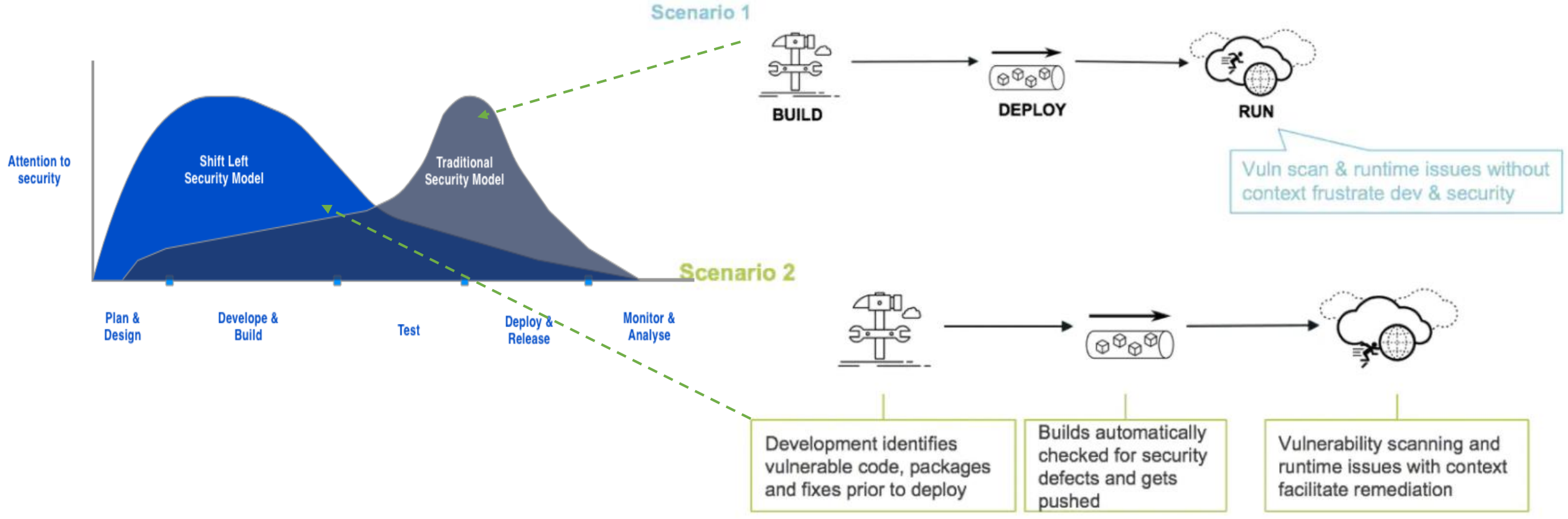
- In cybersecurity, the supply chain involves a wide range of resources (hardware and software), storage (cloud or local), distribution mechanisms (web applications, online stores), wand management software.



https://www.enisa.europa.eu/publications/threat-landscape-for-supply-chain-attacks

Summary – *Shift Left*

- Introduce security checks as early as possible in the development cycle
  - Look for typical bugs in the code; analyze sensitive data streams; look for vulnerabilities in dependencies; dynamically test injected failures;



https://www.paloaltonetworks.com/blog/2019/07/4-practical-steps-shift-left-security/
https://devopedia.org/shift-left