NETWORK & MULTIMEDIA LAB

SECURE SOFTWARE DEVELOPMENT LIFE CYCLE

Fall 2020

Introduction

- 軟體開發生命周期（Software Development Life Cycle, SDLC）
- 安全軟體開發生命周期（Secure Software Development Life Cycle, SSDLC）
  - Introduces security and privacy considerations throughout all phases of the development process

SDLC（Software Development Life Cycle）

需求 Requirements	設計 Design	開發 Implementation	測試 Testing	部署 Deployment
需求訪談、需求分析、研擬可行性方案與排定優先順序	軟體架構、邏輯設計及細部設計	程式開發	驗證程式開發結果符合需求	系統上線與後續維護
• 需求訪談會議 • 建置 Prototype • 需求建議書 • Use Case • …	• 結構化分析設計 • 物件導向分析設計 • …	• 函式 • 程序 • 模組 • 物件 • …	• 單元測試 • 整合測試 • 系統測試 • 驗收測試 • 功能測試 • 效能測試 • 壓力測試 • …	• 硬體安裝 • 系統安裝 • 環境設置 • 教育訓練 • 問題分析與處理 • 變更管理 • 系統轉移 • …

Security principles

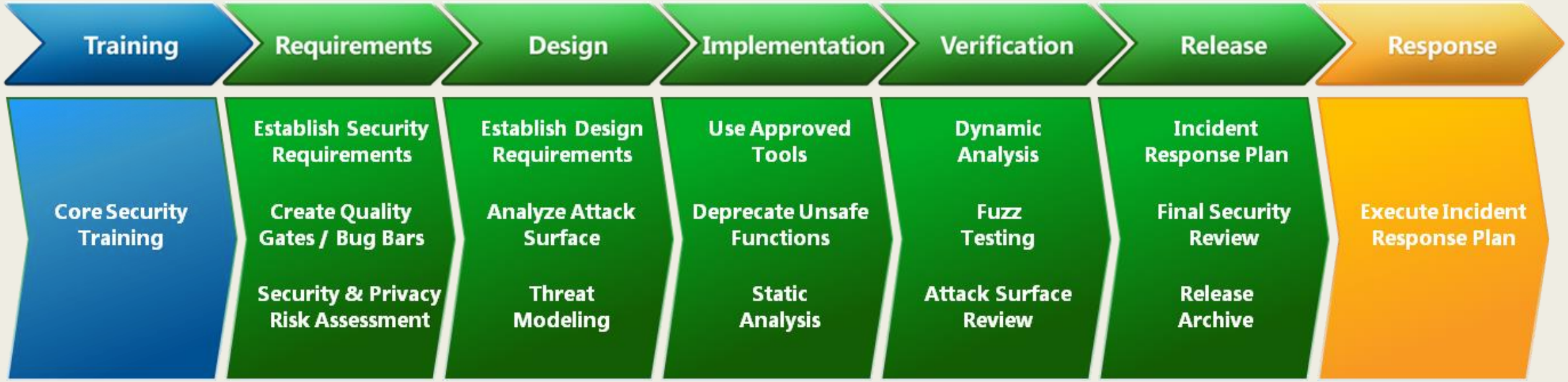
- Minimize attack surface
- Establish secure defaults
- Principle of Least privilege
- Principle of Defense in depth
- Fail securely
- Don’t trust services
- Avoid security by obscurity
- Keep security simple

SSDLC
Popular Frameworks and Standards

- Classical SDL
  - Microsoft Security Development Lifecycle (MS SDL)
  - OWASP Secure Software Development Lifecycle
- Compliance Standards
  - ISO/IEC 27034 (安全軟體開發國際標準)
  - ISO/IEC 15408 (資訊技術安全評估共同準則， Common Criteria)
- Maturity Frameworks
  - Building Security In Maturity Model (BSIMM)
  - Software Security Assessment Maturity Model (SAMM)

Microsoft SDL

- A waterfall framework



BSIMM

- 4 domains
- 12 practices
- https://www.bsimm.com/

The Software Security Framework (SSF)			
Governance	**Intelligence**	**SSDL Touchpoints**	**Deployment**
Strategy and Metrics	Attack Models	Architecture Analysis	Penetration Testing
Compliance and Policy	Security Features and Design	Code Review	Software Environment
Training	Standards and Requirements	Security Testing	Configuration Management and Vulnerability Manage-ment

BSIMM practices are collected from real world implementations

STANDARDS & REQUIREMENTS (SR)			
ACTIVITY DESCRIPTION	ACTIVITY	OBSERVATIONS	PARTICIPANT %
LEVEL 1			
Create security standards.	[SR1.1]	93	71.5%
Create a security portal.	[SR1.2]	90	69.2%
Translate compliance constraints to requirements.	[SR1.3]	94	72.3%
LEVEL 2			
Create a standards review board.	[SR2.2]	64	49.2%
Identify open source.	[SR2.4]	60	46.2%
Create SLA boilerplate.	[SR2.5]	44	33.8%
LEVEL 3			
Control open source risk.	[SR3.1]	30	23.1%
Communicate standards to vendors.	[SR3.2]	9	6.9%
Use secure coding standards.	[SR3.3]	9	6.9%
Create standards for technology stacks.	[SR3.4]	25	19.2%

BSIMM practices are collected from real world implementations

CODE REVIEW (CR)			
ACTIVITY DESCRIPTION	ACTIVITY	OBSERVATIONS	PARTICIPANT %
LEVEL 1			
Perform opportunistic code review.	[CR1.2]	79	60.8%
Use automated tools along with manual review.	[CR1.4]	100	76.9%
Make code review mandatory for all projects.	[CR1.5]	49	37.7%
Use centralized reporting to close the knowledge loop and drive training.	[CR1.6]	41	31.5%
Assign tool mentors.	[CR1.7]	50	38.5%
LEVEL 2			
Use automated tools with tailored rules.	[CR2.6]	23	17.7%
Use a top N bugs list (real data preferred).	[CR2.7]	24	18.5%
LEVEL 3			
Build a capability to combine assessment results.	[CR3.2]	7	5.4%
Create a capability to eradicate bugs.	[CR3.3]	3	2.3%
Automate malicious code detection.	[CR3.4]	2	1.5%
Enforce coding standards.	[CR3.5]	1	0.8%

BSIMM practices are collected from real world implementations

SOFTWARE ENVIRONMENT (SE)			
ACTIVITY DESCRIPTION	ACTIVITY	OBSERVATIONS	PARTICIPANT %
LEVEL 1			
Use application input monitoring.	[SE1.1]	73	56.2%
Ensure host and network security basics are in place.	[SE1.2]	121	93.1%
LEVEL 2			
Define secure deployment parameters and configurations.	[SE2.2]	39	30%
Protect code integrity.	[SE2.4]	33	25.4%
Use application containers.	[SE2.5]	31	23.8%
Ensure cloud security basics.	[SE2.6]	36	27.7%
LEVEL 3			
Use code protection.	[SE3.2]	13	10.0%
Use application behavior monitoring and diagnostics.	[SE3.3]	7	5.4%
Use orchestration for containers and virtualized environments.	[SE3.5]	22	16.9%
Enhance application inventory with operations bill of materials.	[SE3.6]	12	9.2%

Security Design

Designs

- Functional design
- Design for performance and scalability
- Design for manageability
- Security design

What Is Threat Modeling?

- Identifying security threats during the design phase
- The output of Threat Modeling is a list of high priority threats
  - Some or all of them will be mitigated by a set of security designs
- Threat modeling is the key to a **focused defense**

Reasons of Threat Modeling

- Find Security Bugs Early
  - At the design time, not after code complete
- Understand Your Security Requirements
  - Good threat models can help you ask “Is that really a requirement?”
  - You’ll need to make a call between mitigation with a security design, or accepting the risk
- Helps you address classes or groups of attacks, and deliver a more secure product.
- Address Issues Other Techniques Won’t
  - Threat Modeling should NOT be focused on issues that other security practices/tools are likely to find

There is more than one way to Threat Model

- It is reasonable to expect software professionals to know the basics of threat modeling
  - Such that “obvious vulnerabilities” won’t be easily created

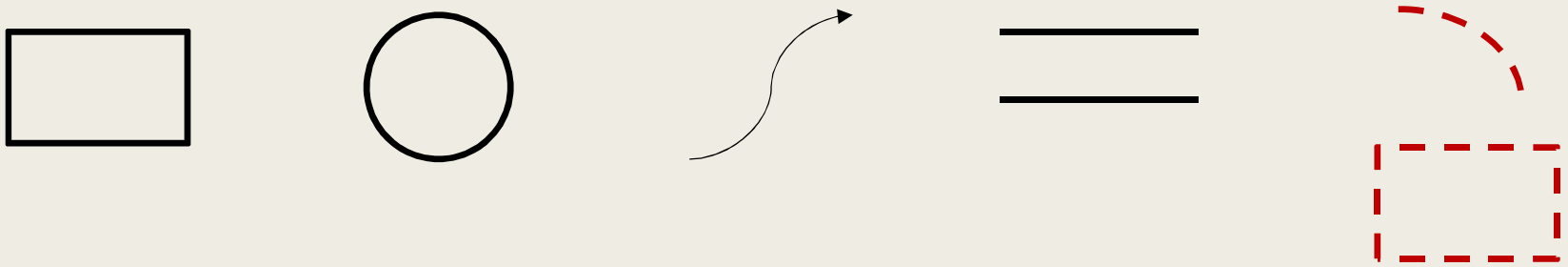
- Examples:
  - DFD (Data Flow Diagram) with a Checklist
  - DFD (Data Flow Diagram) with STRIDE

DFD (Data Flow Diagram)

- Include processes, data stores, data flows, trust boundaries
- Diagram per scenario may be helpful
- Enumerate assumptions, dependencies
- Update diagrams as product changes

Diagram Elements (Examples)

External Entity	Process	Data Flow	Data Store	Trust Boundary
• People • Other Systems	• EXEs • DLLs • Component • Services	• Function call • Network traffic • Remote Procedure Call	• Database • File • Registry • Queue	• Process boundary • Machine boundary • VM • Network



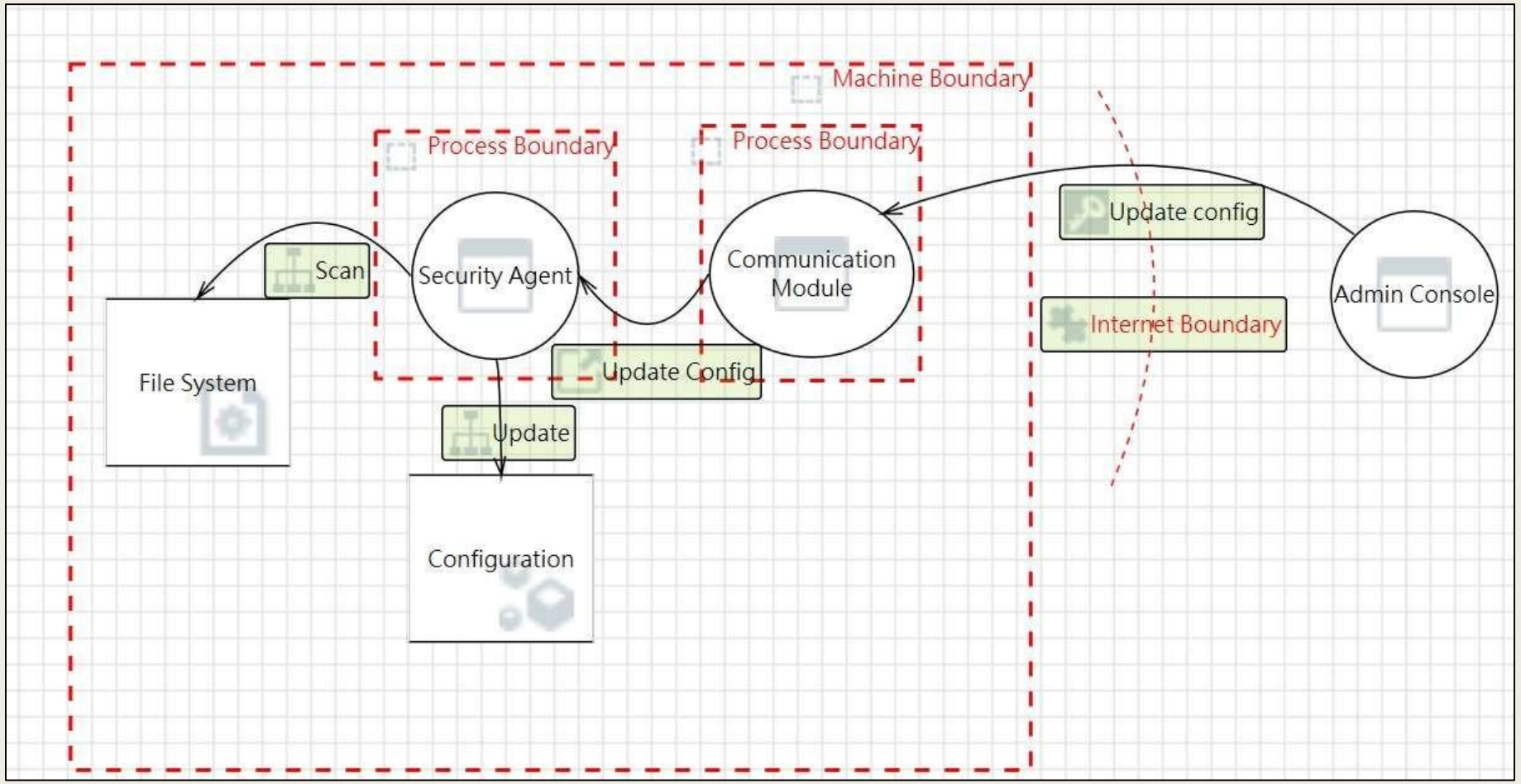
Diagrams: Trust Boundaries

- Add trust boundaries that intersect data flows
- Points/surfaces where an attacker can interject
  - Machine boundaries, privilege boundaries, integrity boundaries are examples of trust boundaries
- Processes talking across a network always have a trust boundary
  - They may create a secure channel, but they're still distinct entities
  - Encrypting network traffic is an 'instinctive' mitigation, but doesn't address tampering or spoofing

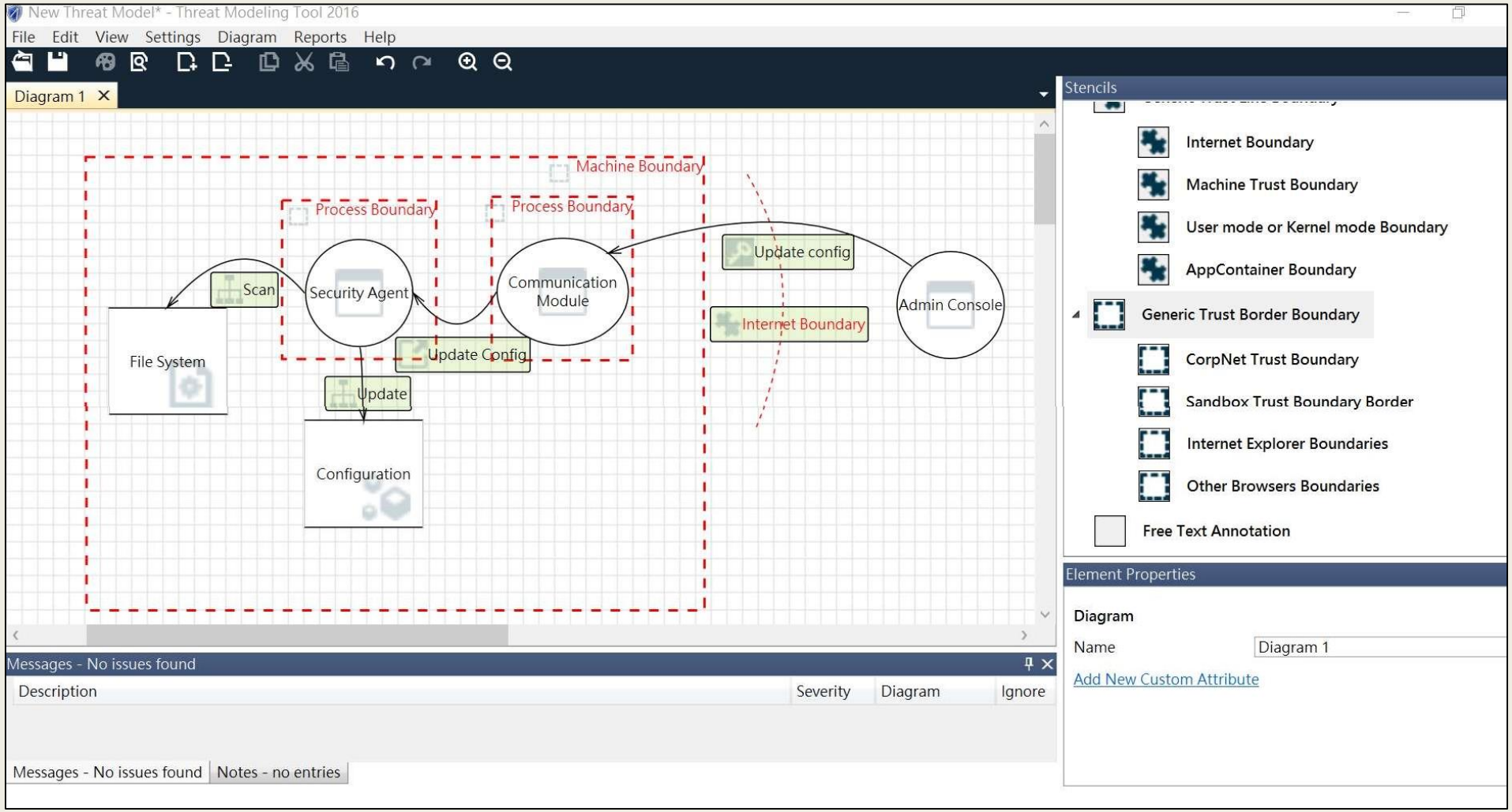
Sensitive Data

- It is advised to identify sensitive data in the diagrams
- Customer privacy data
  - Identifying what privacy data are stored by the product AND where they are stored is critical for GDPR (General Data Protection Regulation，歐盟法規) compliance
- Product specific sensitive data
  - E.g. configuration to control whether a security feature is disabled
  - E.g. AES key, private key
  - E.g. customer ID, user name and password

DFD (Example)


Machine Boundary
Process Boundary
Process Boundary
Update config
Scan
Security Agent
Communication Module
Admin Console
Internet Boundary
File System
Update Config
Update
Configuration

Microsoft Threat Modeling Tool



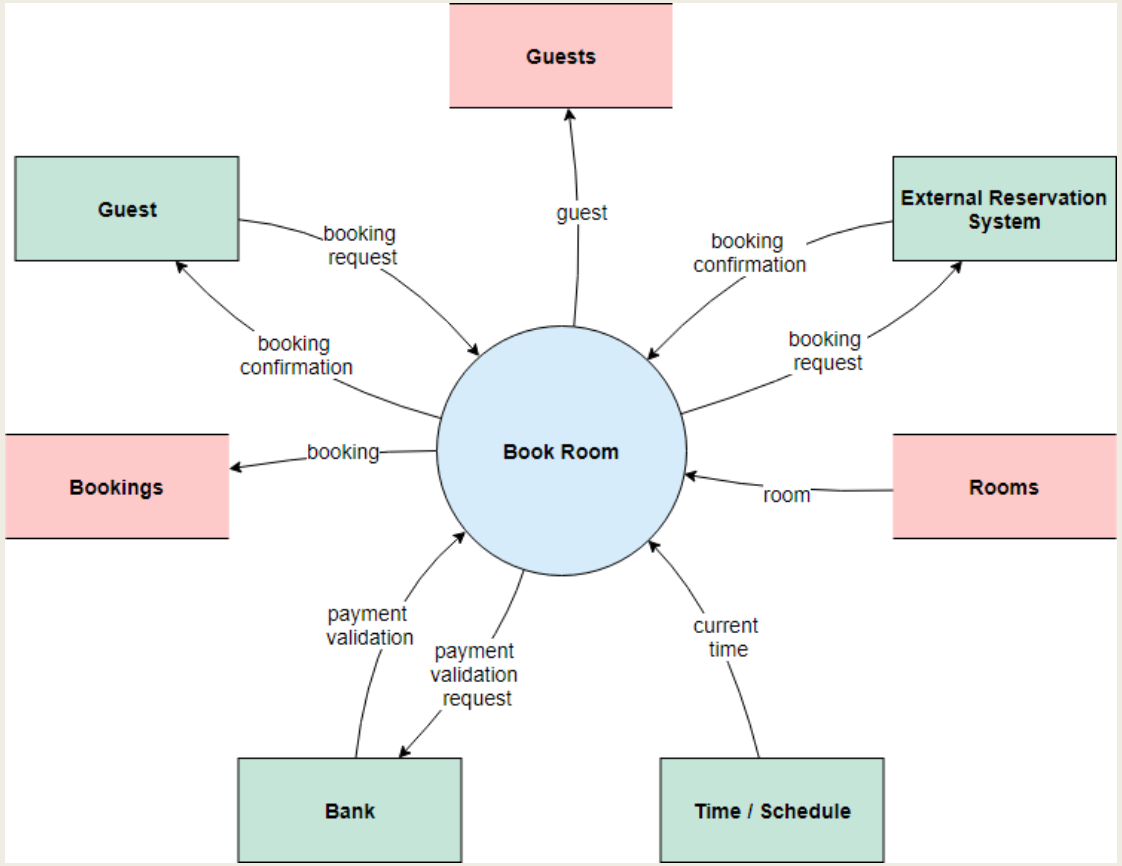
https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling

DFD Layers

- System Context Diagram (Level 0 DFD)
  - Highest level; only one process / product / system
  - It identifies the data flows between the system and external entities.
  - A context diagram is typically included in a requirements document.
- Level 1 DFD
  - High level; single feature / scenario
- Level 2 DFD
  - Low level; detailed sub-components of features
- Level 3 DFD
  - More detailed

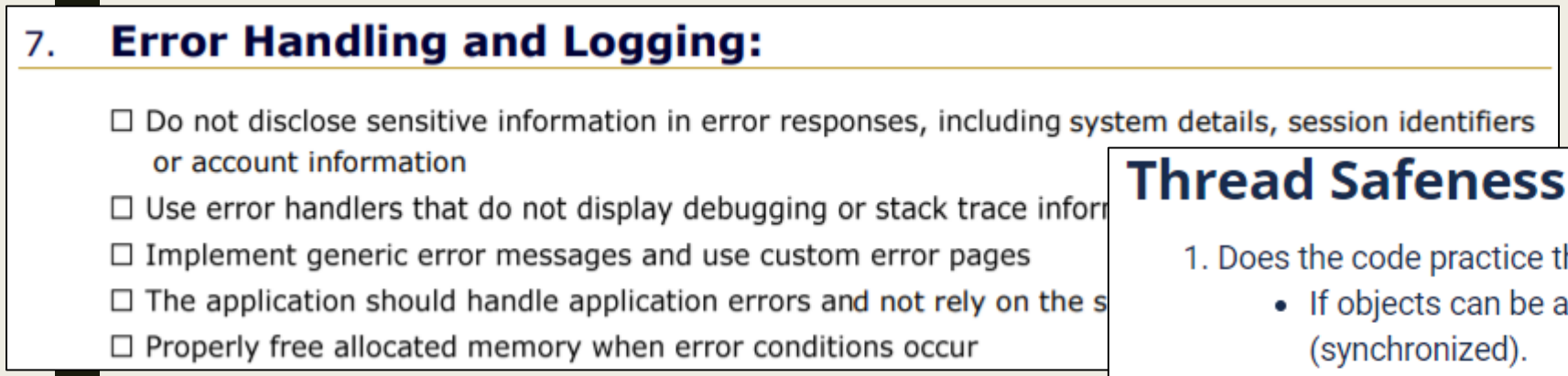
System Context Diagram (Example)

- The entire software system is shown as a single process, with no details of its interior structure, surrounded by all its external entities, interacting systems, and environments.



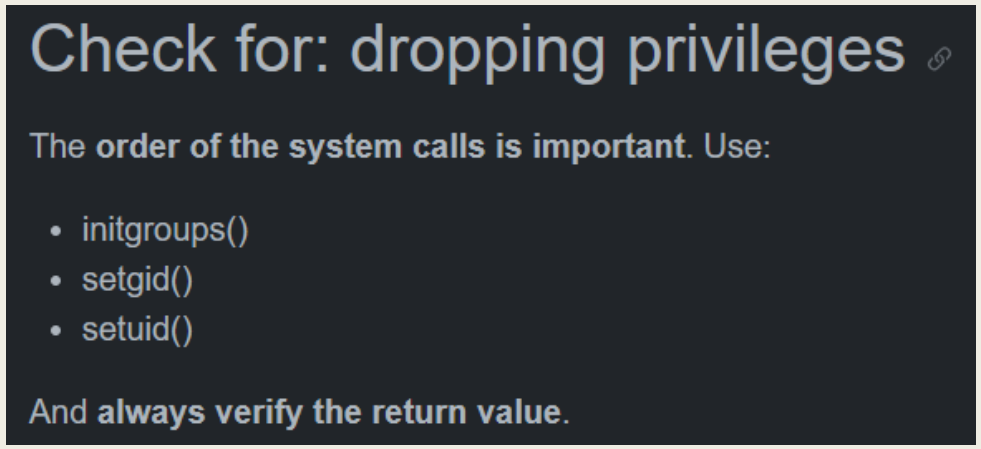
Identifying Threats with a Checklist

- Online resources / Legacy



7. **Error Handling and Logging:**

- ☐ Do not disclose sensitive information in error responses, including system details, session identifiers or account information
- ☐ Use error handlers that do not display debugging or stack trace infor
- ☐ Implement generic error messages and use custom error pages
- ☐ The application should handle application errors and not rely on the s
- ☐ Properly free allocated memory when error conditions occur

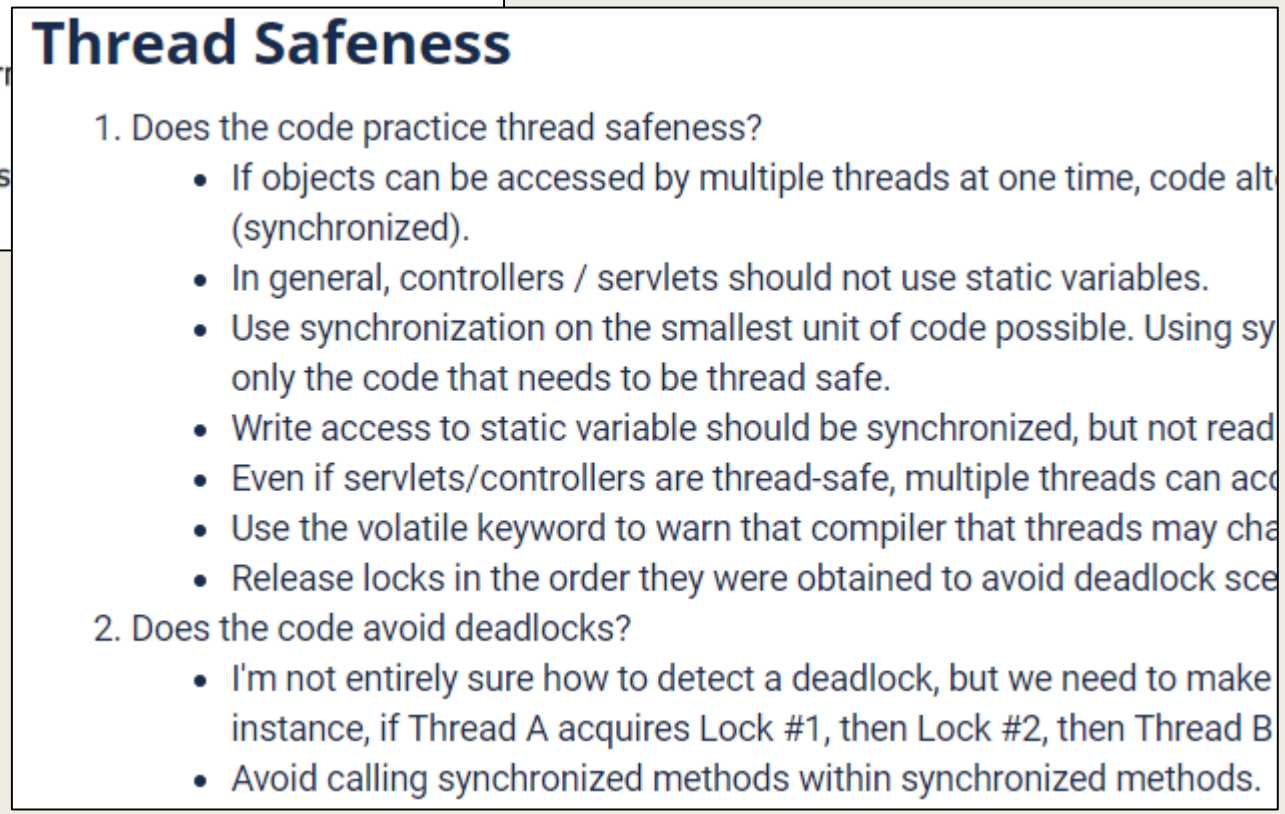


Check for: dropping privileges

The **order of the system calls is important**. Use:

- initgroups()
- setgid()
- setuid()

And **always verify the return value**.



Thread Safeness

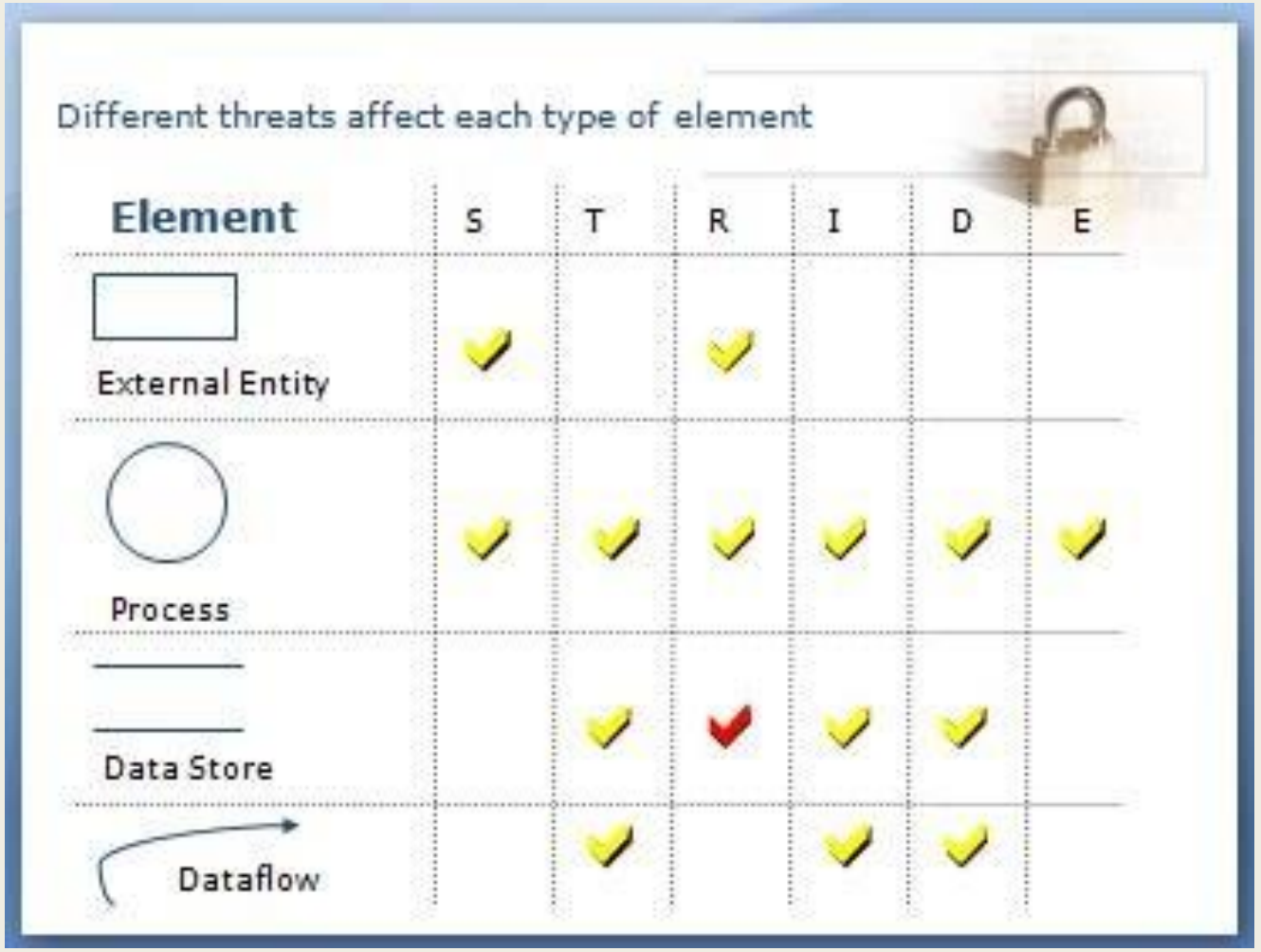
1. Does the code practice thread safeness?
   - If objects can be accessed by multiple threads at one time, code alt (synchronized).
   - In general, controllers / servlets should not use static variables.
   - Use synchronization on the smallest unit of code possible. Using sy only the code that needs to be thread safe.
   - Write access to static variable should be synchronized, but not read
   - Even if servlets/controllers are thread-safe, multiple threads can acc
   - Use the volatile keyword to warn that compiler that threads may cha
   - Release locks in the order they were obtained to avoid deadlock sce
2. Does the code avoid deadlocks?
   - I'm not entirely sure how to detect a deadlock, but we need to make instance, if Thread A acquires Lock #1, then Lock #2, then Thread B
   - Avoid calling synchronized methods within synchronized methods.

Identifying Threats with STRIDE

- STRIDE 從攻擊者的角度，把威脅劃分成 6 個類別

威脅	安全屬性	定義	舉例
偽造（S） Spoofing	認證	冒充人或物	冒充其他用戶帳號 網址偽造偽裝某個的網站
篡改（T） Tampering	完整性	修改資料或代碼	修改訂單資訊、修改軟體所需要的DLL植入惡意程式，或是竄改封包
抵賴否認（R） Repudiation	稽核	不承認做過某行為	不承認修改行為、否認發送過某個Email，或拒絕承認瀏覽過某個網站
資訊洩露（I） Information Disclosure	保密性	資訊被洩露或竊取	使用者資訊被洩露，無論是惡意或是無意，將重要的客戶資料流出，讓未經授權的人使用
阻斷服務（D） Denial of Service	可用性	消耗資源、服務可不用	DDOS 導致網站不可用，系統無法運作，或是網站無法服務
提權（E） Elevation of Privilege	授權	未經授權獲取、提升許可權	普通用戶提升到管理員，讓一個遠端一般使用者，透過非正常的管道，提升權限能夠執行一些只有系統管理員才能使用的指令集

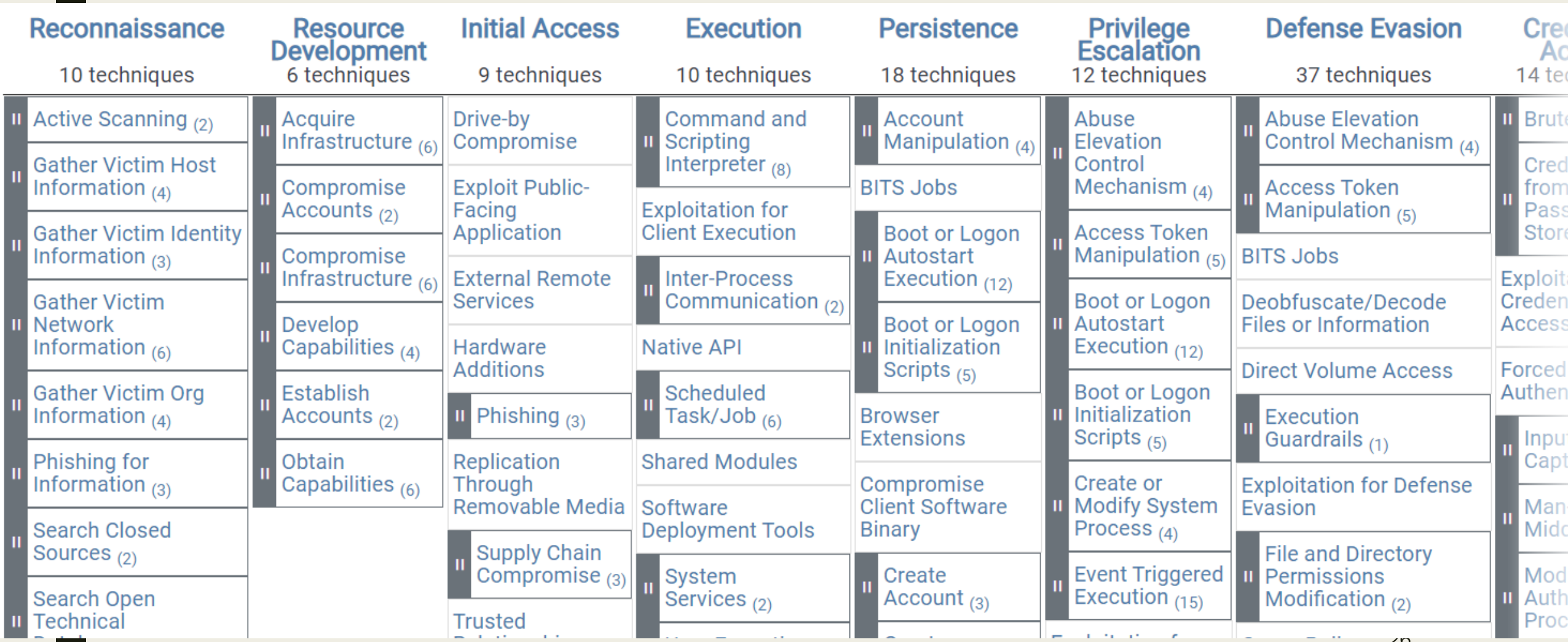
STRIDE per Element



https://webcache.googleusercontent.com/search?q=cache:R9Dh_BM0U2oJ:https://www.microsoft.com/security/blog/2007/10/29/the-stride-per-element-chart/+&cd=1&hl=en&ct=clnk&gl=tw

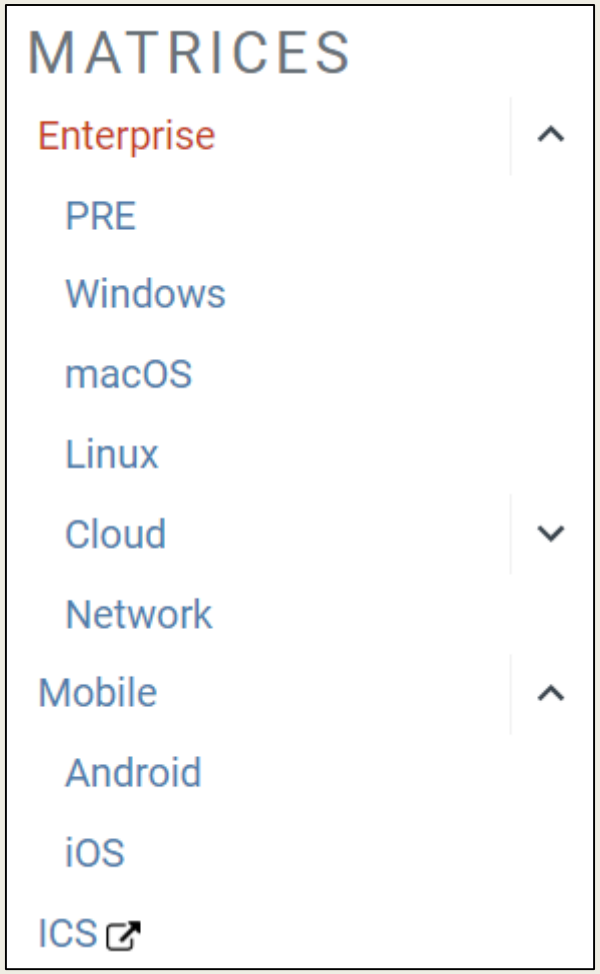
Identifying Threats with MITRE ATT&CK

- A 為 Adversarial，代表攻擊者，兩個 T 是 Tactics 與 Technical，分別代表對手採用的戰略與技術手法，而 C 與 K 則是 Common knowledge



Identifying Threats with MITRE ATT&CK

- 3 Matrices
  - Enterprise
  - Mobile
  - ICS (Industrial Control Systems)

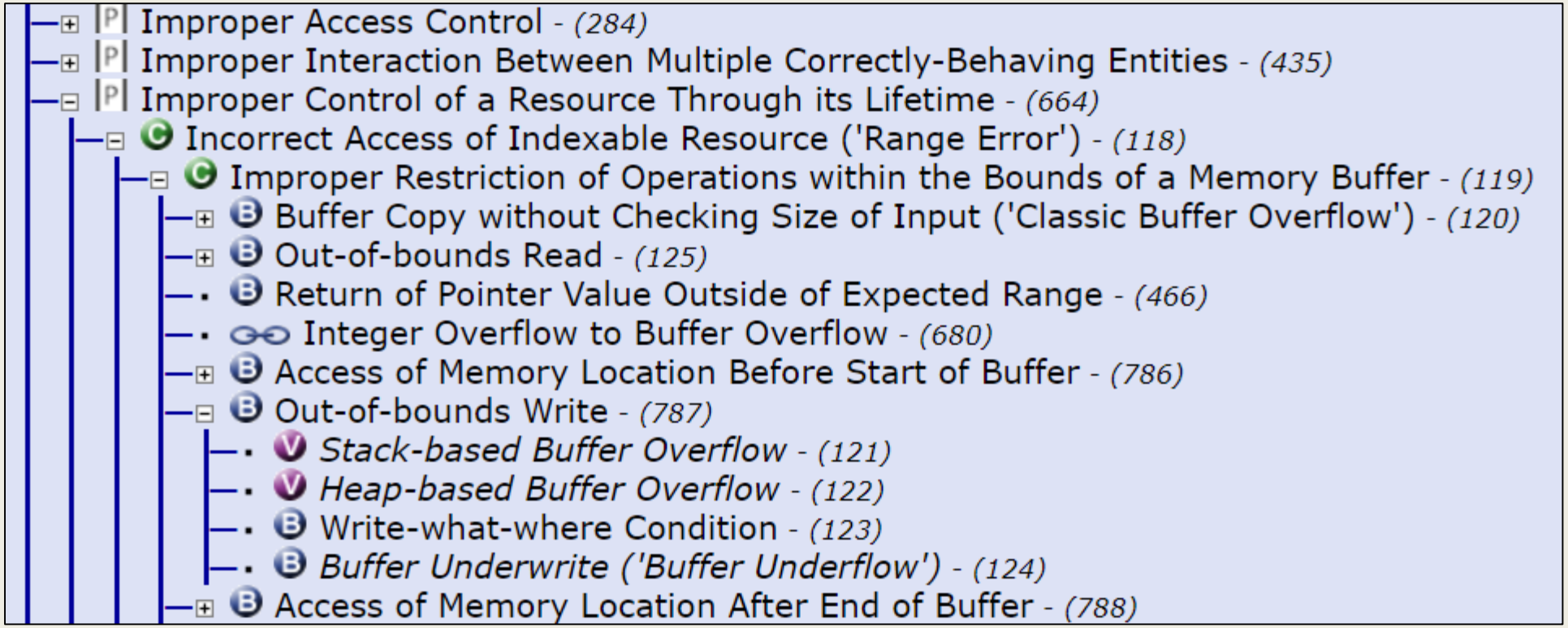


What is Secure Coding Standard?

- Rules and guidelines used to prevent security vulnerabilities
- Especially important for the C/C++ programming languages
- Flexibility and high-performance come with a cost — risk
- Secure Coding Standards:
  - CWE
  - CERT
  - MISRA
  - …

Common Weakness Enumeration (CWE)

- 由 MITRE 機構所定義出來的弱點分類
- Total Weaknesses : 875
- Total Categories: 312



Common Vulnerabilities and Exposures (CVE)

- 由 MITRE 機構維護

CVE-2020-9687 Detail

Current Description

Adobe Photoshop versions Photoshop CC 2019, and Photoshop 2020 have an out-of-bounds write vulnerability. Successful exploitation could lead to arbitrary code execution .

Weakness Enumeration

CWE-ID	CWE Name
CWE-787	Out-of-bounds Write

2020 CWE Top 25

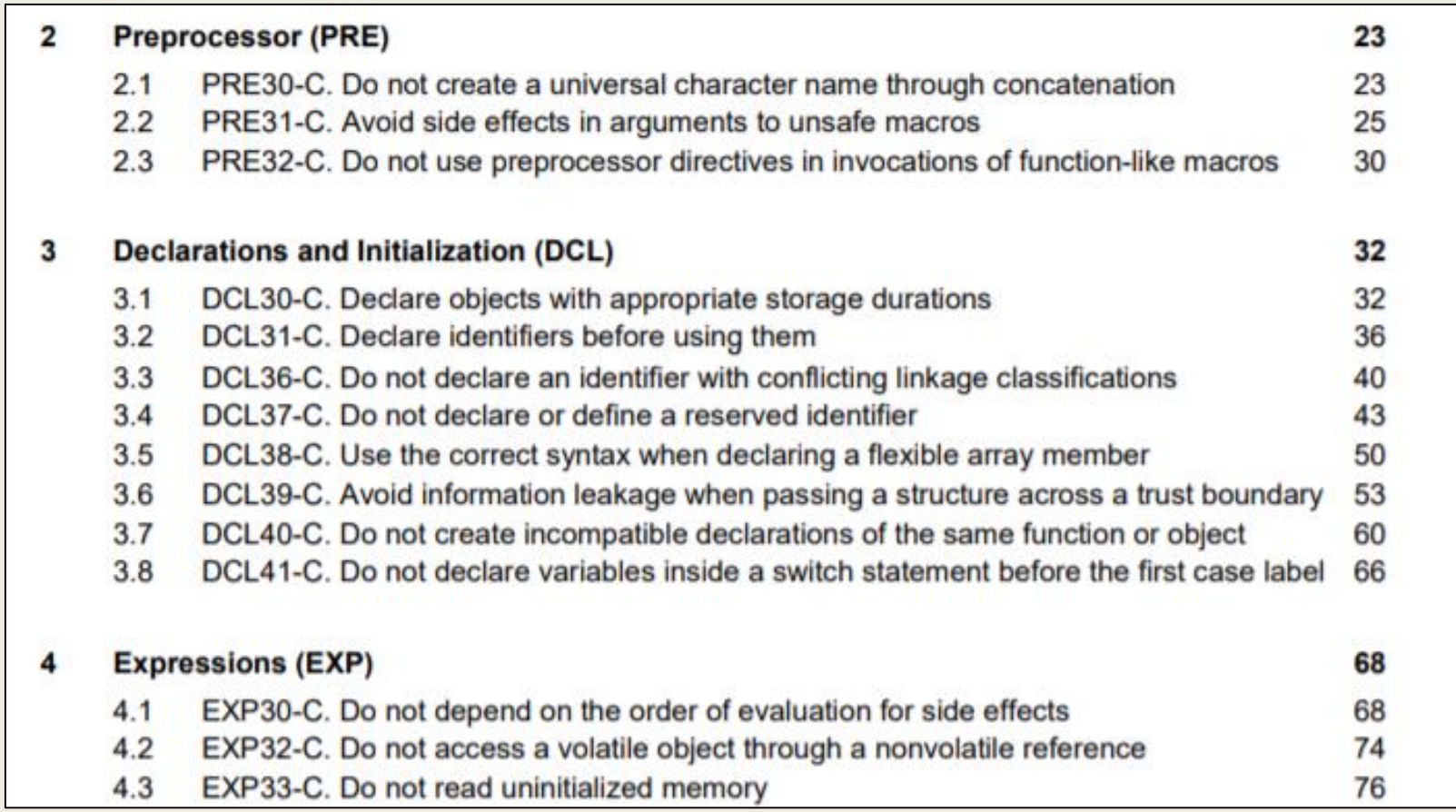
Rank	ID	Name	Score
[1]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	46.82
[2]	CWE-787	Out-of-bounds Write	46.17
[3]	CWE-20	Improper Input Validation	33.47
[4]	CWE-125	Out-of-bounds Read	26.50
[5]	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	23.73
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	20.69
[7]	CWE-200	Exposure of Sensitive Information to an Unauthorized Actor	19.16
[8]	CWE-416	Use After Free	18.87
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	17.29
[10]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	16.44
[11]	CWE-190	Integer Overflow or Wraparound	15.81
[12]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	13.67
[13]	CWE-476	NULL Pointer Dereference	8.35
[14]	CWE-287	Improper Authentication	8.17
[15]	CWE-434	Unrestricted Upload of File with Dangerous Type	7.38
[16]	CWE-732	Incorrect Permission Assignment for Critical Resource	6.95

Carnegie Mellon University (CMU) Software Engineering Institute (SEI) **computer emergency response team (CERT)**

https://resources.sei.cmu.edu/downloads/secure-coding/assets/sei-cert-c-coding-standard-2016-v01.pdf

SEI CERT C Coding Standard (2016 Edition)

1. 預處理 (PRE)
2. 聲明和初始化 (DCL)
3. 表達式 (EXP)
4. 整數 (INT)
5. 浮點數 (FLP)
6. 數組 (ARR)
7. 字符(數組)和字符串 (STR)
8. 內存管理 (MEM)
9. 輸入輸出 (FIO)
10. 環境 (ENV)
11. 信號 (SIG)
12. 錯誤處理 (ERR)
13. 並行性 (CON)
14. 雜項 (MSC)



FIO30-C. Exclude user input from format strings

- The syslog() function is also susceptible to format-string vulnerabilities.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <syslog.h>

void incorrect_password(const char *user) {
  int ret;
  /* User names are restricted to 256 or fewer characters */
  static const char msg_format[] = "%s cannot be authenticated.\n";
  size_t len = strlen(user) + sizeof(msg_format);
  char *msg = (char *)malloc(len);
  if (msg != NULL) {
    /* Handle error */
  }
  ret = snprintf(msg, len, msg_format, user);
  if (ret < 0) {
    /* Handle error */
  } else if (ret >= len) {
    /* Handle truncated output */
  }
  syslog(LOG_INFO, msg);
  free(msg);
}
```

FIO30-C. Exclude user input from format strings

- The syslog() function is also susceptible to format-string vulnerabilities.

```
#include <syslog.h>

void incorrect_password(const char *user) {
  static const char msg_format[] = "%s cannot be authenticated.\n";
  syslog(LOG_INFO, msg_format, user);
}
```

10.1.6 Risk Assessment

Failing to exclude user input from format specifiers may allow an attacker to crash a vulnerable process, view the contents of the stack, view memory content, or write to an arbitrary memory location and consequently execute arbitrary code with the permissions of the vulnerable process.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
FIO30-C	High	Likely	Medium	**P18**	**L1**

10.1.6 Risk Assessment

Failing to exclude user input from format specifiers may allow an attacker to crash a vulnerable process, view the contents of the stack, view memory content, or write to an arbitrary memory location and consequently execute arbitrary code with the permissions of the vulnerable process.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
FIO30-C	High	Likely	Medium	**P18**	**L1**

Severity—How serious are the consequences of the rule being ignored?

Value	Meaning	Examples of Vulnerability
1	Low	Denial-of-service attack, abnormal termination
2	Medium	Data integrity violation, unintentional information disclosure
3	High	Run arbitrary code

Likelihood—How likely is it that a flaw introduced by ignoring the rule can lead to an exploitable vulnerability?

Value	Meaning
1	Unlikely
2	Probable
3	Likely

Remediation Cost—How expensive is it to comply with the rule?

Value	Meaning	Detection	Correction
1	High	Manual	Manual
2	Medium	Automatic	Manual
3	Low	Automatic	Automatic

Priorities and Levels

Level	Priorities
L1	**12, 18, 27**
L2	**6, 8, 9**
L3	**1, 2, 3, 4**

MISRA (Motor Industry Software Reliability Association，汽車工業軟體可靠性協會)

- MISRA 提出的 C/C++ 語言開發標準: MISRA C、MISRA C++

Rule 13.6	The operand of the *sizeof* operator shall not contain any expression which has potential *side effects*
	C99 [Unspecified 21]
Category	Mandatory
Analysis	Decidable, Single Translation Unit
Applies to	C90, C99

Example

```
volatile int32_t i;
         int32_t j;
         size_t  s;

s = sizeof ( j );          /* Compliant              */
s = sizeof ( j++ );        /* Non-compliant          */
s = sizeof ( i );          /* Compliant - exception */
s = sizeof ( int32_t );    /* Compliant              */
```

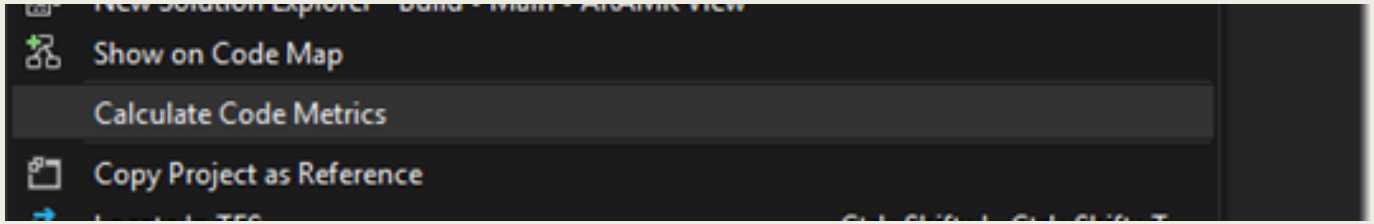
Static Source Code Analysis

- Scan source codes for potential vulnerabilities
  - Lots of False Positives

- Approach
  - Preparation: rule selection
  - Step 1: scan
  - Step 2: review issues
  - Step 3: fix approved issues

Software Metric（軟體度量）

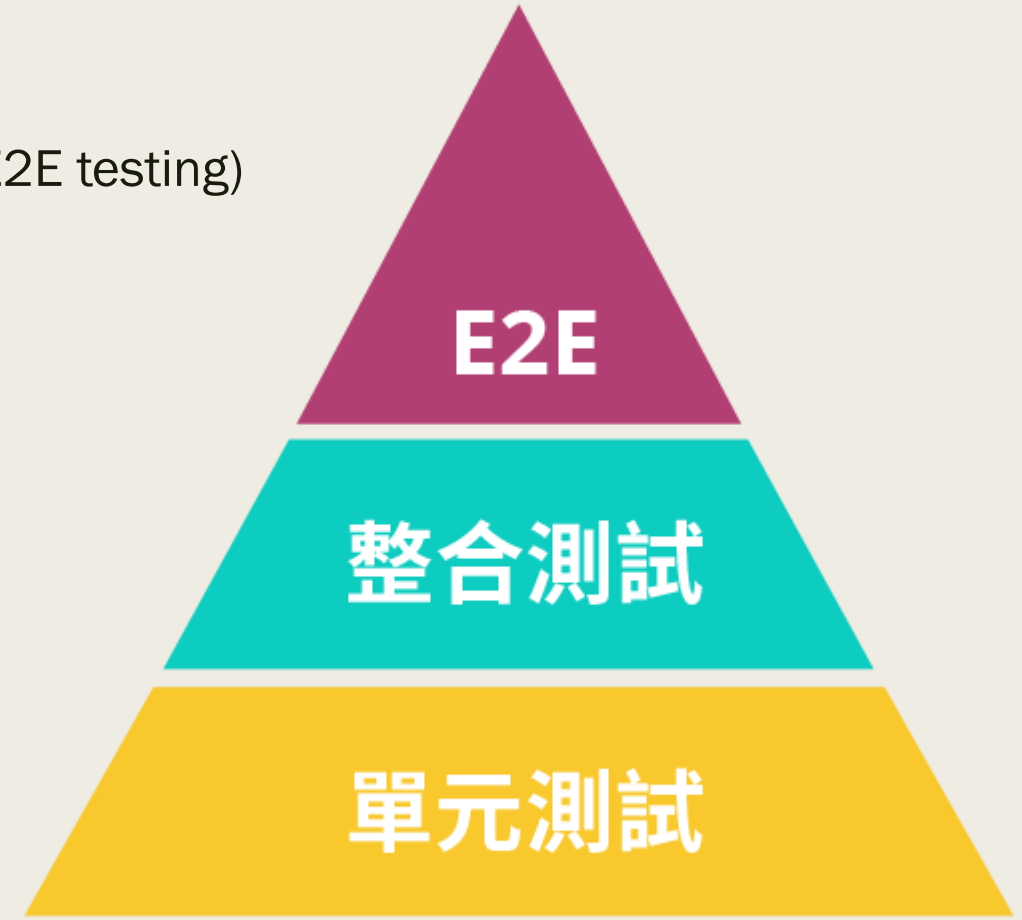
- Visual Studio:



可維護性指標	算出介於 0 到 100 之間的指數值，代表維護程式碼的相對難易程度。 值愈高表示可維護性愈佳。 色彩編碼分級可用來快速識別程式碼中的問題點。 綠色等級介於 20 和 100 之間，表示程式碼的可維護性良好。 黃色等級介於 10 和 19 之間，表示程式碼的可維護性適中。 紅色等級是介於 0 和 9 之間的等級，表示可維護性低。
循環複雜度	測量程式碼在結構上的複雜程度。 建立此複雜度的方式是計算程式流程中不同程式碼路徑的數目。 控制流程較為複雜的程式需要執行較多的測試，才能達到正確的程式碼涵蓋範圍，而且比較不容易維護。
繼承深度	指出延伸到類別 (Class) 階層的根 (Root) 的類別定義數目。 階層愈深，可能愈難找出定義與/或重新定義特定方法和欄位的位置。
類別結合程度	透過參數、區域變數、傳回型別、方法呼叫、泛型或樣板具現化、基底型別、介面實作、外部型別上定義的欄位以及屬性修飾等，測量特殊類別的結合程度。 良好的軟體設計應指定聚結性 (Cohesion) 高但結合程度 (Coupling) 低的型別和方法。 結合程度高表示設計不易重複使用，因為這種設計包含對其他型別的許多相依性。
程式碼行數	指出程式碼中行數的約略值。 這個數目是以 IL 程式碼為依據，因此不是原始程式碼檔案中精確的行數。 如果數目非常大，表示型別或方法嘗試執行的工作可能過多，而應該分割工作。 這也表示該型別或方法可能難以維護。

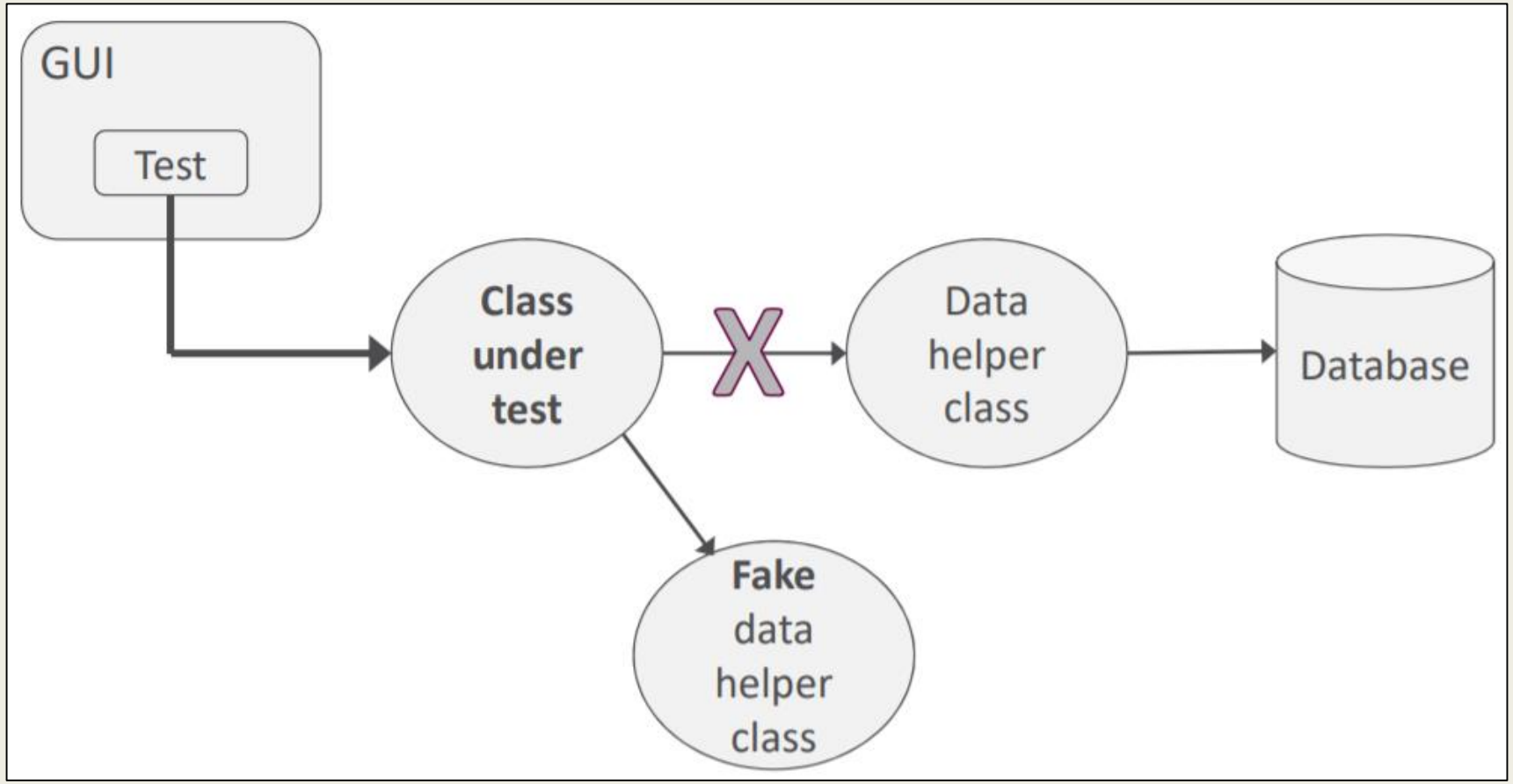
Testing

- 端對端測試 (End-to-end testing) (E2E testing)
- 整合測試 (Integration testing)
- 單元測試 (Unit testing)



Unit tests

- Dependency breaking



Unit tests

- Typically automated tests written and run by software developers

- Should be
  - Automated
  - Fast
  - Easy to run
  - Repeatable
  - Consistent

對一個類別來說，那些行為可提升可測性?

- 不要存取全域變數、config
- 不要自己建立用到的物件，而是透過外部傳入。這樣測試時才方便抽換相依物件，易於準備測試環境
- 不要自己取得系統時間。自己取得時間可能造成程式不確定性，讓測試不好驗證結果。應該要透過別的物件取得時間，才有機會用假時間加速測試。進一步說，通常需要的是 Timer，而不是時間。使用假的 Timer 可降低不確定性。
- 不要使用亂數。有時需要的是 Selector，而不是亂數。改成傳入 Selector 後，除了易於測試外，還多了未來擴充功能的彈性。像是選擇隨機物件、或是 round-robin 方式選物件

Test-Driven Development 測試驅動開發

- 開發團隊寫測試，通常有三種模式：
  - 先寫測試再開發 (TDD)
  - 開發完成再寫測試
  - 無招勝有招 —— 不寫測試（誤）

Test-Driven Development 測試驅動開發

- TDD 先寫測試再開發，開發時的實用「最低限度」通過測試案例即可。
- 正面評價
  - **有助於在開發初期釐清程式介面如何設計**：在實作時，我們通常會希望程式介面維持穩定，越少改動越好。但在開發初期憑空定義出來的介面，常常在開發完成實際使用時才發現不好用，導致介面需要頻繁改動。
  - **避免過度設計**：通過測試案例即可。
- 負面評價
  - **還是需要補充單元測試**：測試驅動開發會導致單元測試的覆蓋度不夠，當代碼完成以後還是需要補充單元測試，提高測試的覆蓋度。
  - **存在盲點**：測試和開發都是同一人(通常)。

Behavior-Driven Development 行為驅動開發

- TDD 的不足 ── 非技術人員難以參與討論。
- BDD 比 TDD 更進一步，在寫測試前先寫測試規格書。
- 這份測試規格會用更接近人類語意的自然語言來描述軟體功能和測試案例。
- 而且這份規格不是單純的敘述文件，而是一份「**可以被執行的規格**」，也就是可以被轉成自動化測試。

Behavior-Driven Development 行為驅動開發

- BDD 可以讓非技術人員一起參與討論，
- 而工程師也可以直接拿這份規格轉化成測試案例，讓規格成為「可執行的規格」，
- 方便各種不同背景的人一起討論，容易對軟體專案有更一致的共識。

```
功能: 工會與各地勞動局的聯絡方式

  場景: 提供工會的聯絡方式，如官方網站、FB 粉絲團、email 以及連絡電話
    當使用者進入網站
    並且點選 "檢舉或諮詢" 連結
    並且點選 "工會" 連結
    那麼應該出現工會列表
    而且從工會列表中點選 "台灣電子電機資訊產業工會" 後，應該出現其聯絡方式
```

Behavior-Driven Development 行為驅動開發

- 將「規格」轉換成「測試案例」

```
Warnings:

1) Scenario: 提供工會的聯絡方式，如官方網站、FB 粉絲團、email 以及連絡電話 - features/contact.feature:5
   Step: 當使用者進入網站 - features/contact.feature:6
   Message:
     Undefined. Implement with the following snippet:

       When('使用者進入網站', function (callback) {
         // Write code here that turns the phrase above into concrete actions
         callback(null, 'pending');
       });

2) Scenario: 提供工會的聯絡方式，如官方網站、FB 粉絲團、email 以及連絡電話 - features/contact.feature:5
   Step: 並且點選 "檢舉或諮詢" 連結 - features/contact.feature:7
   Message:
     Undefined. Implement with the following snippet:

       When('點選 {stringInDoubleQuotes} 連結', function (stringInDoubleQuotes, callback) {
         // Write code here that turns the phrase above into concrete actions
         callback(null, 'pending');
       });
```

Do not release testing codes!

- Minimize Attack Surface、Information Disclosure
  - Remove it

```
1 package com.google.api.client.testing.http;
2 class HttpTesting {
3   static String SIMPLE_URL = "http://google.com"
4   public HttpTesting() {
5     GenericUrl url = new GenericUrl(SIMPLE_URL);
6   } ...
```

Dynamic Security Scanning

- Limited to web security scanning and network scanning
- Use several commercial and free tools
- Process
  - Project teams submit scan requests
  - Operations team conduct scanning
  - Project teams fix issues based on the scanning results

Automated Program Analysis

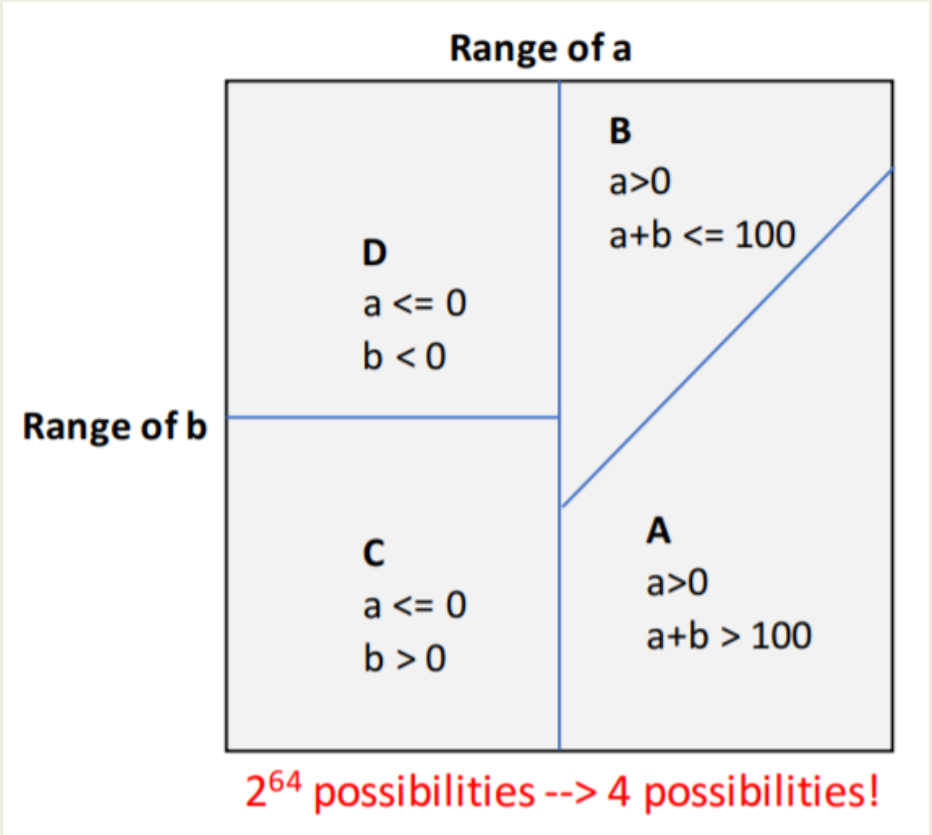
- Core Concept:
  - Examines possible values and see whether any one leads to crash.
- Challenge:
  - How to intelligently explore the input space?
- Two popular techniques
  - Fuzzing
  - Symbolic execution

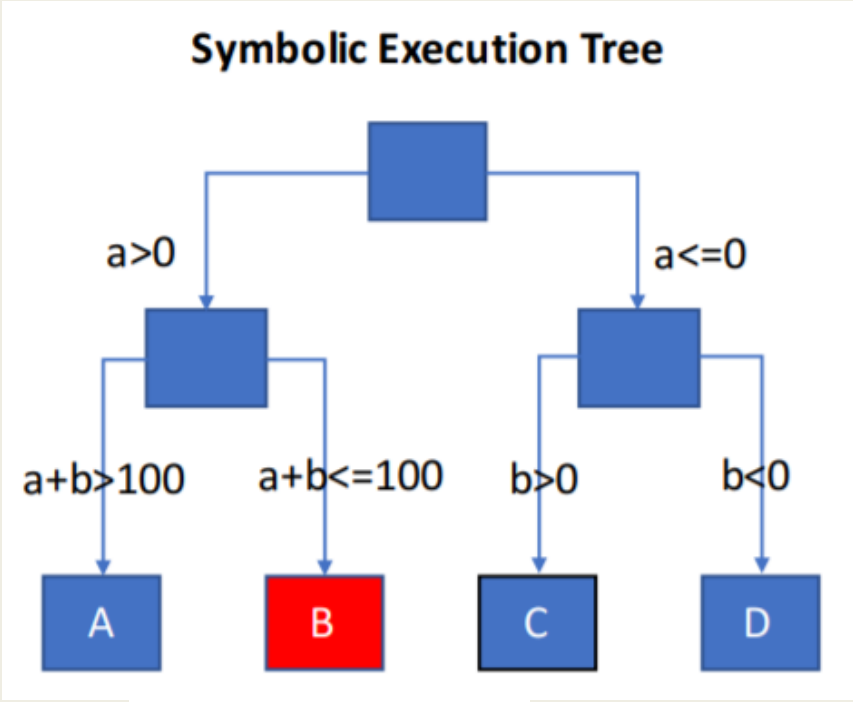
Fuzzing 模糊測試

- 核心概念：隨機地生成輸入，試試看是否能觸發漏洞。
- 通常會藉由某些啟發式策略，如覆蓋率導向（coverage-based），排序尚待測試的輸入值。
- 實務上被廣泛使用之技術
  - E.g. American Fuzzy Lop
- 優點：簡單、有效率
- 缺點：很難通過複雜的條件式
  - 很難聰明地產出正確的輸入，通過比較嚴格的條件式，因此模糊測試通常只能找到位於較淺層之漏洞。

Symbolic Execution 符號執行

```
void foo(int a, int b) {
  if (a > 0){
    if (a + b > 100) {
      ...          A
    }
    else {
      buggy();     B
    }
  }
  else {
    if (b > 0) {
      ...          C
    }
    else {
      ...          D
    }
}
```

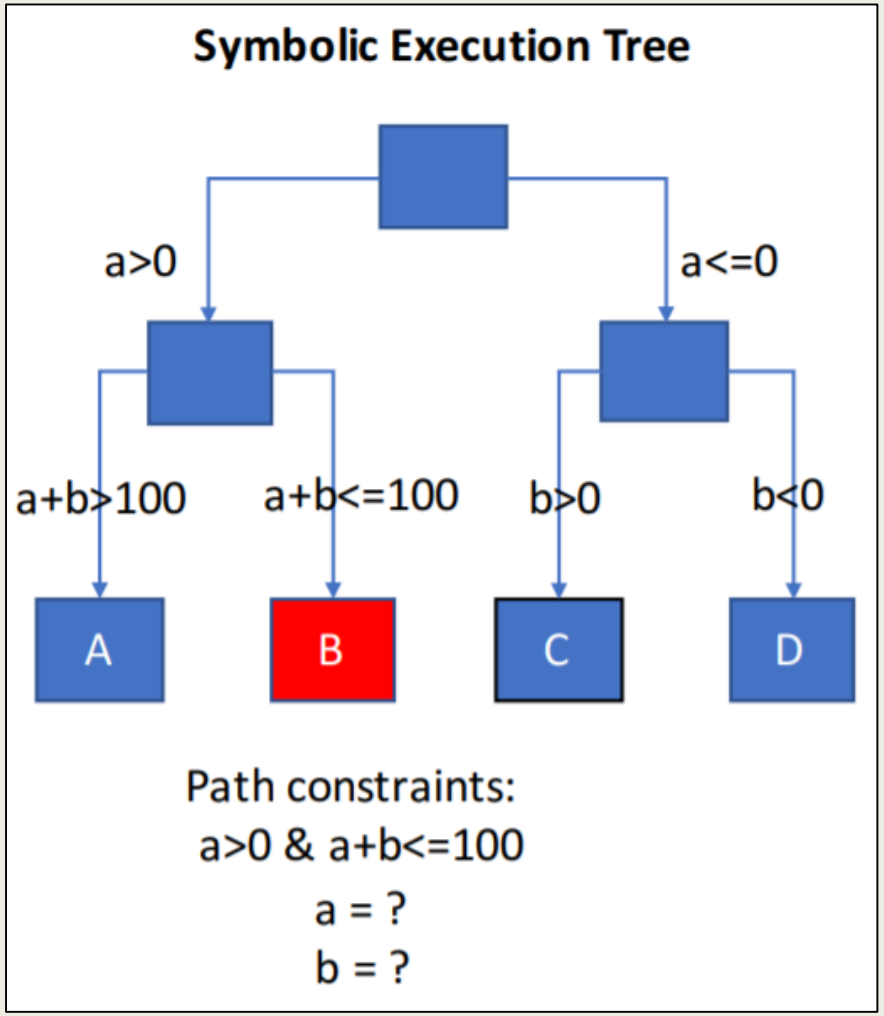




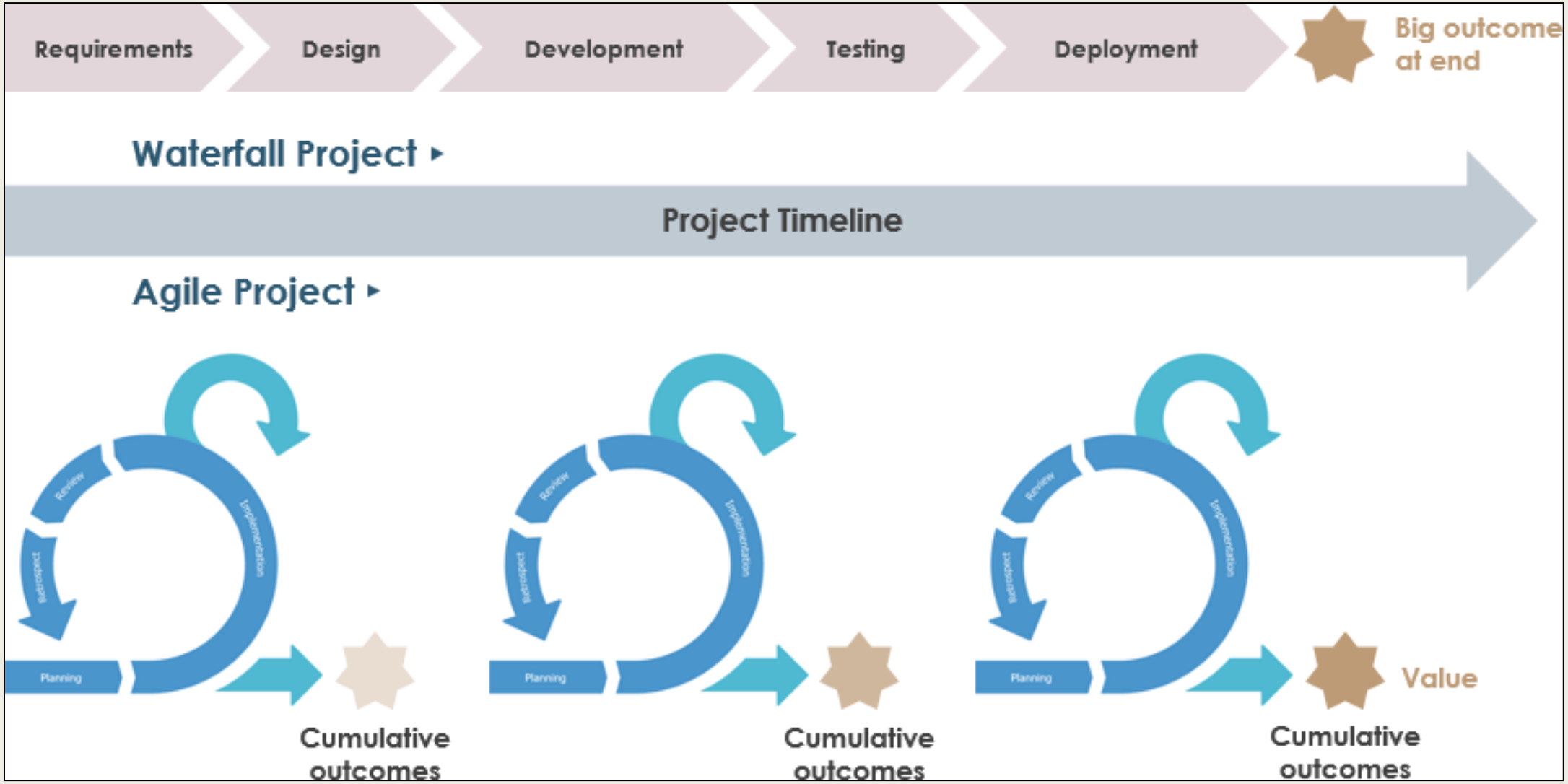
Path constraints:
a>0 & a+b<=100
a = ?
b = ?

SMT Solvers

- Satisfiability Modulo Theories (SMT) generalizes the SAT problem
  - SAT: Satisfiability of Boolean formula
- Checking the feasibility of a path
- Generating assignments to symbolic variables
  - E.g. a,b



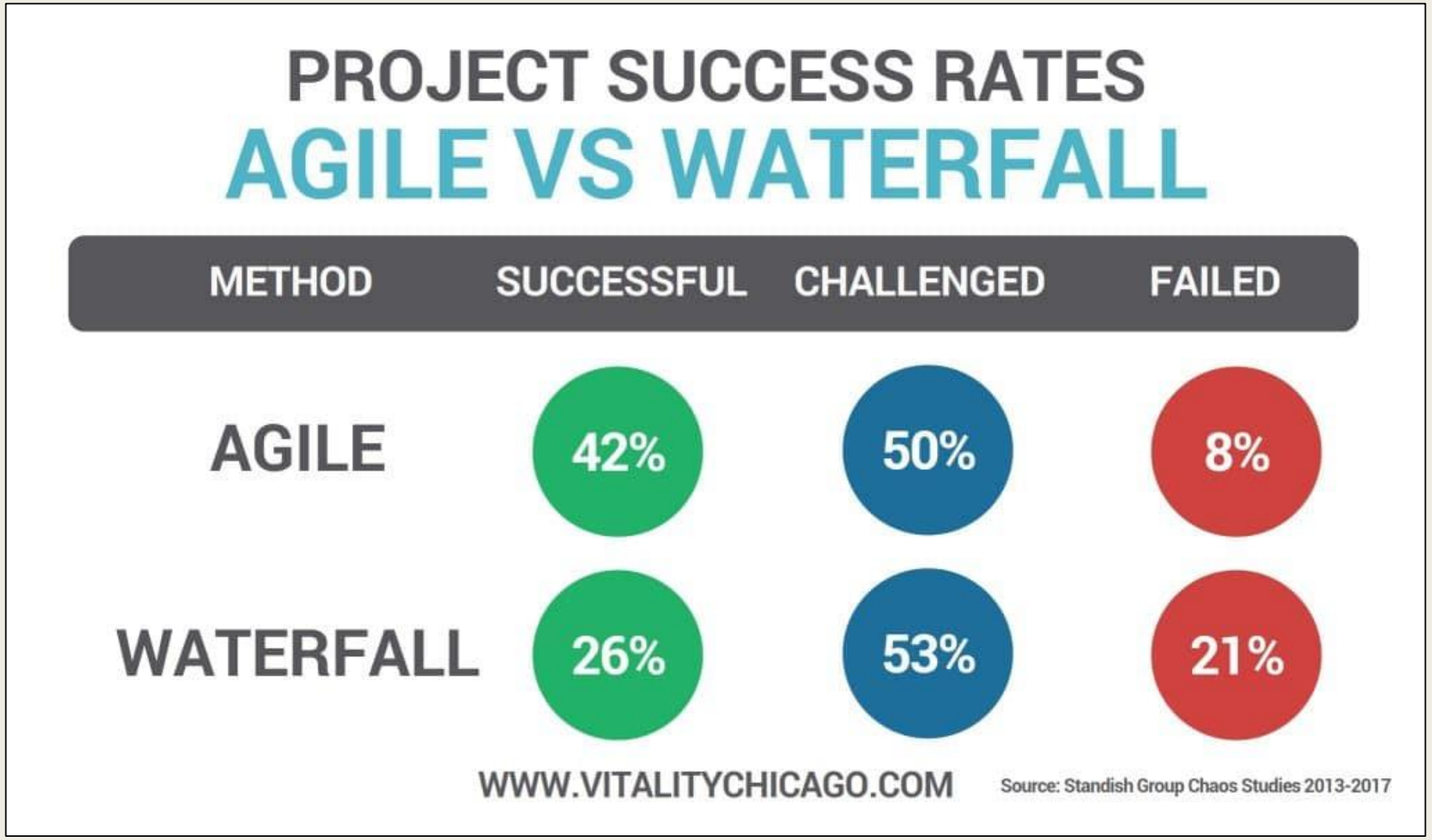
Waterfall vs Agile



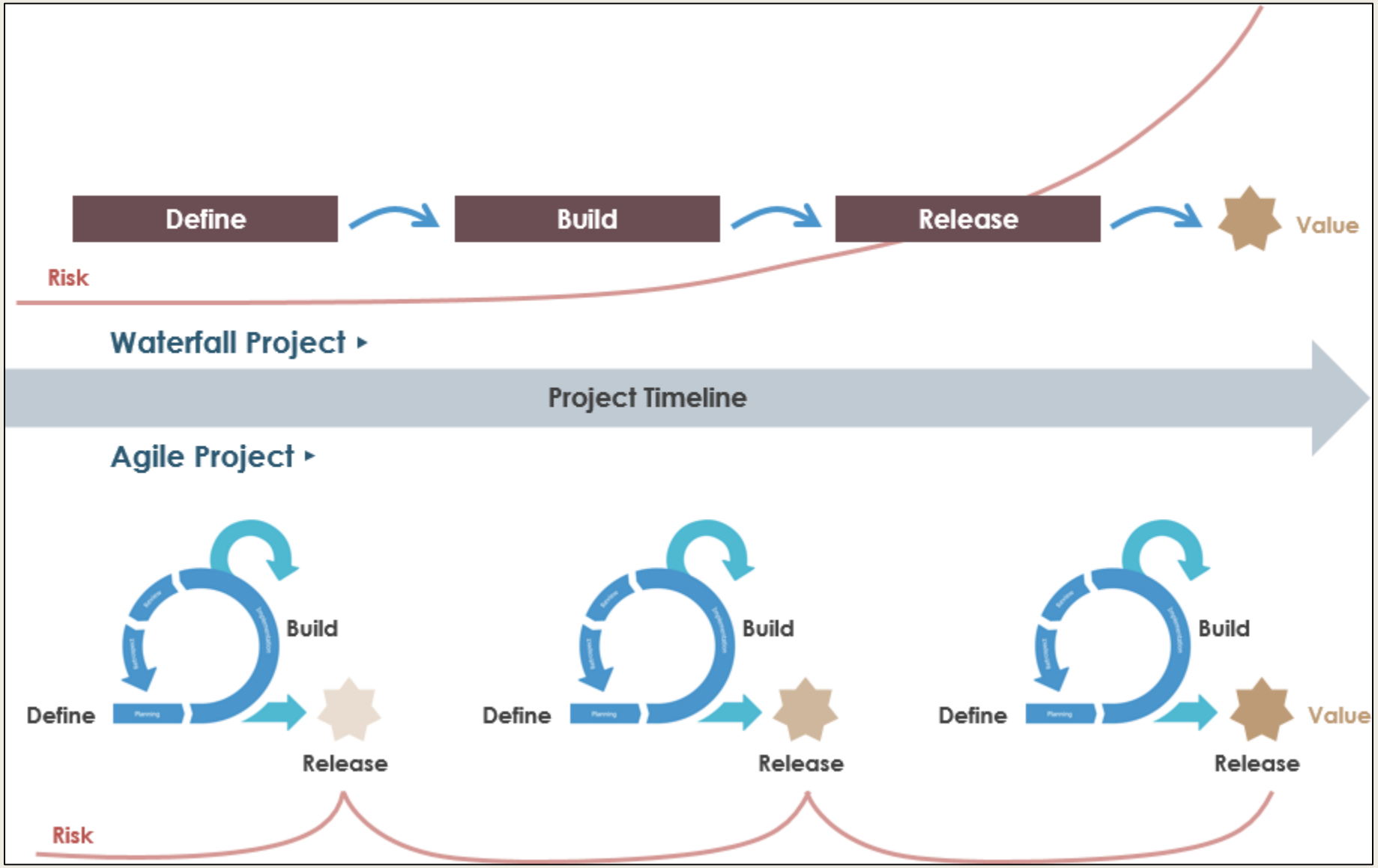
Waterfall vs Agile

- Office 每年的一期一會，但新功能可能沒解決到用戶的痛處，反而增添了更多不必要的麻煩

- Facebook 3天一小改，5天一大改


PROJECT SUCCESS RATES
AGILE VS WATERFALL
METHOD
SUCCESSFUL
CHALLENGED
FAILED
AGILE
42%
50%
8%
WATERFALL
26%
53%
21%
WWW.VITALITYCHICAGO.COM
Source: Standish Group Chaos Studies 2013-2017

Waterfall vs Agile



Waterfall Model 瀑布模型

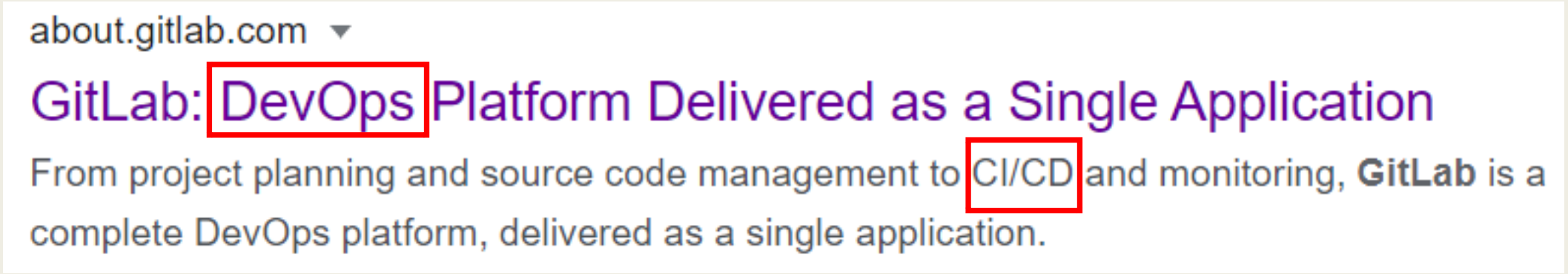
- 好處
  - 目標清晰 容易管理
  - 每個階段都確保沒有出錯才繼續
  - (有好有壞) 不需要客戶恆常參與討論

- 限制
  - 與客戶較少交流 無法緊貼要求
  - 出錯後較難返回上一步
  - 缺乏彈性 無法靈活應對新環境

Agile Development 敏捷式開發

- 好處
  - 快速、持續地完成短期目標
  - 團隊與客戶緊密合作
  - 省卻不必要文件
  - 適應不斷變化的環境
- 限制
  - **缺乏規劃、結果可能與預期不同**：由於初始計劃是粗略的，並且可能在整個開發週期中添加額外的衝刺，因此並不總是能夠及時確定準確的交付日期和完成任務。
  - **花大量時間面對面溝通**：每天在 Stand Up Meeting 匯報進度、跟客戶定期見面交流意見，敏捷式開發重視人與人之間互動，故此需投放大量溝通時間
  - **沒有詳盡文件 新同事難以掌握現況**：這種開發模式專注於 tasks，雖然減省了不必要的文件，但當團隊有新成員時，他們不能透過項目文件自行了解詳情，需向其他成員請教

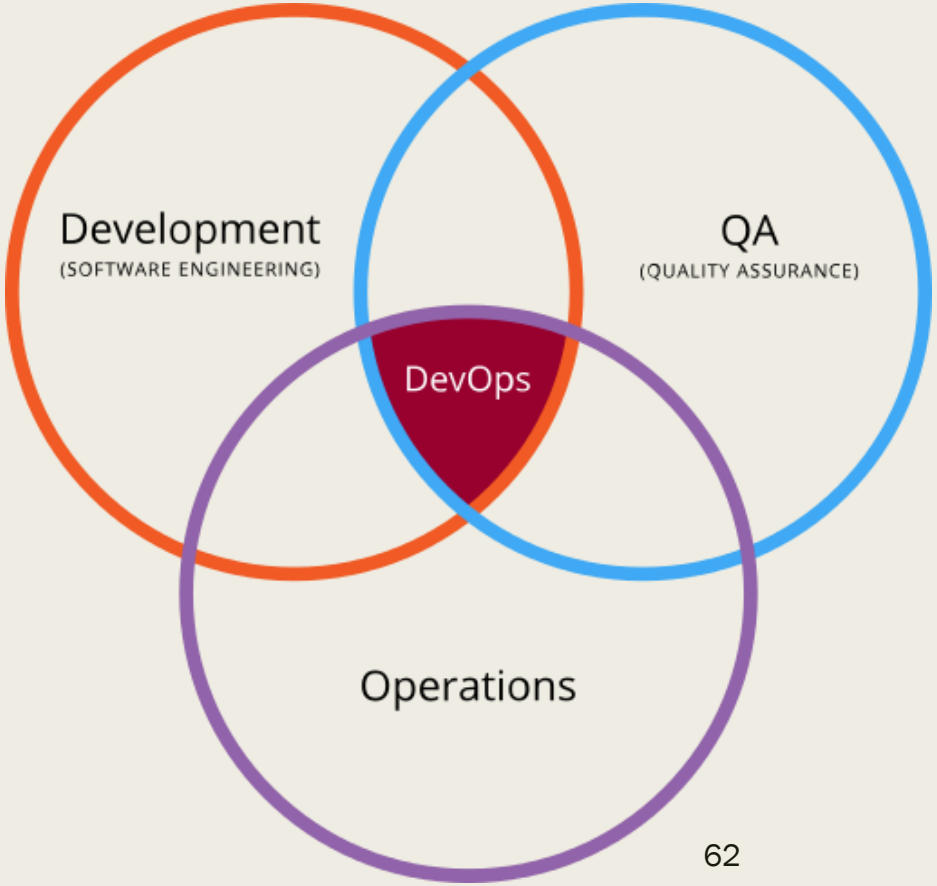
What Is DevOps?
What Is CI/CD?



DevOps

- Development + Operations ，也就是開發與維運。
- 可以把 DevOps 看作「開發」「測試」「維運」三者的緊密結合。

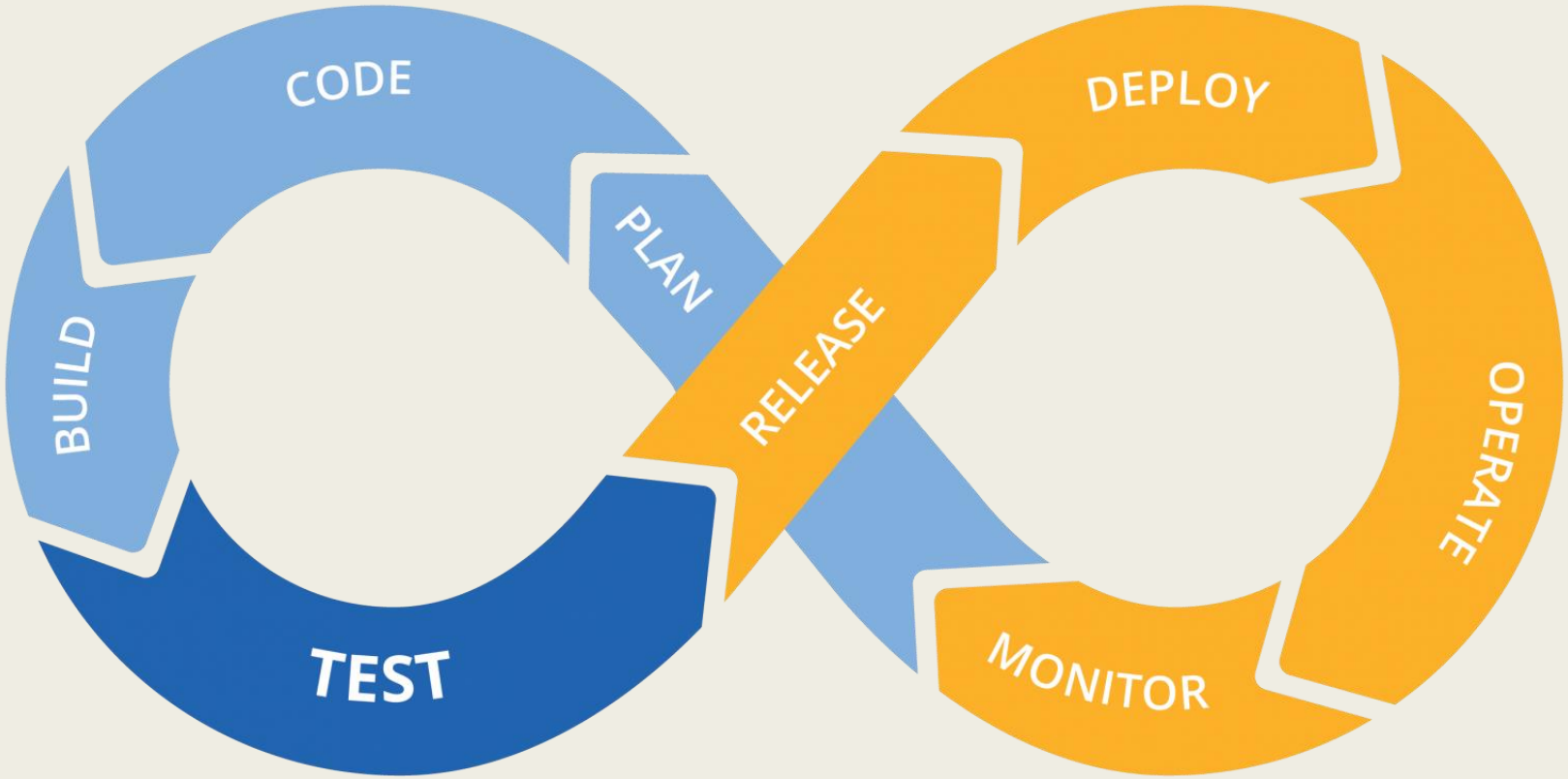
- 目的是提升組織快速交付應用程式和服務的能力：
  - 更快速地開發和改進產品
  - 提供更好的服務給客戶
  - 在市場上更有效率地競爭



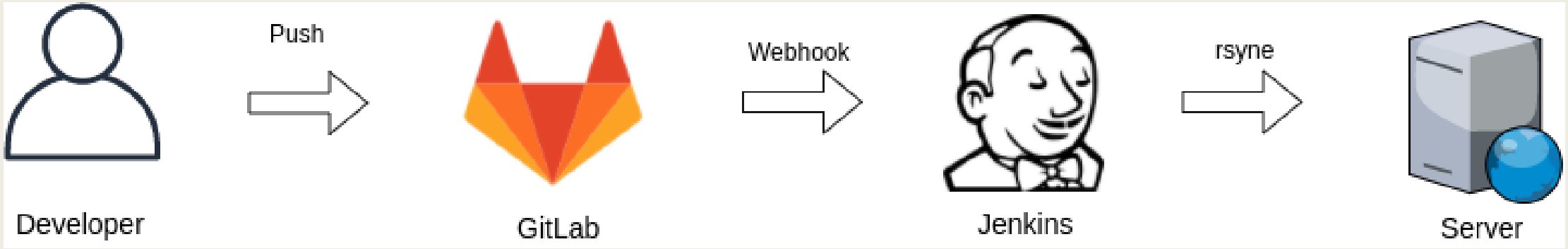
CICD

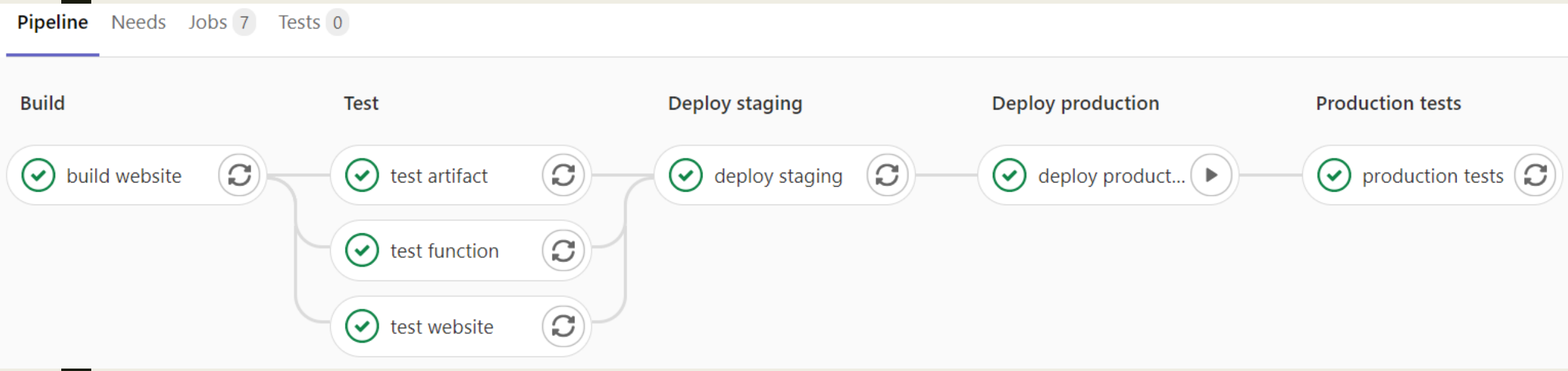
- 持續整合 (Continuous Integration, CI)
  - 新程式碼越多、整合的時間越晚，整合的難度與失敗的機率就越高。
  - 利用頻繁地提交新功能的變更，觸發自動化建置和測試，確保最新版本的軟體是可運行的。
- 持續部署 (Continuous Deployment, CD)
  - 自動化將最新版本的軟體更新到機器上，公開對外服務
- 流程自動化
  - 減少人工手動的反覆步驟
  - 降低人為疏失風險

CICD 通常會遵循著以下流程：



CICD


Push
Webhook
rsyne
Developer
GitLab
Jenkins
Server

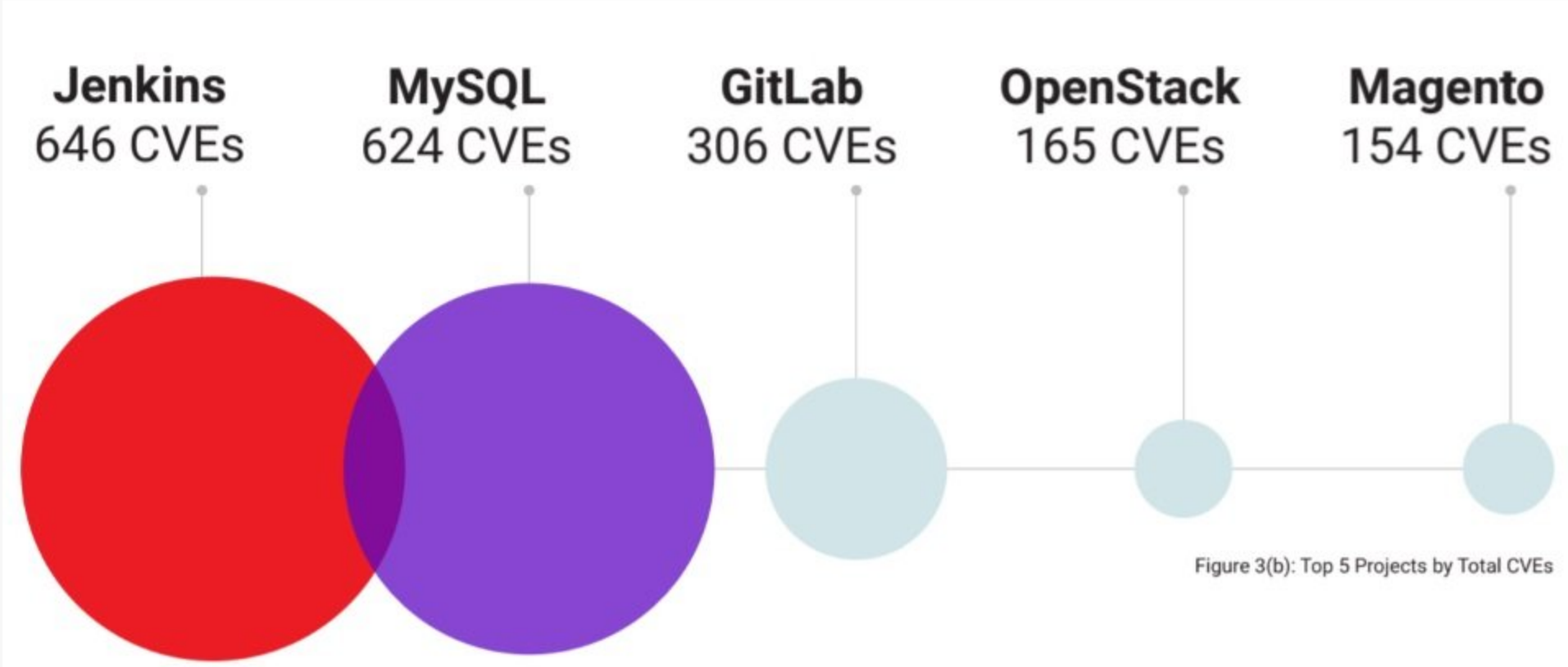

Pipeline
Needs
Jobs 7
Tests 0
Build
Test
Deploy staging
Deploy production
Production tests
build website
test artifact
test function
test website
deploy staging
deploy product...
production tests

新聞

MySQL、Jenkins是漏洞最多的兩個開源碼專案

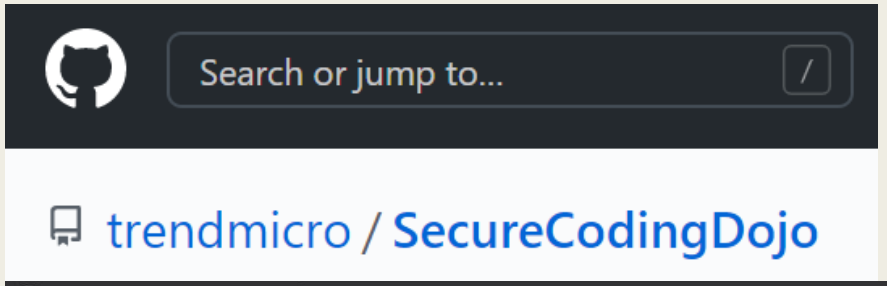
RiskSense追蹤54個主要開源專案從2015年至今公開的CVE漏洞代碼，發現過去3年來的漏洞數量，呈現逐年倍增的趨勢

文/ 林妍溱 | 2020-06-10 發表



RiskSense檢視54項主要的開源專案，從2015年到2020年3月底的公開CVE漏洞代碼，專案含有漏洞數量最多的前二大，是持續整合工具Jenkins（646項漏洞），以及資料庫軟體MySQL（624項漏洞）。（Photo by RiskSense）

https://github.com/trendmicro/SecureCodingDojo



設定檔 VS 環境變數

```
{
    "dojoUrl" : "http://140.112.18.215:8081",
    "moduleUrls" : {
        "blackBelt":"http://140.112.18.215:8080",
        "securityCodeReviewMaster":"https://trendmicro.github.io/SecureCodingDojo/codereview101/?fromPortal"
    },

    "disabledModules":["secondDegreeBlackBelt","redTeam","blueTeam"],

    "playLinks" : {},

    "localUsersPath" : "localUsers.json"
}
```

- Edit `~/.bash_profile` or `/etc/environment`

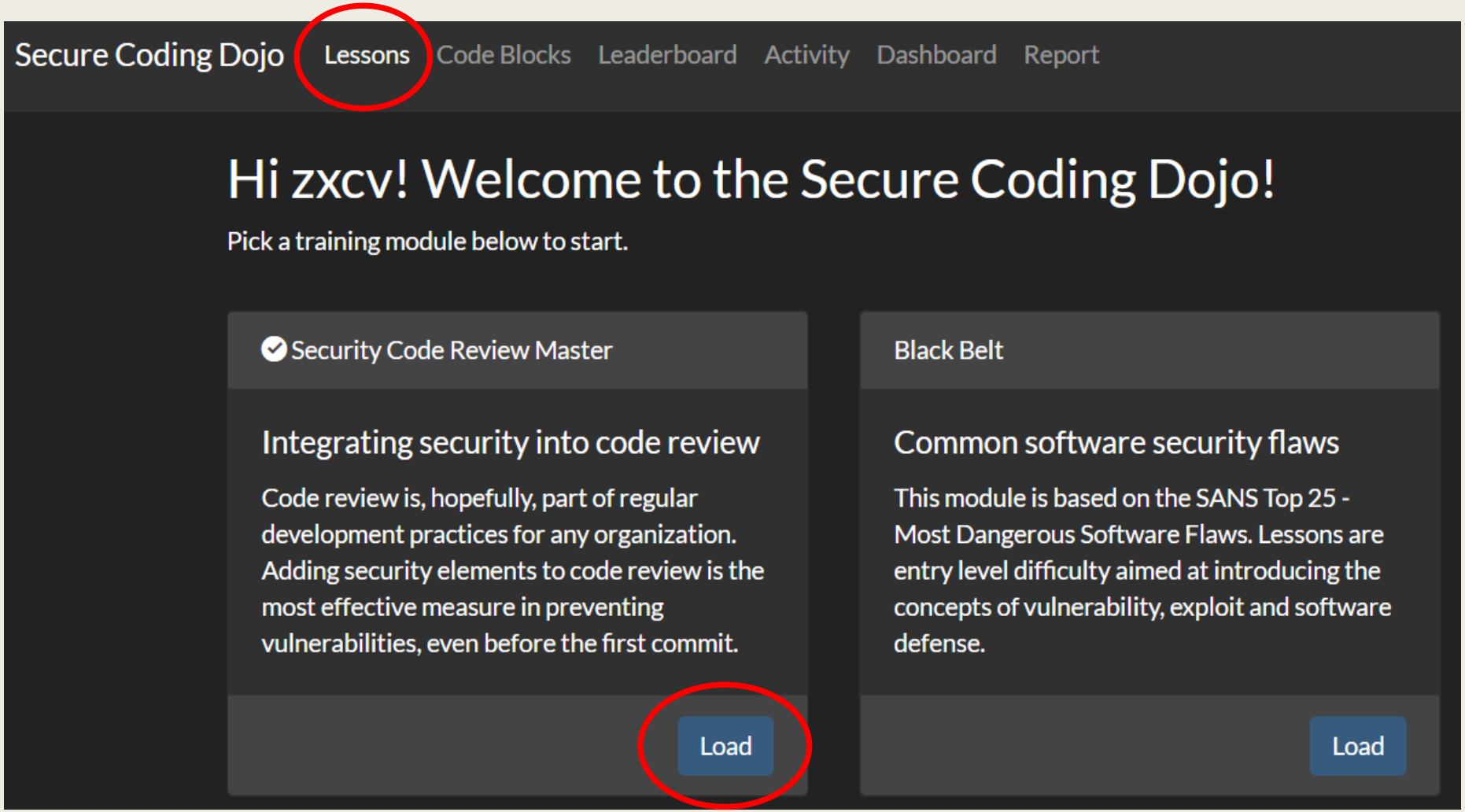
```
export ENC_KEY="put something random here"
export ENC_KEY_IV="put something random here"
export CHALLENGE_MASTER_SALT="put something random here"
```

HW

- http://140.112.18.215:8081/public/index.html
- User name: 小寫學號

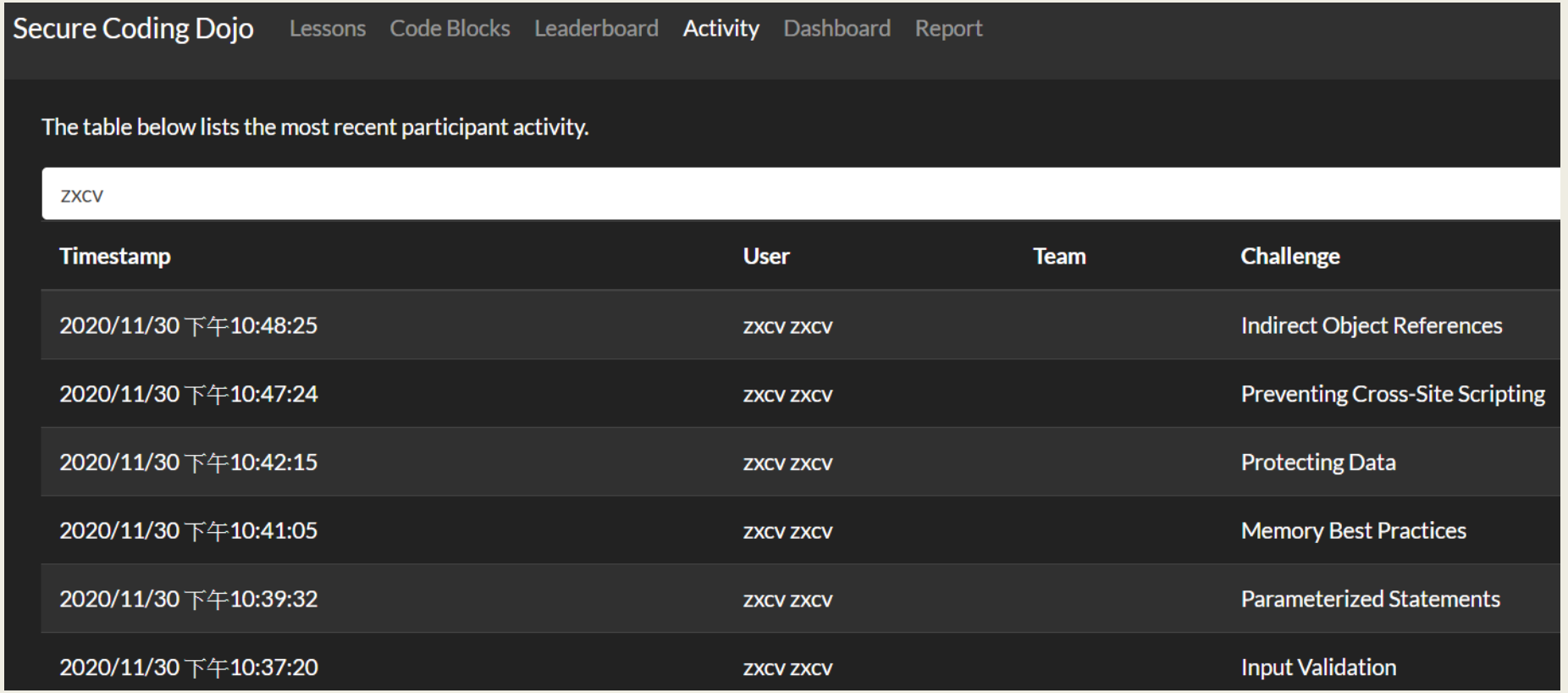
HW

- 完成 Security Code Review Master



HW

- 完成 Security Code Review Master
- 評分方式: 搜尋學號看是否完成這 6 題



SECURITY PROJECT

CRYLOGGER: Detecting Crypto Misuses Dynamically

IEEE Symposium on Security & Privacy (SP) 2021

Crypto rules that are considered

ID	Rule Description	Ref.
R-01	Don't use broken hash functions (`SHA1`, `MD2`, `MD5`, ..)	[8]
R-02	Don't use broken encryption alg. (`RC2`, `DES`, `IDEA` ..)	[8]
R-03	Don't use the operation mode `ECB` with $>$ 1 data block	[5]
R-04 †	Don't use the operation mode `CBC` (client/server scenarios)	[12]
R-05	Don't use a static (= constant) key for encryption	[5]
R-06 †	Don't use a "badly-derived" key for encryption	[5]
R-07	Don't use a static (= constant) initialization vector (IV)	[5]
R-08 †	Don't use a "badly-derived" initialization vector (IV)	[5]
R-09 †	Don't reuse the initialization vector (IV) and key pairs	[46]
R-10	Don't use a static (= constant) salt for key derivation	[5]
R-11 †	Don't use a short salt ($<$ 64 bits) for key derivation	[14]
R-12 †	Don't use the same salt for different purposes	[46]
R-13	Don't use $<$ 1000 iterations for key derivation	[14]

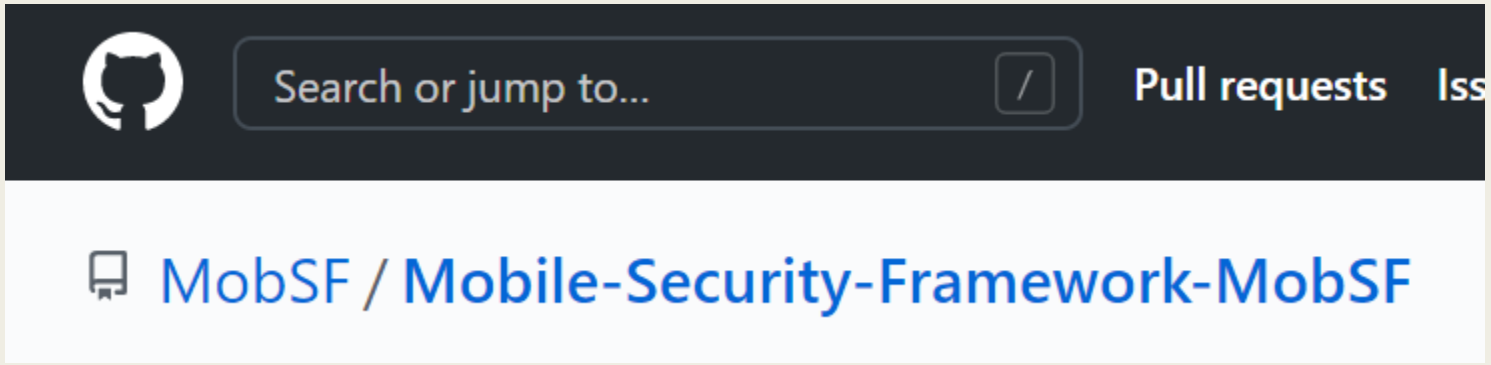
ID	Rule Description	Ref.
R-14 †	Don't use a weak password (score $<$ 3)	[47]
R-15 †	Don't use a NIST-black-listed password	[48]
R-16	Don't reuse a password multiple times	[48]
R-17	Don't use a static (= constant) seed for PRNG	[49]
R-18	Don't use an unsafe PRNG (java.util.Random)	[49]
R-19	Don't use a short key ($<$ 2048 bits) for RSA	[13]
R-20 †	Don't use the textbook (raw) algorithm for RSA	[50]
R-21 †	Don't use the padding `PKCS1-v1.5` for RSA	[51]
R-22	Don't use HTTP URL connections (use HTTPS)	[16]
R-23	Don't use a static (= constant) password for store	[48]
R-24	Don't verify host names in SSL in trivial ways	[16]
R-25	Don't verify certificates in SSL in trivial ways	[16]
R-26	Don't manually change the hostname verifier	[16]

TABLE I

Crypto rules that are considered in this paper. The symbol † indicates the rules that are not covered by other approaches (we used [6] as reference).

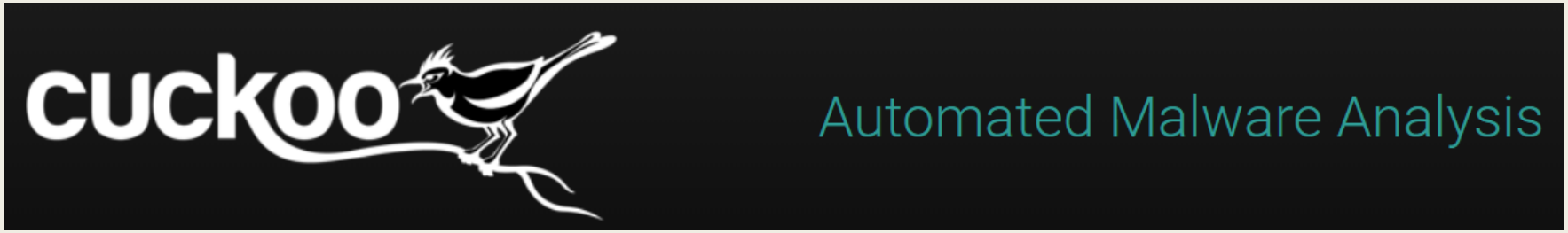
Dynamic analysis

https://github.com/MobSF/Mobile-Security-Framework-MobSF

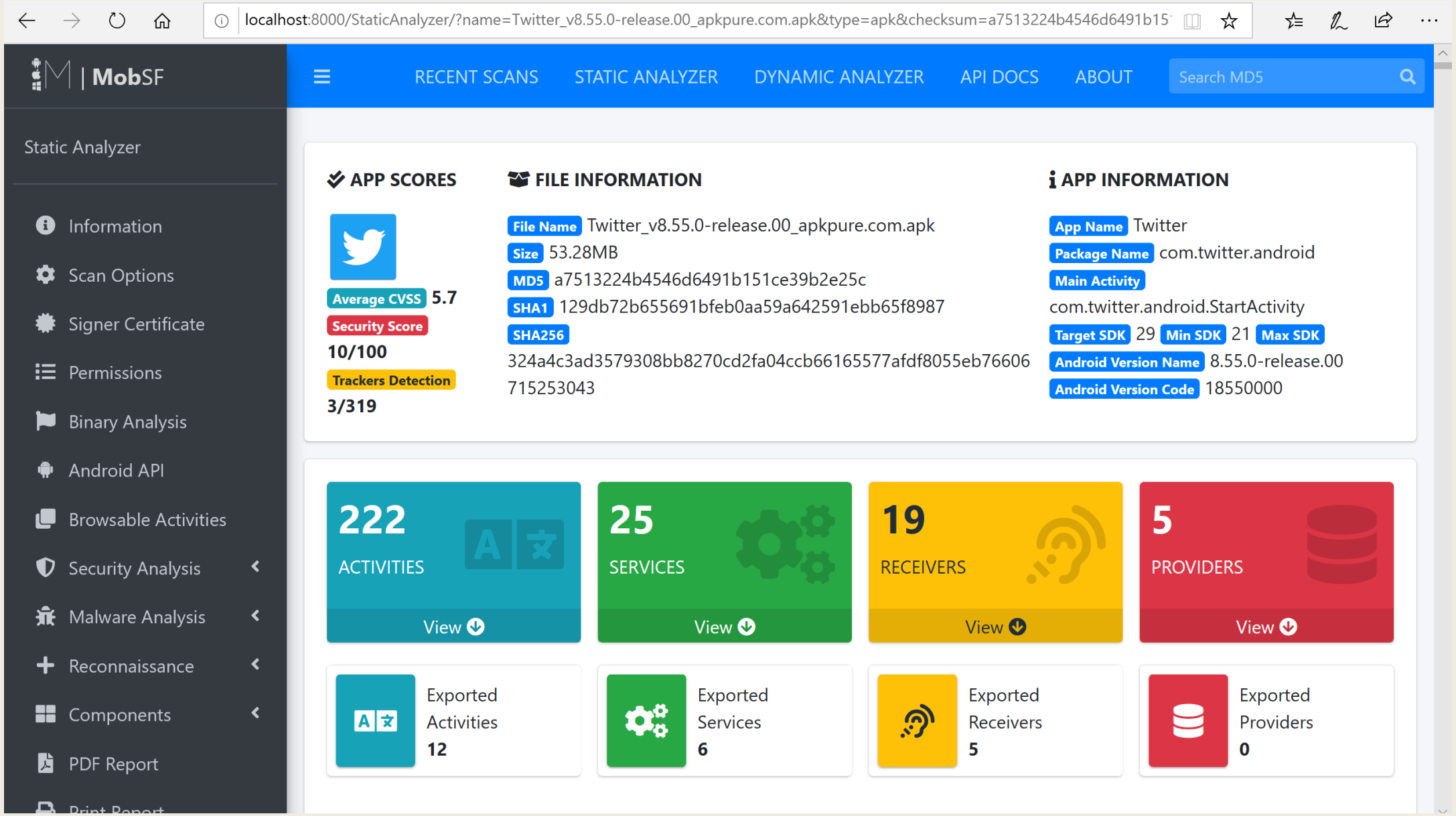


MobSF support mobile app binaries (APK, IPA & APPX) along with zipped source code and provides REST APIs for seamless integration with your CI/CD or DevSecOps pipeline.

https://cuckoosandbox.org/



Cuckoo Sandbox is free software that automated the task of analyzing any malicious file under Windows, macOS, Linux, and Android.


localhost:8000/StaticAnalyzer/?name=Twitter_v8.55.0-release.00_apkpure.com.apk&type=apk&checksum=a7513224b4546d6491b15
MobSF
RECENT SCANS
STATIC ANALYZER
DYNAMIC ANALYZER
API DOCS
ABOUT
Search MD5
Static Analyzer
Information
Scan Options
Signer Certificate
Permissions
Binary Analysis
Android API
Browsable Activities
Security Analysis
Malware Analysis
Reconnaissance
Components
PDF Report
APP SCORES
Average CVSS 5.7
Security Score
10/100
Trackers Detection
3/319
FILE INFORMATION
File Name Twitter_v8.55.0-release.00_apkpure.com.apk
Size 53.28MB
MD5 a7513224b4546d6491b151ce39b2e25c
SHA1 129db72b655691bfeb0aa59a642591ebb65f8987
SHA256
324a4c3ad3579308bb8270cd2fa04ccb66165577afdf8055eb76606715253043
APP INFORMATION
App Name Twitter
Package Name com.twitter.android
Main Activity
com.twitter.android.StartActivity
Target SDK 29 Min SDK 21 Max SDK
Android Version Name 8.55.0-release.00
Android Version Code 18550000
222
ACTIVITIES
View
25
SERVICES
View
19
RECEIVERS
View
5
PROVIDERS
View
Exported Activities 12
Exported Services 6
Exported Receivers 5
Exported Providers 0

Crypto rules checking

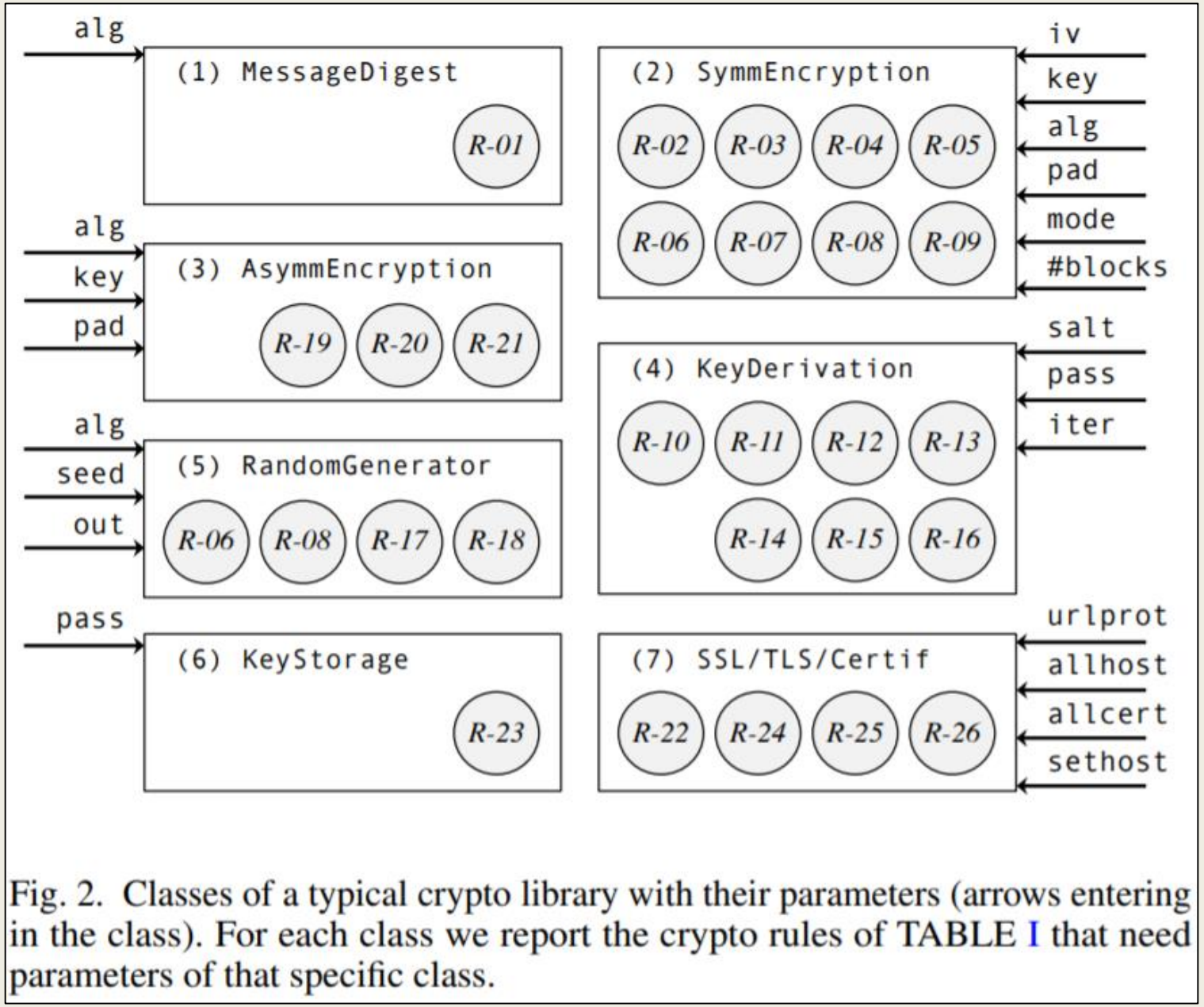
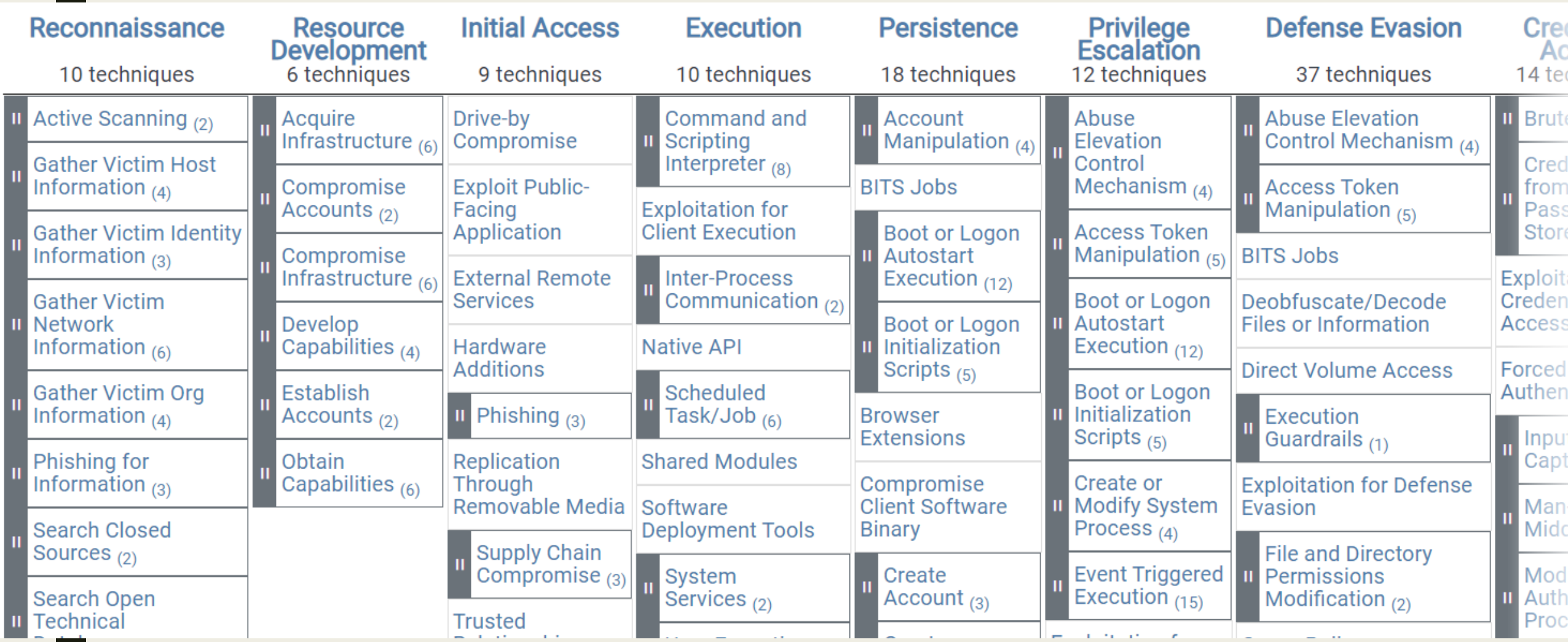


Fig. 2. Classes of a typical crypto library with their parameters (arrows entering in the class). For each class we report the crypto rules of TABLE I that need parameters of that specific class.

MobSF\DynamicAnalyzer\tools\frida_scripts\default\api_monitor.js

```
519         var overloadCount = toHook.overloads.length;
520         for (var i = 0; i < overloadCount; i++) {
521             toHook.overloads[i].implementation = function () {
522                 var argz = [].slice.call(arguments);
523                 // Call original function
524                 var retval = this[method].apply(this, arguments);
525                 if (callback) {
526                     var calledFrom = Exception.$new().getStackTrace().toString().split(',')[1];
527                     var message = {
528                         name: name,
529                         class: clazz,
530                         method: method,
531                         arguments: argz,
532                         result: retval ? retval.toString() : null,
533                         calledFrom: calledFrom
534                     };
535                     retval = callback(retval, message);
536                 }
537                 return retval;
538             }
539         }
```

- Automated
  - ✓ Crypto rules checking
  - – Identifying Threats with MITRE ATT&CK ?



美國 DARPA (國防高等研究計劃署)舉辦的自動網路攻防競賽

- Cyber Grand Challenge 簡介
  - https://kb.hitcon.org/post/131158681227/cyber-grand-challenge-%E7%B0%A1%E4%BB%8B

- DARPA 文件中有提及的技術：
  1. Dynamic Analysis
  2. Static Analysis
  3. Symbolic Execution
  4. Constraint Solving
  5. Data Flow Tracking
  6. Fuzz Testing