Clase 4:
Common Weakness Enumeration (CWE)

Herson Esquivel Vargas

IC8071 Seguridad de software

TEC | Tecnológico de Costa Rica

Introduction

- CWE™ is a **community-developed** list of **software** and hardware **weakness types**. It serves as a common language, a measuring stick for security tools, and as a baseline for weakness identification, mitigation, and prevention efforts.

Reference: https://cwe.mitre.org/

Introduction

- Organized as a hierarchy:
- **Category**: contains a set of entries that share a common characteristic
  - **Base**: a weakness that is still mostly independent of a resource or technology, but with sufficient details to provide specific methods for detection and prevention
    - **Variant**: a weakness that is linked to a certain type of product, typically involving a specific language or technology.
- Numeric ids, descriptions, examples, etc.

Reference: https://cwe.mitre.org/

Common Weakness Enumeration (CWE)

- **Weakness** definitions:
  - Poor coding practices, as exemplified by CWEs.[1]
  - A weakness is a condition in a software, firmware, hardware, or service component that, under certain circumstances, could contribute to the introduction of vulnerabilities.[2]
  - Not even mentioned in RFC 4949 (Internet Security Glossary, Version 2).[3]

Reference: [1]https://csrc.nist.gov/glossary/term/weakness
[2]https://capec.mitre.org/about/glossary.html
[3]https://www.rfc-editor.org/rfc/rfc4949

Common Weakness Enumeration (CWE)

- Why a **weakness** taxonomy?
  - "High quality **tools and services** for finding security weaknesses in code are **maturing** but still address only a **portion** of the suspect areas…"
  - "...Without such a **common description**, these efforts cannot move forward in a meaningful fashion or be aligned and integrated with each other to provide **strategic value**."

Reference: Martin, R. A., & Barnum, S. (2008). Common weakness enumeration (CWE) status update. ACM SIGAda Ada Letters, 28(1), 88-91.

Common Weakness Enumeration (CWE)

- Who uses CWE?

Flawfinder

...and many many more.

Reference: Martin, R. A., & Barnum, S. (2008). Common weakness enumeration (CWE) status update. ACM SIGAda Ada Letters, 28(1), 88-91.

Common Weakness Enumeration (CWE) – Find it!

```
function generateSessionID($userID) {
    srand($userID);
    return rand();
}
```


Fuente: CWE-331: Insufficient Entropy - https://cwe.mitre.org/data/definitions/331.html

Common Weakness Enumeration (CWE) – Find it!

```
public int averageResponseTime
                (int totalTime, int numRequests) {
    return totalTime / numRequests;
}
```


Fuente: CWE-369: Divide By Zero - https://cwe.mitre.org/data/definitions/369.html

Common Weakness Enumeration (CWE) – Find it!

```
#define PATH_SIZE 60


char filename[PATH_SIZE];


for(i=0; i<=PATH_SIZE; i++) {
    char c = getc();
    if (c == EOF) {
        filename[i] = '\0';
            break;
    }
    filename[i] = c;
}
```


Fuente: CWE-193: Off-by-one Error - https://cwe.mitre.org/data/definitions/193.html

Common Weakness Enumeration (CWE) – Find it!

```
int isValid(int value) {
    if (value = 100) {
        printf("Value is valid\n");
        return(1);
    }


    printf("Value is not valid\n");
    return(0);
}
```


Fuente: CWE-480: Use of Incorrect Operator - https://cwe.mitre.org/data/definitions/480.html

```java
public boolean isReorderNeeded(String bookISBN, int rateSold) {
    boolean isReorder = false;
    int minimumCount = 10;
    int days = 0;

    // get inventory count for book
    int inventoryCount = inventory.getIventoryCount(bookISBN);

    // find number of days until inventory count reaches minimum
    while (inventoryCount > minimumCount) {
        inventoryCount = inventoryCount - rateSold;
        days++;
    }

    // if number of days within reorder timeframe
    // set reorder return boolean to true
    if (days > 0 && days < 5) {
        isReorder = true;
    }

    return isReorder;
}
```

Fuente: CWE-835: Loop with Unreachable Exit Condition ('Infinite Loop') - https://cwe.mitre.org/data/definitions/835.html

Common Weakness Enumeration (CWE) – Find it!

```
int returnChunkSize(void *) {
    /* if chunk info is valid, return the size of usable memory,
    * else, return -1 to indicate an error
    */
    ...
}
int main() {
    ...
    memcpy(destBuf, srcBuf, (returnChunkSize(destBuf)-1));
    ...
}
```


Fuente: CWE-252: Unchecked Return Value -https://cwe.mitre.org/data/definitions/252.html

Common Weakness Enumeration (CWE) – Find it!

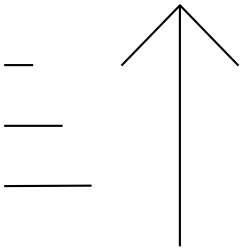
```
char str[20];
strcat(str, "hello world");
printf("%s", str);
```


Fuente: CWE-908: Use of Uninitialized Resource -https://cwe.mitre.org/data/definitions/908.html

Common Weakness Scoring System (CWSS™)

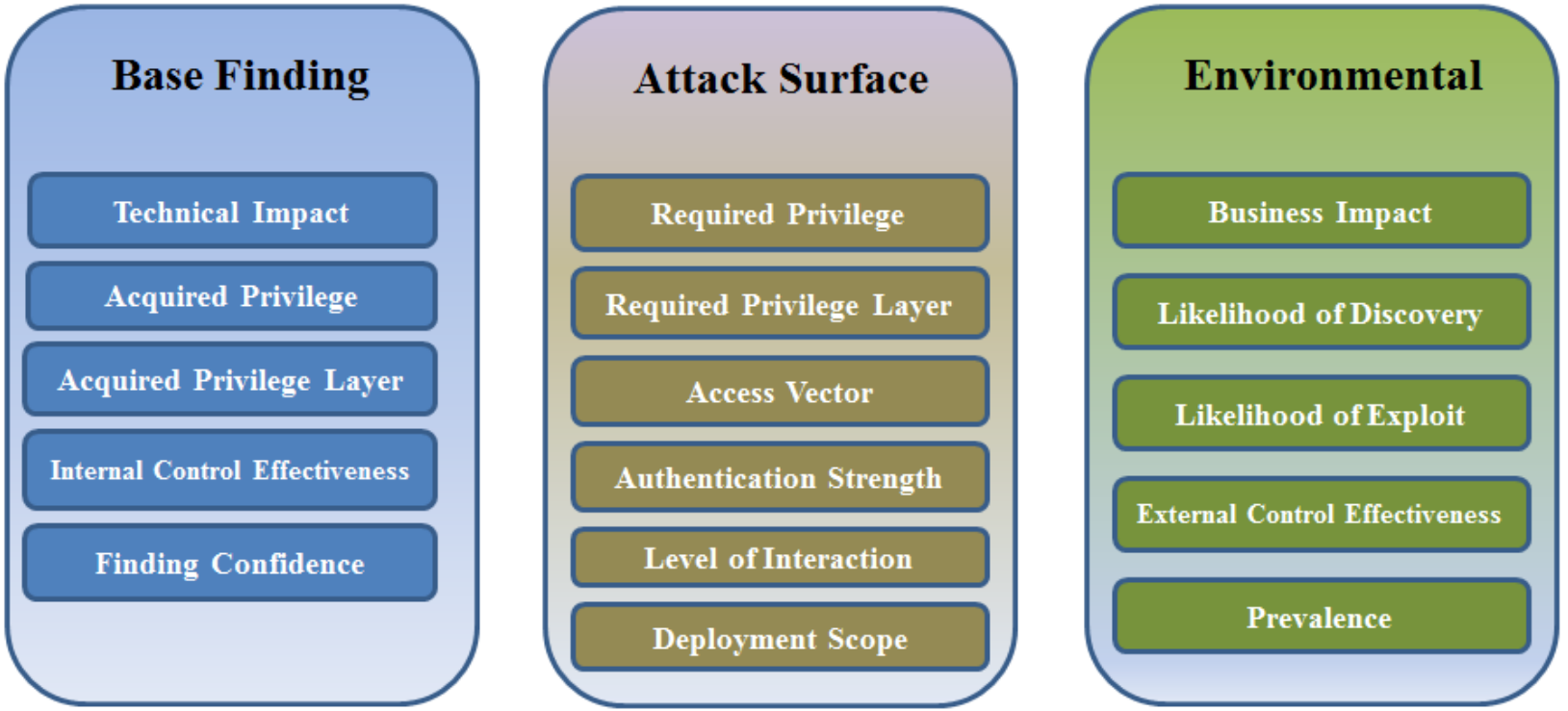
- Shared framework
  - All weaknesses are evaluated using standardized criteria

- Quantitative score
  - Allows comparisons and prioritization



Reference: https://cwe.mitre.org/cwss/cwss_v1.0.1.html

Common Weakness Scoring System (CWSS™)



Reference: https://cwe.mitre.org/cwss/cwss_v1.0.1.html

Common Weakness Scoring System (CWSS™)

- **Example:** Base Finding → Acquired Privilege

Value	Code*	Weight	Description
Administrator	A	1.0	The attacker gains access to an entity with administrator, root, SYSTEM, or equivalent privileges that imply full control over the software under analysis; or, the attacker can raise their own (lower) privileges to an administrator.
Partially-Privileged User	P	0.9	The attacker gains access to an entity with some special privileges, but not enough privileges that are equivalent to an administrator; or, the attacker can raise their own (lower) privileges to a partially-privileged user. For example, a user might have privileges to make backups, but not to modify the software's configuration or install updates.
Regular User	RU	0.7	The attacker gains access to an entity that is a regular user who has no special privileges; or, the attacker can raise their own (lower) privileges to that of a regular user.
Limited / Guest	L	0.6	The attacker gains access to an entity with limited or "guest" privileges that can significantly restrict allowable activities; or, the attacker can raise their own (lower) privileges to a guest. Note: this value does not refer to the "guest operating system" concept in virtualized hosts.
None	N	0.1	The attacker cannot gain access to any extra privileges beyond those that are already available to the attacker. (Note that this value can be useful in limited circumstances in which the attacker can escape a sandbox or other restrictive environment but still cannot gain extra privileges, or obtain access as other users.)
Default	D	0.7	Median of the weights for None, Guest, Regular User, Partially-Privileged User, and Administrator.
Unknown	UK	0.5	There is not enough information to provide a value for this factor. Further analysis may be necessary. In the future, a different value might be chosen, which could affect the score.
Not Applicable	NA	1.0	This factor is being intentionally ignored in the score calculation because it is not relevant to how the scorer prioritizes weaknesses. This factor might not be applicable in an environment with high assurance requirements that wants strict enforcement of privilege separation, even between already-privileged users.
Quantified	Q		This factor could be quantified with custom weights. Note that Quantified values are supported for completeness; however, since privileges and users are discrete entities, there might be limited circumstances in which a quantified model would be useful.

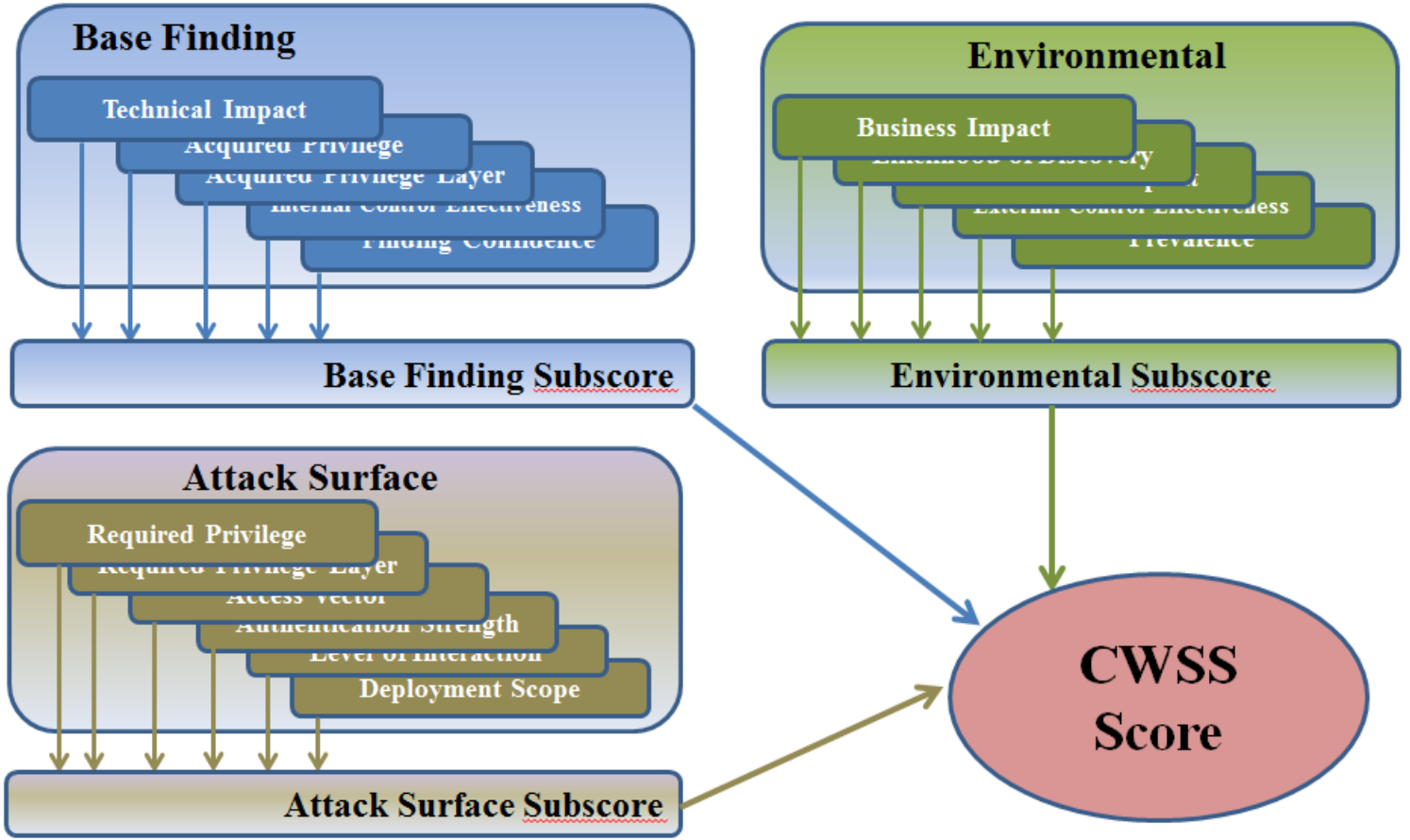
Common Weakness Scoring System (CWSS™)

- **Example:** Base Finding → Technical Impact

Value	Code	Weight	Description
Critical	C	1.0	Complete control over the software being analyzed, to the point where operations cannot take place.
High	H	0.9	Significant control over the software being analyzed, or access to critical information can be obtained.
Medium	M	0.6	Moderate control over the software being analyzed, or access to moderately important information can be obtained.
Low	L	0.3	Minimal control over the software being analyzed, or only access to relatively unimportant information can be obtained.
None	N	0.0	There is no technical impact to the software being analyzed at all. In other words, this does not lead to a vulnerability.
Default	D	0.6	The Default weight is the median of the weights for Critical, High, Medium, Low, and None.
Unknown	UK	0.5	There is not enough information to provide a value for this factor. Further analysis may be necessary. In the future, a different value might be chosen, which could affect the score.
Not Applicable	NA	1.0	This factor is being intentionally ignored in the score calculation because it is not relevant to how the scorer prioritizes weaknesses. This factor might not be applicable in an environment with high assurance requirements; the user might want to investigate every weakness finding of interest, regardless of confidence.
Quantified	Q		This factor could be quantified with custom weights.

Reference: https://cwe.mitre.org/cwss/cwss_v1.0.1.html

Common Weakness Scoring System (CWSS™)



Reference: https://cwe.mitre.org/cwss/cwss_v1.0.1.html

Common Weakness Scoring System (CWSS™)

$$Base = [\ (10 * TechnicalImpact + 5*(AcquiredPrivilege + AcquiredPrivilegeLayer) + 5*FindingConfidence) * \mathbf{f}(TechnicalImpact) * InternalControlEffectiveness\ ] * 4.0$$

$\mathbf{f}(TechnicalImpact) = 0$ if $TechnicalImpact = 0$; otherwise $\mathbf{f}(TechnicalImpact) = 1$.

Reference: https://cwe.mitre.org/cwss/cwss_v1.0.1.html

Common Weakness Scoring System (CWSS™)

$$\text{Attack} = [\ 20*(\text{RequiredPrivilege} + \text{RequiredPrivilegeLayer} + \text{AccessVector}) + 20*\text{DeploymentScope} + 15*\text{LevelOfInteraction} + 5*\text{AuthenticationStrength}\ ] \ /\ 100.0$$

Reference: https://cwe.mitre.org/cwss/cwss_v1.0.1.html

Common Weakness Scoring System (CWSS™)


```
Environmental = [ (10*BusinessImpact +
      3*LikelihoodOfDiscovery +
4*LikelihoodOfExploit + 3*Prevalence) *
          f(BusinessImpact) *
 ExternalControlEffectiveness ] / 20.0

f(BusinessImpact) = 0 if BusinessImpact == 0;
      otherwise f(BusinessImpact) = 1
```


Reference: https://cwe.mitre.org/cwss/cwss_v1.0.1.html

Common Weakness Scoring System (CWSS™)

- Visit https://cwss-score.info and fill-in the form

Consider a reported weakness in which an application is the primary source of income for a company, thus has critical business value. The application allows arbitrary Internet users to sign up for an account using only an email address. A user can then exploit the weakness to obtain administrator privileges for the application, but the attack cannot succeed until the administrator views a report of recent user activities - a common occurrence. The attacker cannot take complete control over the application, but can delete its users and data. Suppose further that there are no controls to prevent the weakness, but the fix for the issue is simple, and limited to a few lines of code,

Reference: https://cwe.mitre.org/cwss/cwss_v1.0.1.html

Summary

- _____
- _____
- _____

Optional readings

- Martin, R. A., & Barnum, S. (2008). Common weakness enumeration (CWE) status update. ACM SIGAda Ada Letters, 28(1), 88-91.