# Quantitative Cyber-Security

**Colorado State University**
**Yashwant K Malaiya**

**CS559**
**L16: CVSS & Testing**



**CSU Cybersecurity Center**
**Computer Science Dept**

# Leaves are falling..



Colorado State University

# Quantitative Security

**Colorado State University**
**Yashwant K Malaiya**

**CS 559**
**CVSS**



**CSU Cybersecurity Center**
**Computer Science Dept**

# Topic Outline

CVSS Metrics/Scores Outline

- Introduction/history
- The Base Score
  - The Base metrics and their values
- Temporal Score
- Environmental Score
- Question of validation



4

# CVSS: Common Vulnerability Scoring System

- How important is a specific vulnerability?
  - Essentially a risk measure
  - Vulnerabilities with highest scores need addressing quickly. Those with lowest scores are low priority.
- CVSS v1: *National Infrastructure Advisory Council* (NIAC), 2005
- CVSS v2: *Forum of Incident Response and Security Teams* (FIRST)
  - 2007
  - Still found
- CVSS V3: 2015
  - Common
- CVSS V4: Nov 2023
  - Recently introduced

# CVSS Scoring System

- How important is a specific vulnerability?
  - Essentially a risk measure*
  - Vulnerabilities with highest scores need addressing quickly. Those with lowest scores are low priority.
- CVSS v1: *National Infrastructure Advisory Council* (NIAC), 2005
- CVSS v2: *Forum of Incident Response and Security Teams* (FIRST)
  - 2007
  - Still common
- CVSS V3: 2015
- CVSS V4: Nov 2023 coming into use

* As we will see


6

# CVSS Metrics and Scores

De facto standard for assessing severity of software vulnerabilities

- Objective: prioritize effort to address vulnerabilities
- Metrics: Components by levels, each translated into a numerical metric (detailed tables soon)
- Scores: Computed score using a set of metrics as given by formulas
  - Formulas based on committee judgement, not derived or proven
- Three metric groups and associated scores;
  - Base (mandatory): intrinsic to the vulnerability
  - Thread ~~Temporal~~: time-dependent variation in risk (renamed Threat metrics in V4)
  - Environmental: risk component dependent on the organization's environment
  - Supplemental (V4)

* As we will see


7

# CVSS Base Metrics & Score: Overview

CVSS Base Score = f(Exploitability sub-score, Impact sub-score,)

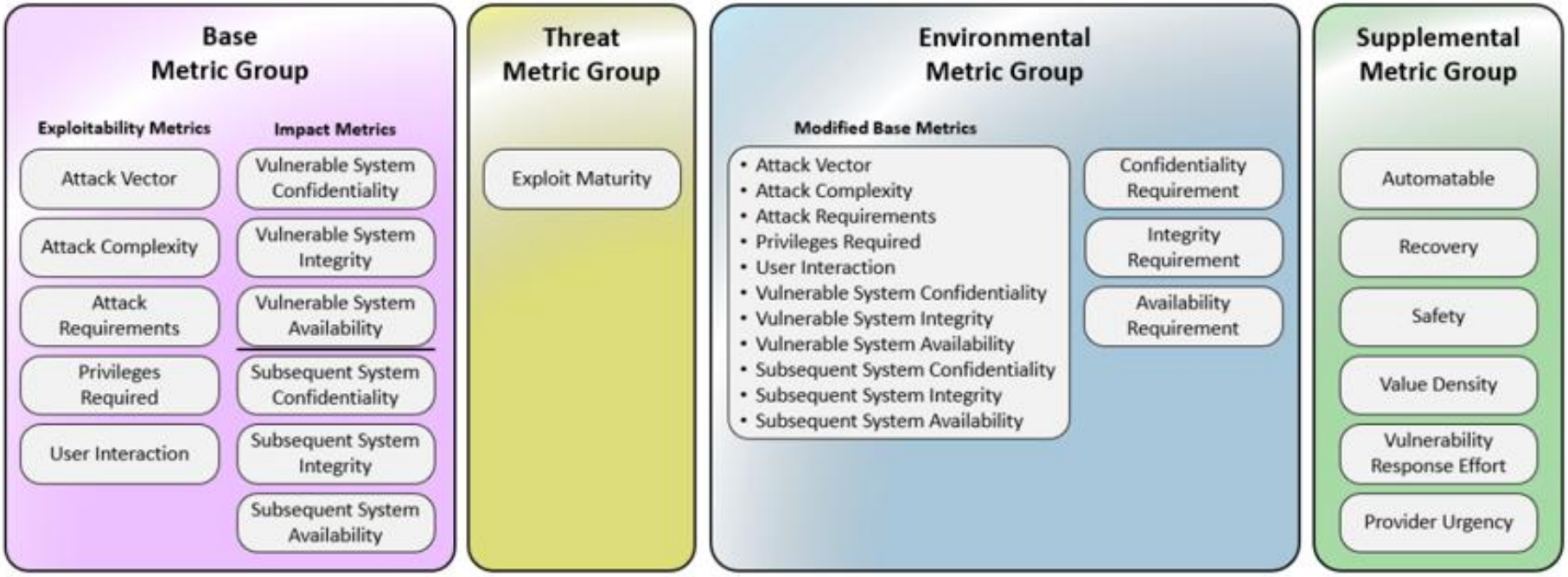
- This is the score used to prioritize vulnerabilities*.
- Ranges from 0 to 10.

- Exploitability metrics: how easy is the vulnerability to exploit
  - Exploitability sub-score = f(Attack Vector, Attack Complexity, Attack Requirements, Privileges Required, User Interaction~~, Scope~~)
- Impact sub-score: what is the extent of the impact.
  - Impact sub-score = f(Confidentiality, Integrity,, Availability, subsequent integrity and availability)
- Intrinsic value of a raw metric ranges from 0 to 1.
- Our observation: exploitability sub-score attempts to measure *likelihood*$_i$, Impact sub-score attempts to measure *impact*$_i$ due to a *vulnerability i.*

* Details to follow


8

# CVSS V4.0 Metrics




*CVSS Metric Groups*

FIRST: [Forum of Incident Response and Security Teams]

883 Teams in nine countries

Colorado State University

# CVSS Base Scores: Ratings

All recognized vulnerabilities are assigned Base Scores*. The complete data for recognized vulnerabilities is available at the National Vulnerability Database.

Ranking according to CVSS V3/V4.

| Rating | CVSS Score | Response time |
|---|---|---|
| None | 0.0 | Next major release accept risk |
| Low | 0.1 - 3.9 | Next major release accept risk |
| Medium | 4.0 - 6.9 | 90-day SLA maintenance window |
| High | 7.0 - 8.9 | 30-day SLA planned sprint |
| Critical | 9.0 - 10.0 | 24-72 hours Emergency |

* Has this measure been validated? Interesting question.


10

# CVSS Metrics: Data bases

NVD (National Vulnerability database): U.S. government repository of standards based vulnerability management data. represented using the Security Content Automation Protocol (SCAP).

- sponsored by the Department of Homeland Security (DHS), NCCIC and US-CERT.

## Documentation/Data hosted at NVD

Example: MySQL Stored SQL Injection (CVE-2013-0375)

- Oracle MySQL 5.1.66 and earlier, and 5.1.28 and earlier
- CVSS v3.0 Base Score: 5.4 medium
  - Vector: CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:L/I:L/A:N
- CVSS v4.0 Base Score: not yet assessed

**CVSS v3.1 Severity and Metrics:**
**Base Score:** 5.4 MEDIUM
**Vector:** AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N
**Impact Score:** 2.5
**Exploitability Score:** 2.8

**Attack Vector (AV):** Network
**Attack Complexity (AC):** Low
**Privileges Required (PR):** Low
**User Interaction (UI):** None
**Scope (S):** Unchanged
**Confidentiality (C):** Low
**Integrity (I):** Low
**Availability (A):** None

* Has this measure been validated? Interesting questions.


11

# CVE Statuses in NVD

NVD analyzes each vulnerability received. Possible status at a time:

- **Received**: CVE has been recently published to the CVE dictionary and has been received by the NVD.
- **Awaiting Analysis**: about 24 hours
- **Undergoing Analysis**: CVE is currently being analyzed by NVD staff, results in association of reference link tags, CVSS scores
- **Analyzed**
- **Modified**: CVE has been amended by a source (CVE Primary CNA or another CNA)
- **Deferred**
- **Rejected**: These CVEs are in the NVD, but currently do not show up in search results.

*


12

# Examining CVSS V.3

- CVSS V3.1 is still widely used, CVSS v4.0 is only used by a small fraction of the industry.
  - Database and Metric Incompatibility: CVSS v4.0 completely restructures the scoring string (removing the "Scope" metric and adding subsequent impacts and supplemental metrics).
  - Hard to compute v4.0 scores. Only 25.9% of published CVEs include a CVSS v4.0 score!
  - Formulas have been replaced by a complex procedure:
    - The 15 million possible metric combinations are grouped into 270 equivalence sets called *MacroVectors*. Experts analyzed these sets to establish baseline severity boundaries and scores mapped out from 0.0 to 10.0.
- We thus examine CVSS V3 is detail.

# CVSS V.3 Details: Base Metrics 1/4

## Exploitability components:

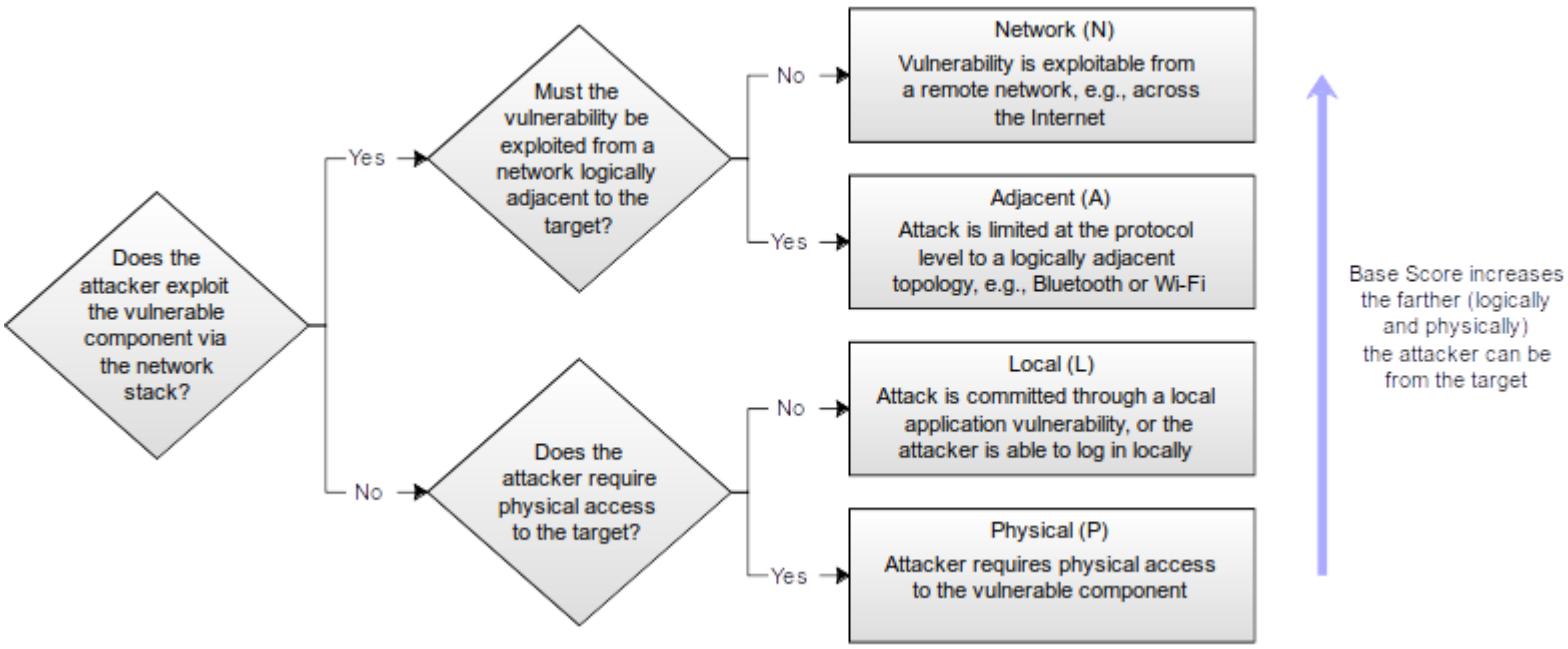
How did they come up with these numbers?

| Attack Vector (AV) | context by which vulnerability exploitation is possible | |
|---|---|---|
| **Metric level** | **Represents** | **Value V3.0** |
| Network (N) | "remotely exploitable" one or more network hops away | 0.85 (highest risk) |
| Adjacent (A) | attack is limited to the same shared physical/logical network | 0.62 |
| Local (L) | exploitable with local access | 0.55 |
| Physical (P) | requires the attacker to physically access vulnerable component | 0.2 |

| Attack Complexity (AC) | conditions beyond the attacker's control that must exist in order to exploit the vulnerability | |
|---|---|---|
| **Metric level** | **Represents** | **Value V3.0** |
| Low | No specialized access conditions required | 0.77 |
| High | A successful attack depends on specific conditions beyond the attacker's control. | 0.44 |


14

# CVSS V.3 Graphical visualization: Attack vector 2/4




For other flow-charts see: [CVSS User Guide]


15

# CVSS V.3 Details: Base Metrics 3/4

## Exploitability components (cont):

| **Privileges Required (PR)** | **the level of privileges an attacker must possess before successfully exploiting the vulnerability.** | |
|---|---|---|
| **Metric level** | **Represents** | **Value V3.0** |
| None (N) | attacker is does not require any access to settings or files to carry out an attack | 0.85 |
| Low (L) | The attacker needs basic user privileges for the attack | 0.62 (0.68 if scope/ modified scope is changed) |
| High (H) | attacker needs administrative privileges | .27 (0.50 if scope/ modified scope is changed) |

| **User Interaction (UI)** | **whether a separate user (or user-initiated process) must participate in some manner** | |
|---|---|---|
| **Metric level** | **Represents** | **Value V3.0** |
| None (N) | can be exploited without interaction from any user | 0.85 |
| Required (R) | requires a user to take some action before the vulnerability can be exploited | 0.62 |


16

# CVSS V.3 Details: Base Metrics 4/4

Scope change : When the vulnerability governed by one authorization scope is able to affect resources governed by another authorization scope.

| Scope (dropped V4) | impact to the confidentiality/Integrity/Availability of the information resources | |
|---|---|---|
| **Metric level** | **Represents** | **Value V3.0** |
| Unchanged (U) | An exploited vulnerability can only affect resources managed by the same authority. | - |
| Changed (C) | An exploited vulnerability can affect resources beyond the authorization privileges intended by the vulnerable component | Modified scope |

Impact : The Impact metrics refer to the properties of the impacted component.

| Confidentiality Impact (C)<br>Integrity Impact (I)<br>Availability Impact (A) | impact to the confidentiality/Integrity/Availability of the information resources | |
|---|---|---|
| **Metric level** | **Represents** | **Value V3.0** |
| High (H) | Total loss | 0.56 |
| Low (L) | loss is constrained | 0.22 |
| None (N) | no loss | 0 |

V4 adds additional metrics to Impact.



17

# CVSS V.3 Base Score: Formulas

Exploitability sub-score

- Exploitability = $8.22 \times Attack\ Vector \times Attack\ Complexity \times Privilege\ \ Required \times User\ Interaction$
- Impact sub-score (ISC)

Impact sub-score= $Scope\ Unchanged\ 6.42 \times ISC_{\text{Base}}$

$= Scope\ Changed\ 7.52 \times [ISC_{Base} - 0.029] - 3.25 \times [ISC_{Base} - 0.02]^{15}$

where $ISCBase$= 1−[ (1−$ImpactConf$)×(1− $ImpactInteg$)×(1− $ImpactAvail$)]

Base Score is

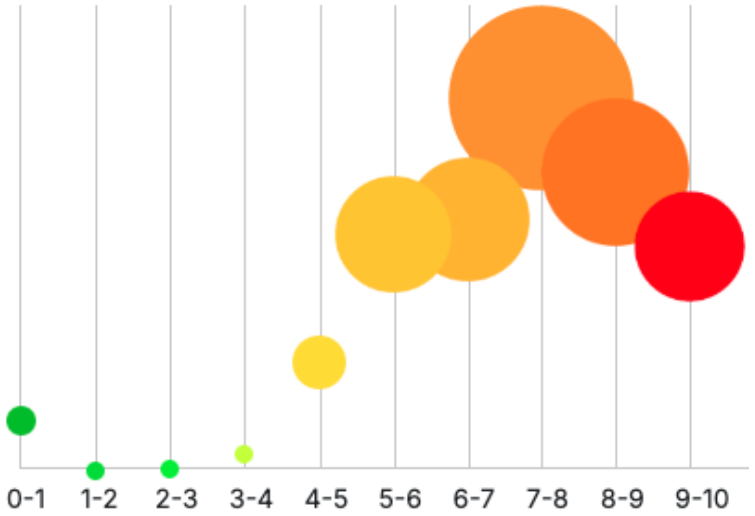
- If (Impact sub score =< 0)
  = 0 else,
- Scope Unchanged
  = $Round\ up$ ($Minimum$ [($Impact+Exploitability$),10])
- Scope Changed
  = $Round\ up$ ($Minimum$ [1.08 ×($Impact+Exploitability$),10])

All the subjects and objects under the jurisdiction of a single *security authority* are considered to be under one *security scope*. If a vulnerability in a vulnerable component can affect a component which is in a different *security scope* than the vulnerable component, a Scope change occurs

# CVSS V.3 Base Score: Distribution

| CVSS Score Range | Vulnerabilities |
|---|---|
| 0-1 | 3236 |
| 1-2 | 86 |
| 2-3 | 777 |
| 3-4 | 1600 |
| 4-5 | 5923 |
| 5-6 | 13024 |
| 6-7 | 13831 |
| 7-8 | 20612 |
| 8-9 | 16534 |
| 9+ | 12371 |
| Total | 87994 |

Weighted Average CVSS Score: 7.4




CVSS V.3 score distribution for CVEs published between 2025-10-09 and 2026-10-09

CVEDetails



19

# CVSS V.3 Temporal Metrics 1

Temporal metrics measure the current state: exploit availability, patches or workarounds, or the confidence in the description of the vulnerability.

Note: An exploit is a piece of code that exploits the vulnerability. Available exploits are documented at the Exploit databse.

*My Comment: Good idea, but the concept needs some work.*

| **Exploit Code Maturity (E)** | **context by which vulnerability exploitation is possible** | |
|---|---|---|
| Metric | **Represents** | **Value V3.0** |
| Not Defined (X) | No information available | 1 |
| High (H) | widely available exploit code works in every situation | 1 |
| Functional (F) | exploit code works in most situations | 0.97 |
| Proof-of-Concept (P) | Proof-of-concept exploit available but an attack demonstration is not practical for most systems | 0.94 |
| Unproven (U) | No exploit code is available, or an exploit is theoretical | 0.91 Lowest risk |


20

# Temporal Metrics 2

## Remediation Level (RL): patched yet?

| Remediation Level (RL) | context by which vulnerability exploitation is possible | |
|---|---|---|
| Not Defined (X) | No information available | 1 |
| Unavailable (U) | no solution available | 1 |
| Workaround (W) | an unofficial, non-vendor solution available | 0.97 |
| Temporary Fix (T) | official but temporary fix available | 0.96 |
| Official Fix (O) | official patch, or an upgrade is available | 0.95 |

## Report Confidence:

| Report Confidence (RC) | the degree of confidence in the existence of the vulnerability | |
|---|---|---|
| Not Defined (x) | No information available | 1 |
| Confirmed (C) | Detailed reports exist | 1 |
| Reasonable (R) | researchers do not have full confidence in the root cause | 0.96 |
| Unknown (U) | questionnable report | 0.92 |


21

# Temporal score and comments

Temporal score

$$= Round\ up(Base\ Score \times Exploit\ Code\ Maturity \times Remediation\ Level \times Report\ Confidence)$$

Comments*: The significance of the presence of an Exploit and a Patch is obvious. It is not clear by all the values are close to 1.0, i.e. the levels did not appear to matter for the FIRST people.*

*Note that*

- Most vulnerabilities are declared along with patch releases. However users often do not apply the patches immediately.
- Known exploits often appear quickly, if the vulnerability is significant.
  - *However we have found a lack of correlation between vulnerability severity and delay in exploit appearance. It may be explained that a high severity vulnerability may already have disappeared by in new release when it is disclosed, making exploit development largely useless.*

*V4: Terms this Threat, retains only Exploit maturity metric in it)*


22

# Environmental Metrics

The environmental score

= If (Modified Impact Sub score =< 0) 0

else,

Modified Scope Unchanged

= $Round\ up(Minimum\ [(M.Impact+M.Exploitability)\times$ $Exploit\ Code\ Maturity\times Remediation\ Level\times$ $Report\ Confidence$,10])

Modified Scope Changed

= $Round\ up(\ Round\ up[Minimum[1.08\times(M.Impact+$ $M.Exploitability)],10]\times$ $Exploit\ Code\ Maturity\times Remediation\ Level\times$ $Report\ Confidence))$

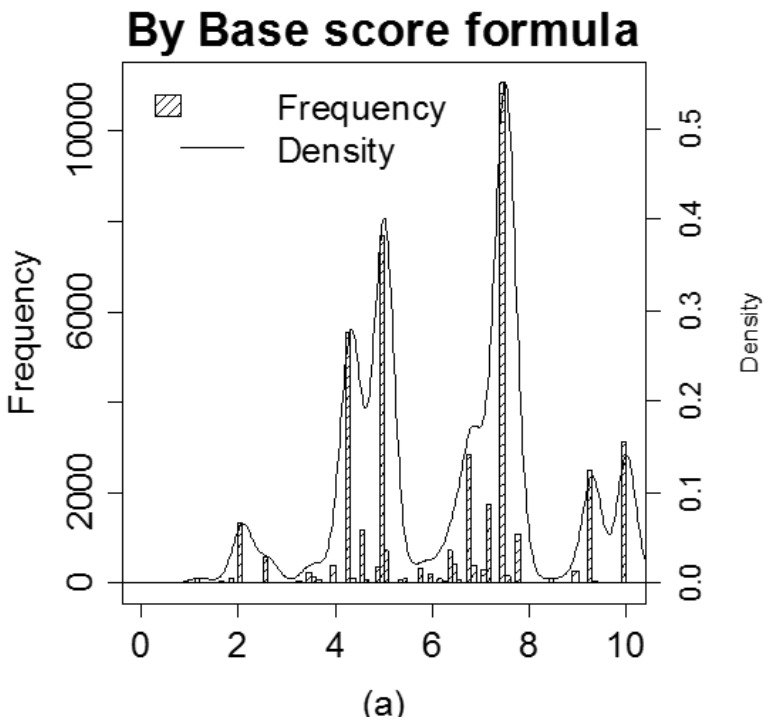
V4: drops Scope. Adds additional metrics

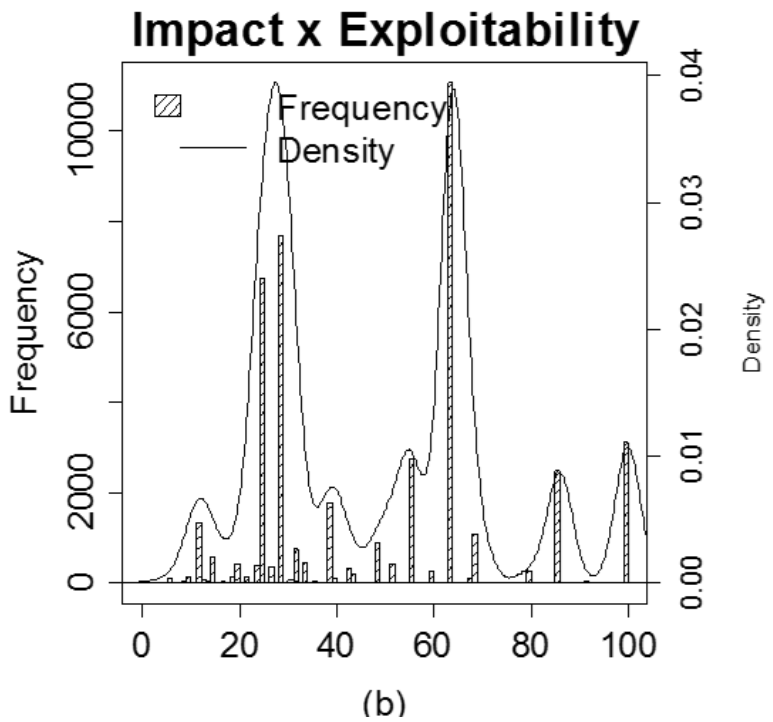


23

# CVSS: How useful it is?

# Distribution of Base score




| | Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. | Combinations |
|---|---|---|---|---|---|---|---|
| (a) | 0 | 5 | 6.8 | 6.341 | 7.5 | 10 | 63 |
| (b) | 0 | 29 | 49 | 48.59 | 64 | 100 | 112 |

H. Joh and Y. K. Malaiya, "Defining and Assessing Quantitative Security Risk Measures Using Vulnerability Lifecycle and CVSS Metrics," SAM'11, The 2011 International Conference on Security and Management, pp.10-16, 2011



Colorado State University

# Has CVSS worked?

- Windows 7 Correlation among
  - CVSS Exploitability
  - Microsoft Exploitability metric
  - Presence of actual exploits Perhaps these is a reason ..
- No significant correlation found.
- Continuing research

Correlation

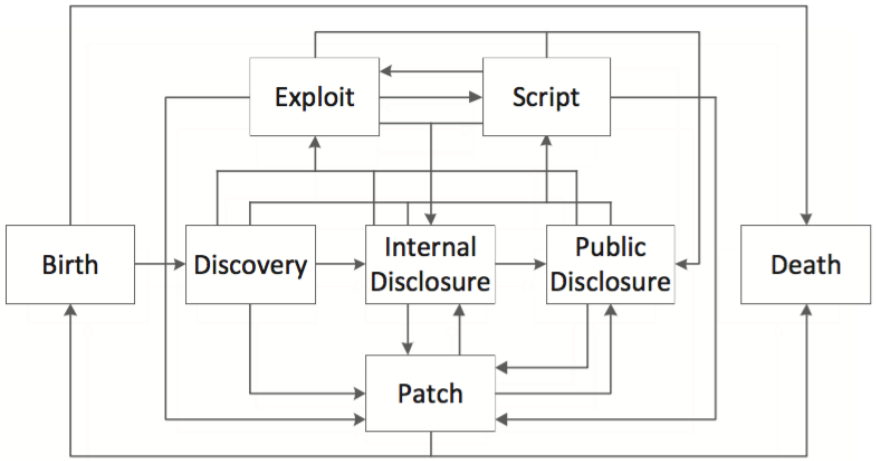
| Variables | Exploit Existence | MS-EXP | CVSS-EXP |
|---|---|---|---|
| Exploit Existence | 1 | -0.078 | -0.146 |
| MS-EXP | **-0.078** | 1 | -0.116 |
| CVSS-EXP | **-0.146** | -0.116 | 1 |

A. Younis and Y.K. Malaiya, "Comparing and Evaluating CVSS Base Metrics and Microsoft Rating System", The 2015 IEEE Int. Conf. on Software Quality, Reliability and Security, pp. 252-261

Colorado State University

# Likelihood of Individual Vulnerabilities Discovery

- **Ease of discovery**
  - Human factor (skills, time, effort, etc.), Discovery technique, Time
- Time:




Example
- Apache HTTP server
- CVE-2012-0031, **(01/18/2012)**
- V. 1.3.0→1998-06-06

- Time to discover: 13 years!

**Time to Discovery =** Discovery Time Date – First Effected version Release Date



Colorado State University

27

# Correlation: Access complexity vs Time to Discover

## ❖ Access complexity vs Time

- **AC= Low**

**2.000**

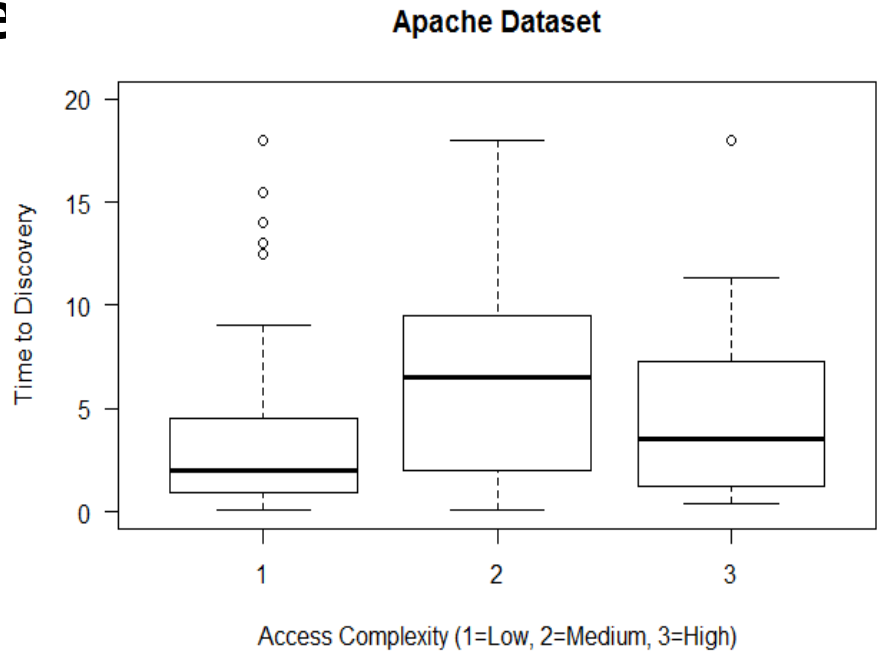
- **AC= Medium**

**6.500**

- **AC= High (very few points)**

**3.500**

- **Observation: Some correlation between Access Complexity and Time to Discover**

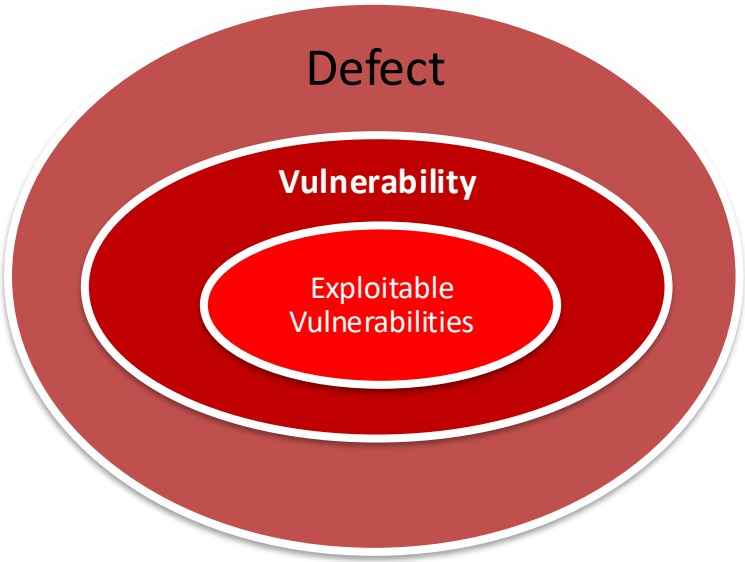





28

# Characterizing Vulnerability with Exploits

- **1 to 5 %** of defects are vulnerabilities.
- Finding vulnerabilities can take considerable expertise and effort.
- Out of 49599 vulnerabilities reported by NVD, only **2.10% have an exploit**.
- A vulnerability with an exploit written for it presents more risk.
- **What characterizes a vulnerability having an exploit?**
  - Observation: Exploiters like smaller modules?




| Vulnerability | In-Degree | Out-Degree | CountPath | ND | CYC | Fan-In | No of Invocation | SLOC | Exploit Existence |
|---|---|---|---|---|---|---|---|---|---|
| **CVE-2009-1891** | 1 | 9 | 9000 | 6 | 68 | 45 | 2 | 211 | NEE |
| **CVE-2010-0010** | 4 | 9 | 145 | 4 | 11 | 16 | 4 | 38 | EE |
| **CVE-2013-1896** | 26 | 5 | 8 | 1 | 5 | 37 | 3 | 29 | EE |


29

# Vulnerability with Exploits

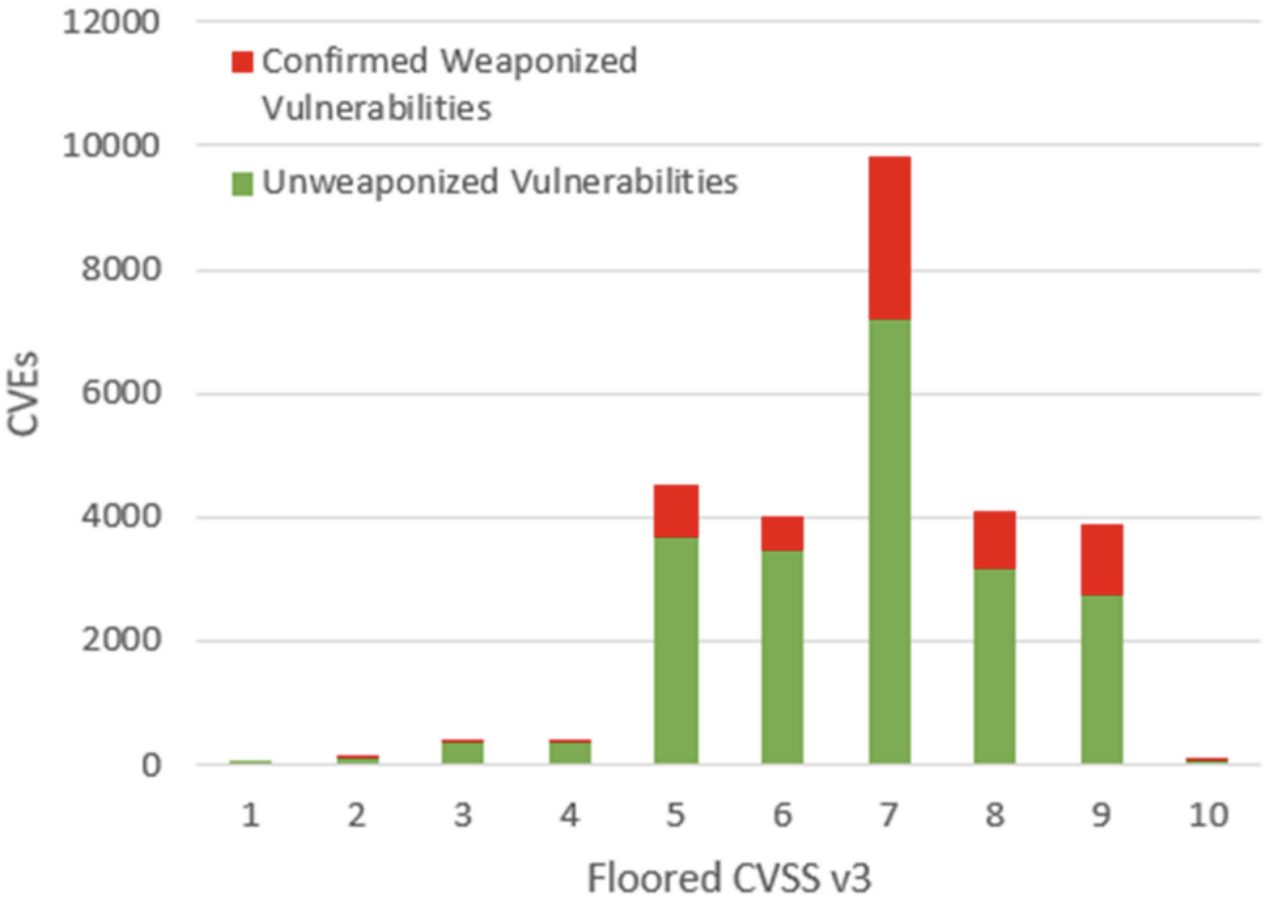



Fig. 2. Weaponized vs Unweaponized CVEs by CVSS score.

Henry Howland. 2023. CVSS: Ubiquitous and Broken. Digit. Threat.: Res. Pract. 4, 1, Article 1 (March 2023)

Colorado State University

# CVSS Base Score vs Vulnerability Rewards Programs

- We examined 1559 vulnerabilities of Mozilla Firefox and Google Chrome browsers for which records were available.
- Looked at Mozilla and Google vulnerability reward programs (VRPs) records for those vulnerabilities.

| Firefox | | |
|---|---|---|
| Vulnerabilities | Rewarded | Not rewarded |
| 547 | 225 | 322 |
| VRP severity | Rewarded | Not rewarded |
| Critical & High | 210 | 202 |
| Medium | 15 | 89 |
| Low | 0 | 31 |

A. Younis, Y. Malaiya and I. Ray, "Evaluating CVSS Base Score Using Vulnerability Rewards Programs", Proc. 31th Int. Information Security and Privacy Conference, IFIP SEC, Ghent, Belgium, 2016, pp. 62-75



Colorado State University

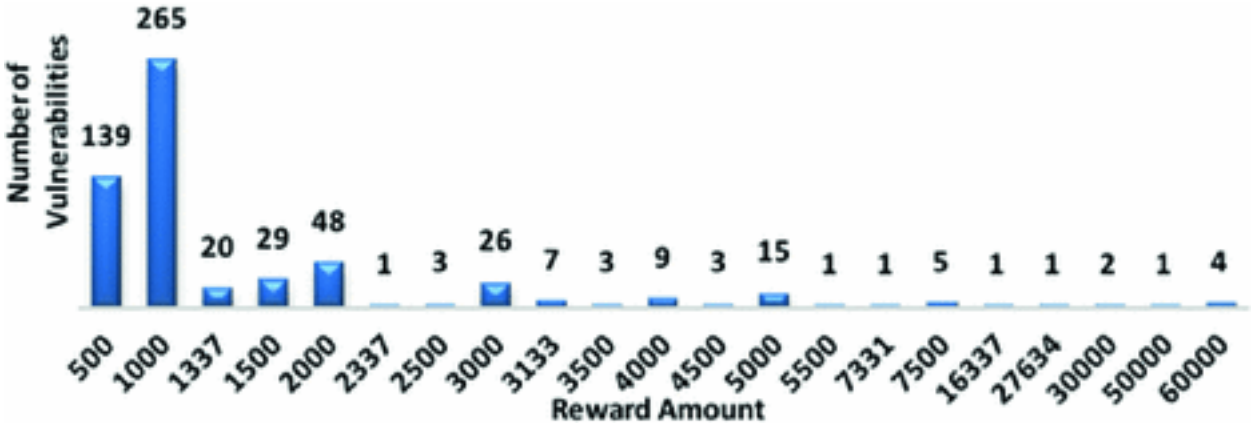
# CVSS Base Score vs Vulnerability Rewards Programs

| Chrome | | |
|---|---|---|
| Vulnerabilities | Rewarded | Not rewarded |
| 1012 | 584 | 428 |
| VRP severity | Rewarded | Not rewarded |
| Critical & High | 441 | 175 |
| Medium | 136 | 137 |
| Low | 7 | 116 |

The results results show that CVSS Base Score may have some correlation with the vulnerability reward program.

# How much did Chrome pay?

- Incidental result

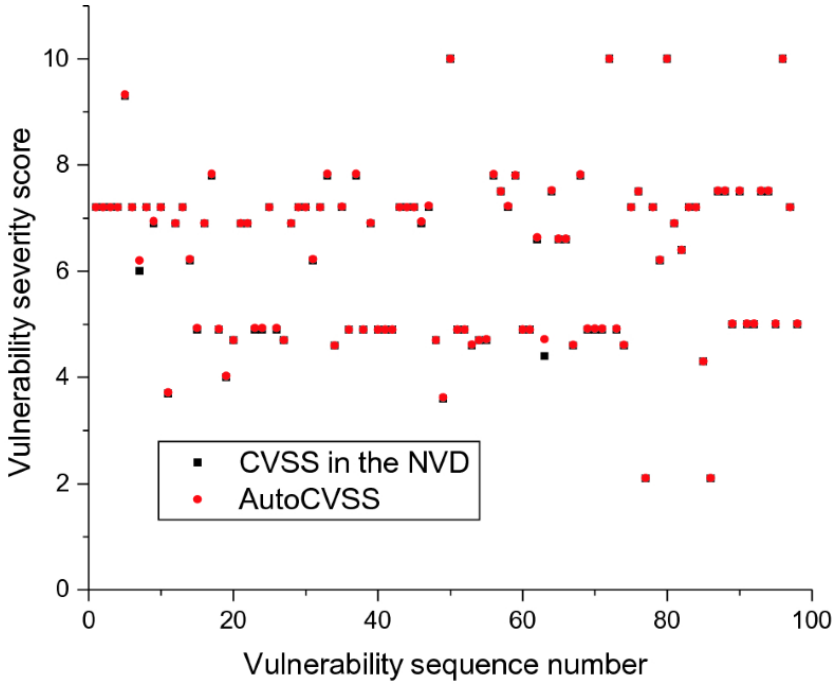





Colorado State University

# AutoCVSS?

- Zou et al. 2019: 98 vulnerabilities from Linux kernel, FTP service, and Apache service with their exploits from exploit-db.
- CVSS relies on human experts to determine metric values during the process of vulnerability severity assessment. They have attempted to automate the process.
- Result is that only two vulnerability severity scores assessed by AutoCVSS are obviously different from those in the NVD for CVSS v2.



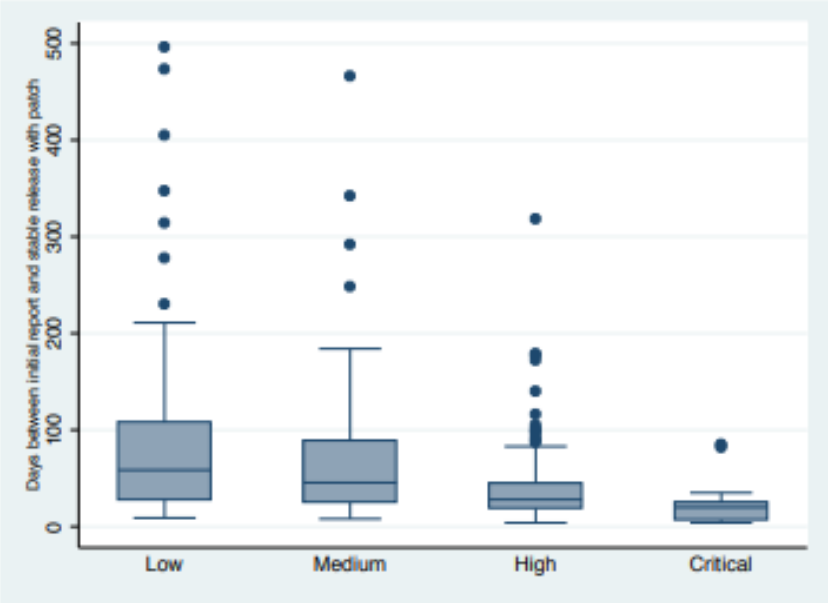

D. Zou, J. Yang, Z. Li, X. Ma, , AutoCVSS: An Approach for Automatic Assessment of Vulnerability Severity Based on Attack Process, Int. Conf. on Green, Pervasive, and Cloud Computing, April 2019

# VRP Cost effectiveness

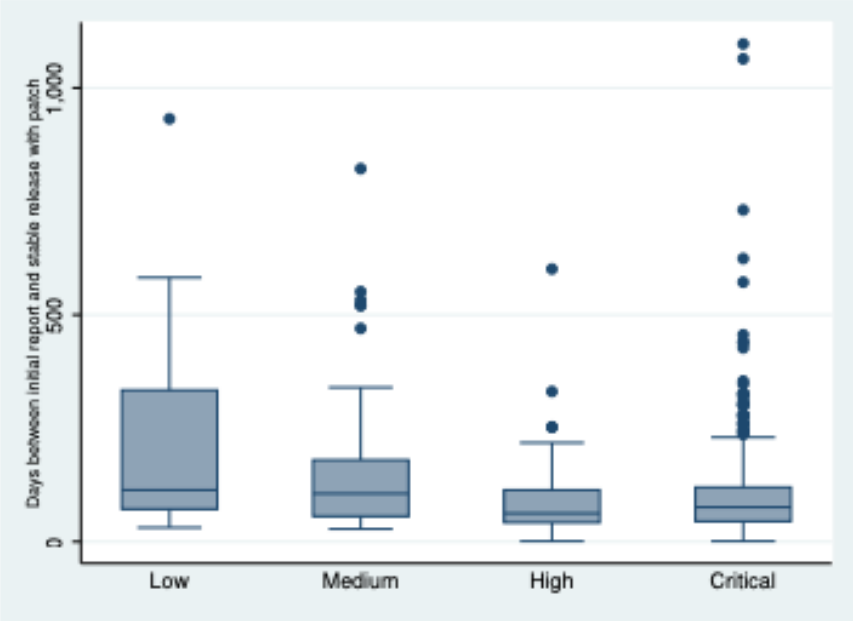
- Hypothesis: A VRP can be a cost-effective mechanism for finding security vulnerabilities.
  - Period studied: 7/09-1/13
  - Chrome's VRP has cost $485 per day on average, and that of Firefox has cost $658 per day.
  - Average North American developer on a browser security team (i.e., that of Chrome or Firefox) would cost around $500 per day (assuming a $100,000 salary with a 50% overhead).
- Hypothesis: Contributing to a single VRP is, in general, not a viable full-time job, though contributing to multiple VRPs may be, especially for unusually successful vulnerability researchers.
- Hypothesis: Successful independent security researchers bubble to the top, where a full-time job awaits them

M. Finifter, D. Akhawe, D. Wagner, An empirical study of vulnerability rewards programs. In USENIX Security Symposium 2013 (2013), pp. 273-288

# Time to patch




*(a)* Chrome




*(b)* Firefox

M. Finifter, D. Akhawe, D. Wagner, An empirical study of vulnerability rewards programs. In USENIX Security Symposium 2013 (2013), pp. 273-288

# CVSS: Review

- Developed to assess severity levels of vulnerabilities.
- V3: **L**: 0.1-, **M**:4.0-, **H**: 7.0-, **Crit**: 9.0-10.0

Formulas:

- Base Score = f( Impact, Exploitability)
- Sub-scores Exploitability and Impact are computed using Base Metrics, that depend on the vulnerability itself.
- Temporal Score = f(Base score, exploit, patch)
- Environmental score = f(CIA requirements, developments)

# CVSS: How useful it is?

- What if they had multiplied exploitability and impact sub-scores instead of adding?
- Correlation among
  - CVSS Exploitability
  - Microsoft Exploitability metric
  - Presence of actual exploits
- Time to discovery?
- Reward program?
- Can metric/score determination be automated?

# Course notes

- HW2a due 10/16/2026
- HW2b Should be available soon. Involves SRGMs.
  - Due Nov 4..
- Midterm Nov 3 (4 for S801)

# Quantitative Security

**Colorado State University**

**Yashwant K Malaiya**

**CS 559**

**Testing**



**CSU CyberCenter**
***Course Funding Program* – 2019**

# Testing for bugs






41

# Faults

- Faults cause a system to respond in a way different from expected.
- Faults can be associated with bugs in the system/software structure or functionality.
  - Structure: viewed as an interconnection of components like statements, blocks, functions, modules.
  - Functionality: Described by the input/output/state behavior, described externally. Black-box view.
  - Both structure and functionality can be described at a higher level and a lower (finer) level.
    - Example: a file > classes > methods etc. > statements


42

# Testing

- Testing and debugging is an essential part of software development and maintenance. 15-75% cost
  - Static analysis: code inspection
  - Dynamic: involves execution
- Defects cause functionality/reliability and security problem.
- Vulnerabilities are a subset of the defects (1-5%)
  - If exploited, allow violation of security related assumptions.
  - Vulnerability discovery can involve testing with
    - Random tests (Fuzzing)
    - Generated tests based on security requirements
- The following discussion is general for all defects.

# Testing

- We assume that tests are applied at the inputs and the response is observed at the outputs of the unit-under-test.
- A test detects the presence of a fault(s), if the output is different from the expected output.
- Two test approaches:
  - Functional (or Black-box): uses only the functional description of the unit, not its structure to obtain tests. Often random ("fuzzing")
  - Structural testing: uses the structural information to generate tests. Requires more effort but can be more thorough.
  - Combined


44

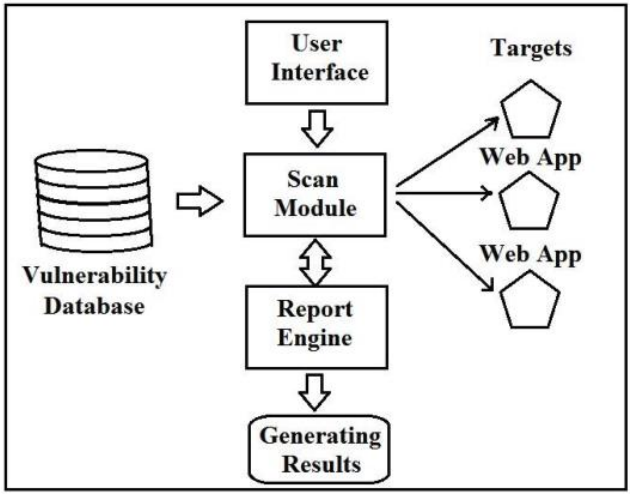
# Random vs Directed Testing

- Random testing: Termed Black-box, fuzzing when used for vulnerabilities
  - Random testing is a form of functional testing. In random testing, each test is chosen such that it does not depend on past tests.
  - In actual practice, the "random" tests are generated using Pseudo-random algorithms that approximate randomness.
  - As we will discuss later, random testing can be effective for moderate degree of testing, but not for thorough testing.
- Directed Testing: Specifically trying to cover the input space or structural components (statements, branches, modules etc.)


45

# Testing objectives

- Ordinary faults:
  - Fault detection: Apply a test input. Is the output what is expected? Triggering fault and propagating error.
  - Fault location: where is the fault?
  - Fixing: what will fix the fault? (debugging)
- Known vulnerabilities
  - use vulnerability reports/database for the product (Vulnerability scanners)
- Discovering unknown Vulnerabilities
  - Apply a slightly unexpected input. Does a program crash or hang?
  - If it does, examine it to see if it leads to a vulnerability.
  - Can the vulnerability be exploited?



# Partitioning

- Software can be partitioned to ensure that the software is thoroughly exercised during testing
- It is necessary to partition it to identify tests that would be effective for detecting the defects in different sections of the code.
- For testing purposes, a program may be partitioned either functionally or structurally.
- *Functional partitioning* refers to partitioning the input space of a program.
  - For example, if a program performs five separate operations, its input space can be partitioned into five partitions.
  - Functional partitioning only requires the knowledge of the functional description of the program, the actual implementation of the code is not required.
- *Structural partitioning* requires the knowledge of the structure at the code level.
  - If a software is composed of ten modules (which may be classes, functions or other types of units), it can be thought of as having ten partitions

Y. K. Malaiya, "Assessing Software Reliability Enhancement Achievable through Testing", Recent Advancements in Software Reliability Assurance 2019, pp. 107-138

# Sub-Partitioning

- A partition of either type can be subdivided into lower-level partitions, which may themselves be further partitionable at a lower level if higher resolution is needed (Elbaum 2001).
- Let us assume that a partition $p_i$ can be subdivided into sub-partitions $\{p_{i1}, p_{i2} \ldots p_{in}\}$.
  - Random testing within the partition $p_i$ will randomly select from $\{p_{i1}, p_{i2} \ldots p_{in}\}$. It is possible that some of them will get selected more often in a non-optimal manner.
  - Code within a sub-partition may be correlated relative to the probability of exercising some faults. Thus, the effectiveness of testing may be diluted if the same sub-partition frequently gets chosen.
  - Sub-partitioning has a practical disadvantage when the *operational profile* is constructed, it will require estimating the operational probabilities of the associated sub-partitions.

# Test coverage

- A single test typically covers (i.e., tests for related faults) several sub-partitions (elements such as functions, branches, statements)
- The coverage obtained by a test-set (test suite) can be obtained using coverage tools.
- The test coverage achieved by a test-set is given by ratio:

$$coverage = \frac{number\ of\ elements\ covered}{total\ number\ of\ elements}$$

- 100% coverage may be very hard or impossible to achieve in most practical cases.


49

# Coverage Tools

- There are several code coverage tools: Jcov, Gcov etc. for Java, C/C++ etc.
  - Compilation using the tool, instruments the compiled code to collect metrics covered.
  - Coverage metrics:
    - Statements/Blocks
    - Branches/Edges
    - Paths
    - Methods/Functions
    - Data-flow coverage metrics
  - Subsumption hierarchy
    - Complete Path coverage => 100% Branch coverage
    - Complete Branch coverage => 100% Statement coverage

Assumptions:
- A fault is associated with one or more elements.
- Exercising the element may trigger the fault to create an error
- Complete coverage does not guarantee finding all the faults.

# Input mix: Test Profile

- The inputs to a system can represent different types of operations. The input mix called "Profile" can impact effectiveness of testing.
- Example:
  - elements $e_1$, $e_2$, …$e_i$, …$e_n$ exercised with probabilities $p_1$, $p_2$, …$p_i$, …$p_n$
  - Profile then is $\{(e_i, p_i)\}$ for all elements
- For example, a Search program can be tested for text data, numerical data, data already sorted etc. If most testing is done using numerical data, more bugs related to text data may remain unfound.

# Input mix: Test Profile

- **The ideal Profile (input mix) will depend on the objective**
  - A. Find bugs fast? or
  - B. Estimate operational failure rate?

A. Best mix for **functional** bug finding (Li & Malaiya' 94)

- Quick & limited testing: *Use **operational profile: how the inputs are encountered in actual operation.***
- High reliability: *Probe input space evenly*
  - Operational profile will not execute rare and special cases, the main cause of failures in highly reliable systems.
- Very high reliability: corner cases and rare combinations

B. For **security bugs**: corner cases and rare combinations

- Vulnerability finders / exploiters look for these.

N. Li and Y.K. Malaiya, On Input Profile Selection for Software Testing, Proc. Int. Symp. Software Reliability Engineering, Nov. 1994, pp. 196-205.

H. Hecht, P. Crane, Rare conditions and their effect on software failures, Proc. Annual Reliability and Maintainability Symposium, 1994, pp. 334-337

# Who is writing bad code?

