Writing a Good Report

By documenting our findings clearly and concisely, we get straight to our point in a way that the security or triage team can comprehend. Most importantly, bug reports should include information on how exploitation of each vulnerability can be reproduced step-by-step.

Please note that when reporting to less mature companies, we may have to translate technical security issues into more understandable/business terms for them to understand the actual impact of each vulnerability.

The essential elements of a good bug report are (the element order can vary):

`Vulnerability Title`	Including vulnerability type, affected domain/parameter/endpoint, impact etc.
`CWE & CVSS score`	For communicating the characteristics and severity of the vulnerability.
`Vulnerability Description`	Better understanding of the vulnerability cause.
`Proof of Concept (POC)`	Steps to reproduce exploiting the identified vulnerability clearly and concisely.
`Impact`	Elaborate more on what an attacker can achieve by fully exploiting the vulnerability. Business impact and maximum damage should be included in the impact statement.
`Remediation`	Optional in bug bounty programs, but good to have.

Readable and well-formatted bug reports can drastically minimize both vulnerability reproduction time and time to triage.

Why CWE & CVSS?

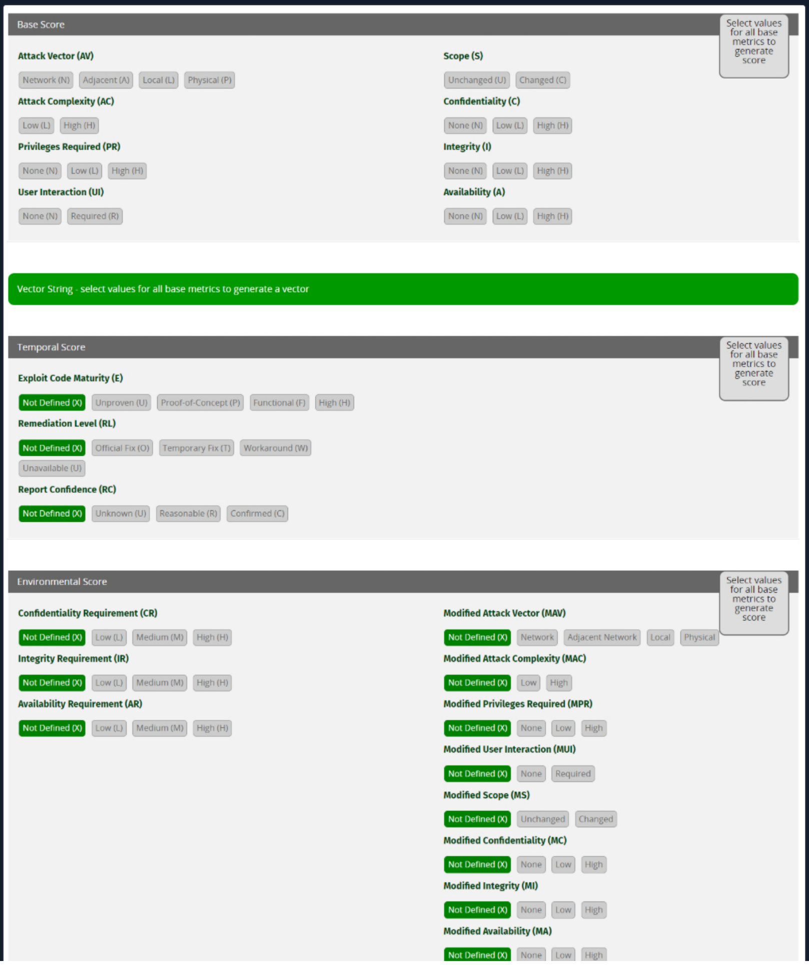
MITRE describes Common Weaknesses Enumeration (CWE) as a community-developed list of software and hardware weakness types. It serves as a common language, a measuring stick for security tools, and as a baseline for weakness identification, mitigation, and prevention efforts. In the case of a vulnerability chain, choose a CWE related to the initial vulnerability.

When it comes to communicating the severity of an identified vulnerability, then Common Vulnerability Scoring System (CVSS) should be used, as it is a published standard used by organizations worldwide.

Using CVSS Calculator

Let us now see how we can use the CVSS v3.1 Calculator identify the severity of an identified vulnerability.

We will focus on the `Base Score` area only.



Base Score — Select values for all base metrics to generate score

Attack Vector (AV): Network (N) | Adjacent (A) | Local (L) | Physical (P)

Attack Complexity (AC): Low (L) | High (H)

Privileges Required (PR): None (N) | Low (L) | High (H)

User Interaction (UI): None (N) | Required (R)

Scope (S): Unchanged (U) | Changed (C)

Confidentiality (C): None (N) | Low (L) | High (H)

Integrity (I): None (N) | Low (L) | High (H)

Availability (A): None (N) | Low (L) | High (H)

Vector String - select values for all base metrics to generate a vector

Temporal Score — Select values for all base metrics to generate score

Exploit Code Maturity (E): [✓ Not Defined (X)] | Unproven (U) | Proof-of-Concept (P) | Functional (F) | High (H)

Remediation Level (RL): [✓ Not Defined (X)] | Official Fix (O) | Temporary Fix (T) | Workaround (W) | Unavailable (U)

Report Confidence (RC): [✓ Not Defined (X)] | Unknown (U) | Reasonable (R) | Confirmed (C)

Environmental Score — Select values for all base metrics to generate score

Confidentiality Requirement (CR): [✓ Not Defined (X)] | Low (L) | Medium (M) | High (H)

Integrity Requirement (IR): [✓ Not Defined (X)] | Low (L) | Medium (M) | High (H)

Availability Requirement (AR): [✓ Not Defined (X)] | Low (L) | Medium (M) | High (H)

Modified Attack Vector (MAV): [✓ Not Defined (X)] | Network | Adjacent Network | Local | Physical

Modified Attack Complexity (MAC): [✓ Not Defined (X)] | Low | High

Modified Privileges Required (MPR): [✓ Not Defined (X)] | None | Low | High

Modified User Interaction (MUI): [✓ Not Defined (X)] | None | Required

Modified Scope (MS): [✓ Not Defined (X)] | Unchanged | Changed

Modified Confidentiality (MC): [✓ Not Defined (X)] | None | Low | High

Modified Integrity (MI): [✓ Not Defined (X)] | None | Low | High

Modified Availability (MA): [✓ Not Defined (X)] | None | Low | High

? Go to Questions

Table of Contents

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

Attack Vector

Shows how the vulnerability can be exploited.

- `Network (N):` Attackers can only exploit this vulnerability through the network layer (remotely exploitable).
- `Adjacent (A):` Attackers can exploit this vulnerability only if they reside in the same physical or logical network (secure VPN included).
- `Local (L):` Attackers can exploit this vulnerability only by accessing the target system locally (e.g., keyboard, terminal, etc.) or remotely (e.g., SSH) or through user interaction.
- `Physical (P):` Attackers can exploit this vulnerability through physical interaction/manipulation.

Attack Complexity

Depicts the conditions beyond the attackers' control and must be present to exploit the vulnerability successfully.

- `Low (L):` No special preparations should take place to exploit the vulnerability successfully. The attackers can exploit the vulnerability repeatedly without any issue.
- `High (H):` Special preparations and information gathering should take place to exploit the vulnerability successfully.

Privileges Required

Show the level of privileges the attacker must have to exploit the vulnerability successfully.

- `None (N):` No special access related to settings or files is required to exploit the vulnerability successfully. The vulnerability can be exploited from an unauthorized perspective.
- `Low (L):` Attackers should possess standard user privileges to exploit the vulnerability successfully. The exploitation in this case usually affects files and settings owned by a user or non-sensitive assets.
- `High (H):` Attackers should possess admin-level privileges to exploit the vulnerability successfully. The exploitation in this case usually affects the entire vulnerable system.

User Interaction

Shows if attackers can successfully exploit the vulnerability on their own or user interaction is required.

- `None (N):` Attackers can successfully exploit the vulnerability independently.
- `Required (R):` A user should take some action before the attackers can successfully exploit the vulnerability.

Scope

Shows if successful exploitation of the vulnerability can affect components other than the affected one.

- `Unchanged (U):` Successful exploitation of the vulnerability affects the vulnerable component or affects resources managed by the same security authority.
- `Changed (C):` Successful exploitation of the vulnerability can affect components other than the affected one or resources beyond the scope of the affected component's security authority.

Confidentiality

Shows how much the vulnerable component's confidentiality is affected upon successfully exploiting the vulnerability. Confidentiality limits information access and disclosure to authorized users only and prevents unauthorized users from accessing information.

- `None (N):` The confidentiality of the vulnerable component does not get impacted.
- `Low (L):` The vulnerable component will experience some loss of confidentiality upon successful exploitation of the vulnerability. In this case, the attackers do not have control over what information is obtained.
- `High (H):` The vulnerable component will experience total (or serious) loss of confidentiality upon successfully exploiting the vulnerability. In this case, the attackers have total (or some) control over what information is obtained.

Integrity

Shows how much the vulnerable component's integrity is affected upon successfully exploiting the vulnerability. Integrity refers to the trustworthiness and veracity of information.

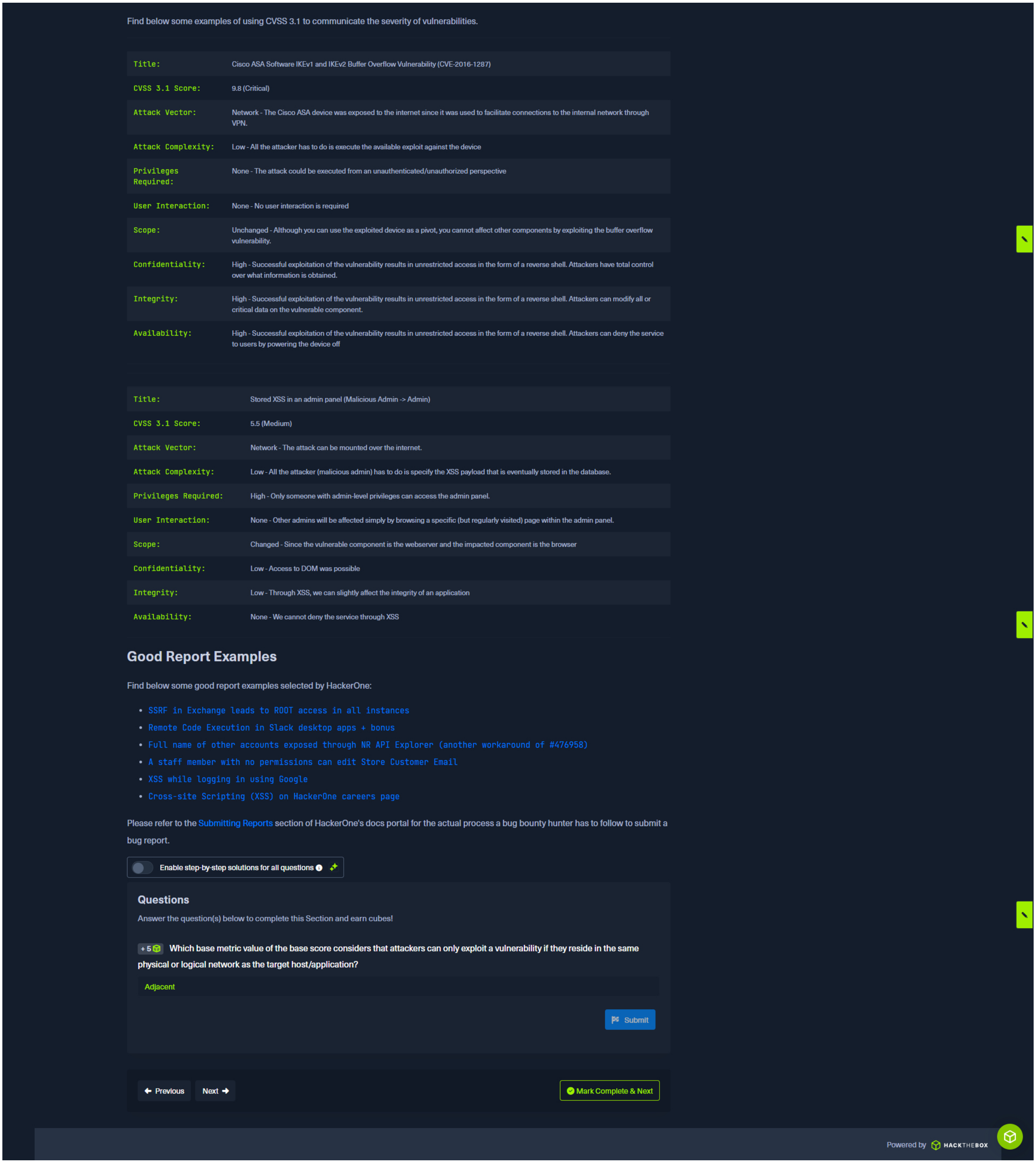
- `None (N):` The integrity of the vulnerable component does not get impacted.
- `Low (L):` Attackers can modify data in a limited manner on the vulnerable component upon successfully exploiting the vulnerability. Attackers do not have control over the consequence of a modification, and the vulnerable component does not get seriously affected in this case.
- `High (H):` Attackers can modify all or critical data on the vulnerable component upon successfully exploiting the vulnerability. Attackers have control over the consequence of a modification, and the vulnerable component will experience a total loss of integrity.

Availability

Shows how much the vulnerable component's availability is affected upon successfully exploiting the vulnerability. Availability refers to the accessibility of information resources in terms of network bandwidth, disk space, processor cycles, etc.

- `None (N):` The availability of the vulnerable component does not get impacted.
- `Low (L):` The vulnerable component will experience some loss of availability upon successfully exploiting the vulnerability. The attacker does not have complete control over the vulnerable component's availability and cannot deny the service to users, and performance is just reduced.
- `High (H):` The vulnerable component will experience total (or severe) availability loss upon successfully exploiting the vulnerability. The attacker has complete (or significant) control over the vulnerable component's availability and can deny the service to users. Performance is significantly reduced.

Examples



Find below some examples of using CVSS 3.1 to communicate the severity of vulnerabilities.

Title:	Cisco ASA Software IKEv1 and IKEv2 Buffer Overflow Vulnerability (CVE-2016-1287)
CVSS 3.1 Score:	9.8 (Critical)
Attack Vector:	Network - The Cisco ASA device was exposed to the internet since it was used to facilitate connections to the internal network through VPN.
Attack Complexity:	Low - All the attacker has to do is execute the available exploit against the device
Privileges Required:	None - The attack could be executed from an unauthenticated/unauthorized perspective
User Interaction:	None - No user interaction is required
Scope:	Unchanged - Although you can use the exploited device as a pivot, you cannot affect other components by exploiting the buffer overflow vulnerability.
Confidentiality:	High - Successful exploitation of the vulnerability results in unrestricted access in the form of a reverse shell. Attackers have total control over what information is obtained.
Integrity:	High - Successful exploitation of the vulnerability results in unrestricted access in the form of a reverse shell. Attackers can modify all or critical data on the vulnerable component.
Availability:	High - Successful exploitation of the vulnerability results in unrestricted access in the form of a reverse shell. Attackers can deny the service to users by powering the device off

Title:	Stored XSS in an admin panel (Malicious Admin -> Admin)
CVSS 3.1 Score:	5.5 (Medium)
Attack Vector:	Network - The attack can be mounted over the internet.
Attack Complexity:	Low - All the attacker (malicious admin) has to do is specify the XSS payload that is eventually stored in the database.
Privileges Required:	High - Only someone with admin-level privileges can access the admin panel.
User Interaction:	None - Other admins will be affected simply by browsing a specific (but regularly visited) page within the admin panel.
Scope:	Changed - Since the vulnerable component is the webserver and the impacted component is the browser
Confidentiality:	Low - Access to DOM was possible
Integrity:	Low - Through XSS, we can slightly affect the integrity of an application
Availability:	None - We cannot deny the service through XSS

Good Report Examples

Find below some good report examples selected by HackerOne:

- SSRF in Exchange leads to ROOT access in all instances
- Remote Code Execution in Slack desktop apps + bonus
- Full name of other accounts exposed through NR API Explorer (another workaround of #476958)
- A staff member with no permissions can edit Store Customer Email
- XSS while logging in using Google
- Cross-site Scripting (XSS) on HackerOne careers page

Please refer to the Submitting Reports section of HackerOne's docs portal for the actual process a bug bounty hunter has to follow to submit a bug report.

Questions

Answer the question(s) below to complete this Section and earn cubes!

+ 5 Which base metric value of the base score considers that attackers can only exploit a vulnerability if they reside in the same physical or logical network as the target host/application?

Adjacent