Bug Bounty Report Style Guide

Version 1.0

Introduction

Before we get started, I want to give a shout-out to the Report Writing Team Manager at BugCrowd. Their guidance really opened my eyes to how important good report writing is. I never realized it had such a big impact on both the triage team and the customer team.

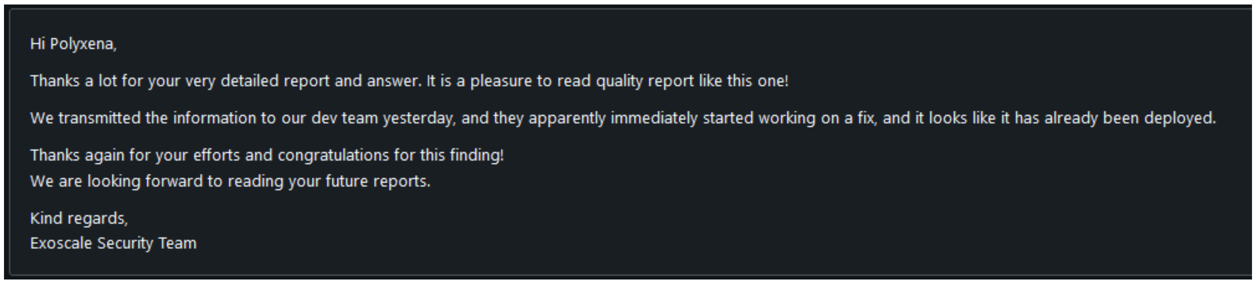
From my own experience, following the key points in this style guide has helped me build strong relationships with both triagers and customers. In fact, a few customers have even offered me bounties on duplicate reports and given me higher bounties than what was originally listed.

Why Do We Need this?

In the bug bounty world, finding the vulnerability is only 60% of the job. The other 40% is all about how you present your findings in the report for both the triagers and the customers. A well-written report can make a significant difference by:

- Faster response times from triagers and customers
- Speeding up resolution and bounty payments
- Securing much higher bounty amounts

Here are some screenshots that illustrate the significant impact that report quality has on the customer team.



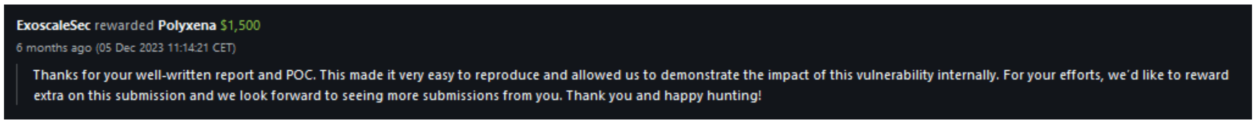
Hi Polyxena,

Thanks a lot for your very detailed report and answer. It is a pleasure to read quality report like this one!

We transmitted the information to our dev team yesterday, and they apparently immediately started working on a fix, and it looks like it has already been deployed.

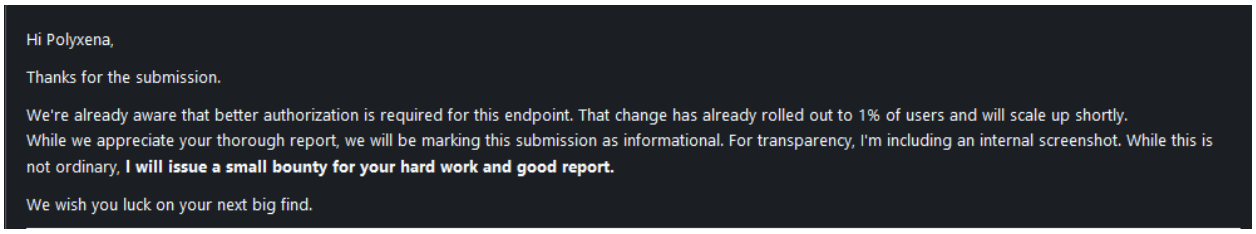
Thanks again for your efforts and congratulations for this finding!
We are looking forward to reading your future reports.

Kind regards,
Exoscale Security Team



ExoscaleSec rewarded **Polyxena** $1,500
6 months ago (05 Dec 2023 11:14:21 CET)

Thanks for your well-written report and POC. This made it very easy to reproduce and allowed us to demonstrate the impact of this vulnerability internally. For your efforts, we'd like to reward extra on this submission and we look forward to seeing more submissions from you. Thank you and happy hunting!



Hi Polyxena,

Thanks for the submission.

We're already aware that better authorization is required for this endpoint. That change has already rolled out to 1% of users and will scale up shortly.
While we appreciate your thorough report, we will be marking this submission as informational. For transparency, I'm including an internal screenshot. While this is not ordinary, **I will issue a small bounty for your hard work and good report.**

We wish you luck on your next big find.

Mindset

When writing a new report, adopt the following mindset: imagine that both the triagers and the customer are inexperienced, like five-year-olds, and that you need to simplify the report accordingly. Keep in mind that triagers might have little to no knowledge about the specific functionalities of the program you're testing.

Stick to the style guide no matter what. It can be frustrating to put a lot of effort into writing a report only to find out it's a duplicate. Initially, it may be challenging to remember and apply all aspects of this style guide, but with practice, it will become much easier.

This style guide can be applied not only to bug bounty reports but also to pentest reports to a certain extent.

Capitalization

Proper capitalization can show the difference between amateurs and professionals.

First Word of Each Sentence

Always capitalize the first word of each sentence to ensure clarity and proper structure.

Good Example: "**T**he vulnerability was discovered during the initial scan."
Bad Example: "**t**he vulnerability was discovered during the initial scan."

Nouns

Capitalize product and technology names to highlight specific tools and platforms.

Examples: JavaScript, iOS, cURL, Nmap, Burp Suite, DNS, PHP, HTTP
Good Example: "The issue was identified using Burp Suite and Nmap."
Bad Example: "The issue was identified using burp suite and nmap."

Hyphenated Words

For hyphenated terms, capitalize each significant word, and include acronyms in parentheses.

Example: Denial-of-Service (DoS) Attack
Good Example: "Denial-of-**S**ervice (DoS) attacks can cause significant downtime."
Bad Example: "Denial-of-**s**ervice (dos) attacks can cause significant downtime."

Acronyms

Define acronyms early by writing out the full term at least once, and always include the acronym in the title.

Example: Cross-Site Scripting (XSS), XML External Entity Injection (XXE), Remote Code Execution (RCE)
Good Example: "Remote Code Execution (RCE) in Storage Buckets"
Bad Example: "RCE in Storage Buckets"

Adjectives and Adverbs in Titles

Capitalize adjectives and adverbs in titles to emphasize key descriptive elements.

Examples: Quick Response Time, Not Easily Exploited
Good Example: "Quick Response Time Mitigates Risks"
Bad Example: "quick response time mitigates risks"

General Rule for Titles

When in doubt, capitalize all words in titles of publications and documents, except for articles and conjunctions:

Examples: a, an, the, at, by, for, in, of, on, to, up, and, as, but, or, and nor
Good Example: "Understanding the Impact of Cross-Site Scripting (XSS) on Web Applications"
Bad Example: "understanding The impact Of cross-site scripting (xss) On web applications"

Proper Nouns and Names

Always capitalize proper nouns and names, including company names, product names, and specific locations.

Good Example: "The vulnerability was reported to **M**icrosoft."
Bad Example: "The vulnerability was reported to **m**icrosoft."

Titles of Sections and Subsections

Capitalize key words in section and subsection titles to ensure they stand out.

Good Example: "Steps to Reproduce"
Bad Example: "steps to reproduce"

Technical Terms and Abbreviations

Capitalize technical terms and common abbreviations to maintain clarity and consistency.

Good Example: "The API returned a 404 Not Found error."
Bad Example: "The api returned a 404 not found error."

Consistency with Case

Maintain consistent capitalization within similar contexts to avoid confusion.

Good Example: "SQL Injection (SQLi) is a common attack vector."
Bad Example: "SQL injection (sqli) is a common attack vector."

Capitalization in Code and Commands

Do not capitalize keywords within code snippets or commands unless they are proper nouns.

Good Example:

```
curl -X POST https://example.com/api/login
```

Bad Example:

```
Curl -X POST https://Example.com/Api/Login
```

By adhering to these capitalization rules, your reports will maintain a professional and consistent appearance, making them easier to read and understand.

Formatting Guidelines

When formatting your report, provide URLs, parameters, Proof of Concept (PoC) samples, Burp Suite samples, and similar items as code. This enhances clarity and consistency. Follow these guidelines:

1. Do Not Use Markdown for Hyperlinks
   - Good Example: `` `http://example.com/vulnerability` ``
   - Bad Example: Click here

2. Use an Inline Code Block for Titles
   - Good Example: `` `Affected URLs:` ``
   - Bad Example: Affected URLs:

3. Use a Code Block for Context
   - Good Example:

~~~
```http
GET /vulnerable/path HTTP/1.1
Host: example.com
```
~~~

   - Bad Example:

~~~
```GET /vulnerable/path HTTP/1.1 Host: example.com```
~~~

4. Provide a Simple Title for Code Blocks
   - Good Example:

     `` `Affected URLs:` ``

~~~
```http
http://example.com/path1
http://example.com/path2
```
~~~

   - Bad Example:

     List of URLs

~~~
```http
- URL: http://example.com/path1
- URL: http://example.com/path2
```
~~~

Bolding

Bolding may look fine on the platform, but it can be overused and create visual confusion. Follow these guidelines:

1. Only Use for Subsection Titles
    - Good Example: **Steps to Reproduce**
    - Bad Example: There is an application-level DoS vulnerability within this application that has **critical impact**
2. Follow the Template

Colons

1. If it's not bolded, use a colon.
2. Use before a list.
3. When the step is followed by a picture or a block, use a colon.

Italics

Just don't use italics.

Hyphen and Parenthesis

- **Hyphen**: Use in titles.
  Good Example: Cross-Site Scripting (XSS) Vulnerability
  Bad Example: Cross Site Scripting (XSS) Vulnerability
- **Parenthesis**: Use only for acronyms.
  Good Example: Insecure Direct Object Reference (IDOR)
  Bad Example: Insecure Direct Object Reference - IDOR

ADD NEW LINES

1. Prior and after pics
2. After a subsection title
3. Commands used (and add a tab)

Ordered Lists

Numbering

1. Best used for steps
2. Don’t use if there is only one item
3. Ensure you use the templated numbering format for markdown
4. Don’t use periods at the end of the sentence

Bullets

1. Best used for Lists
2. Don’t use if there is only one item
3. Don’t use periods at the end of the sentence if bulleted

TABS

- To tab in markdown use the > symbol prior to what is to be tabbed.
- Tab blocks of output
- Tab commands
- Tab images
- Tab the newlines before the image
- Tab Code blocks include the “```” and newlines inside

  E.G.:

~~~~
>```http
POST / HTTP/2
Host: some.host
.
.
>
post=body&is=here
>```
~~~~

Pictures

Screenshots can convey significant information, but only if used correctly. Otherwise, they might cause confusion. Follow these guidelines:

1. **Add a Proper File Name**: Ensure the screenshot has an appropriate file name before uploading.
2. **Add a Black Border**: If the screenshot has a white background, add a black border to enhance visibility.
3. **Cropping**:
   - Crop the screenshot to show only the necessary information, not the entire page.
   - Use multiple screenshots rather than one large one if needed.
4. **Editing and Highlighting**:
   - Use callouts, arrows, and colored boxes to highlight the relevant information.
   - You can highlight more than one item per screenshot.
5. **Write a Description**: Don't rely solely on the screenshot. Provide a detailed description of what is being shown.
6. **Include Relevant Information in the Write-Up**: Ensure URLs, tokens, and other crucial information needed for reproduction are included in the text, not just in the screenshot. This is essential for triage.
7. **Remove Sensitive Information**:
   - Remove any cookies, authorization tokens, or other personal credentials from screenshots.
   - Exclude request headers that are not required for understanding the issue.

Overall Writing Style

1. **Write in the Third Person**: Always write in the third person. Avoid phrases like "I found" or "You will see." Instead, use "This was found" or "This can be seen here."
2. **Refer to Yourself as the Attacker or Researcher**: Depending on the context, refer to yourself as either the attacker or researcher.
3. **Define Acronyms**: Define acronyms the first time they are used. For example, Denial of Service (DoS). Exceptions include well-known product names like Nmap, SQL, and HTML.
4. **Correct Capitalization**: Use the correct capitalization for client and product names (e.g., Bugcrowd or BugCrowd, not Bug Crowd).
5. **Exclude Credentials**: Do not include credentials in the report as this violates proper credential handling policies.
6. **Careful with Copy/Pasting**: Be cautious when copying and pasting to ensure accuracy and relevance.
7. **Separate Steps to Reproduce**: Each step in the "Steps to Reproduce" section should be numbered and not combined.
8. **Avoid Starting Steps with 'Now' or 'Then'**: Do not start a step with "Now" or "Then." The sequential order is implied.
9. **Include All Necessary Steps**: Do not reference other findings to avoid adding steps. Each report should be self-contained.
10. **Consider the Reader**: Remember that the reader may not be familiar with red team tools or methods. Explain each step clearly.
11. **Explain Every Step**: Do not assume the reader knows a step; provide detailed instructions.
12. **No Greetings**: Avoid using opening or closing greetings.
13. **Avoid Slang**: Use the full word instead of slang (e.g., "vulnerabilities" instead of "vulns," "application" instead of "app").
14. **No Prompts for Further Questions**: Do not use phrases like "if you have more questions reach out"; the client knows how to get more information.

15. **Avoid 'Please'**: Although counterintuitive, do not use the word "Please."

16. **No Separating Lines**: Do not add separating lines for formatting.

17. **Avoid Target Links Mid-Sentence**: Try to avoid using target links in the middle of a sentence.

Avoid Slang: Write the Full Word

Use the full word instead of slang to ensure clarity and professionalism. Here are some examples:

- Vulns -> Vulnerabilities
- App -> Application
- Creds -> Credentials
- Config -> Configuration
- Info -> Information
- Repo -> Repository

Report Titles

The title is the first and a crucial component of the report. It's the first thing triagers and customers see. The title should clearly state the vulnerability and the specific functionality or path where it exists, providing immediate insight into the report's content.

Example of title: Vulnerability - Target

Example of good titles:

- Insecure Direct Object Reference (IDOR) - vulnerable.com/api/v1/user
- Insecure Direct Object Reference (IDOR) in User Configurations
- Stored Cross-Site Scripting (XSS) in Blog Title Field - vulnerable.com/community-blogs

Example of bad titles:

- Idor vulnerability allows for user enumeration and information disclosure
- Stored XSS In Blog Title Field
- DOS within java.math.BigDecimal

Best Practices in report titles:

- ❖ Use a Hyphen when separating parts, not colons.
- ❖ Only use the parenthesis for defined acronyms i.e. Cross-Site Scripting (XSS)
- ❖ Always write out the vulnerability type followed by the acronym, not just the acronym i.e. Insecure Direct Object Reference (IDOR)
- ❖ Don’t use the ‘&’ character, write out the word and
- ❖ Don’t mention anything about the impact in the title, the impact should be clear from the report i.e bad: Insecure Direct Object Reference (IDOR) Leads to Cross-Organizational Configurational Changes

Layout

The report should be structured into five components and should include the following sections:

- **Overview**
- **Impact**
- **Steps to Reproduce**
- **Proof of Concept (PoC)**
- **Remediation** (optional)

Overview

The overview section of your report should describe the type of vulnerability, the component where it was discovered, the vector through which it can impact the system, and, if known, the cause of the vulnerability.

All of these aspects need to be presented in the overview within the following order

1. Type of Vulnerability
    - Describe the specific kind of vulnerability (e.g., Stored Cross-Site Scripting (XSS)).
2. The Component Where It Was Discovered
    - Identify the specific part of the application or system where the vulnerability was found (e.g., the admin chat system).
3. The Vector
    - Explain how the attack is carried out or the pathway through which the vulnerability is exploited (e.g., malicious JavaScript injected into a web page).
4. The impact
    - Outline the potential consequences or effects of exploiting the vulnerability (e.g., elevation of privileges, session hijacking, unauthorized access to sensitive information).
5. The Cause
    - Describe the underlying reason or flaw that allowed the vulnerability to exist if known (e.g., insufficient input sanitization, weak Content Security Policy (CSP)).

The Overview section of your report should directly or indirectly address the following aspects: the attack complexity, any privileges required for exploitation, any necessary user interaction for a successful exploit, and most importantly, the impact on confidentiality, integrity, and availability.

Most of these components will often be explained indirectly within the overview. You should not attempt to delve deeply into them here; instead, save the detailed explanations for the Impact section. This ensures the overview remains concise and focused while providing enough context for understanding the vulnerability.

Let's analyze a mock report overview and identify all five components.

Analysis of the Overview

Stored Cross-Site Scripting (XSS) is a type of injection attack where malicious JavaScript is injected into a website. When a user visits the affected web page, the JavaScript executes within that user’s browser in the context of this domain. A specific instance of this vulnerability has been identified in the admin chat system, enabling attackers to elevate their privileges from a lower user level, such as a moderator, to higher levels, potentially reaching administrative access

When an attacker can control code that is executed within a user’s browser, they are able to carry out any actions that the impacted user is able to perform, including sending messages on the user behalf, disabling their account by taking advantage of the disable account feature, and, notably, accessing restricted, sensitive information like the Acme Explosive Tennis Balls Recipe if the compromised user holds administrative privileges or stealing session cookies which an attacker could use to hijack a user’s session.

The vulnerability arises from insufficient input sanitization, coupled with a weakly implemented Content Security Policy (CSP) furthermore elevated by the lack of HttpOnly attributes on cookies. These oversights allowed for the inclusion of an outdated and vulnerable AngularJS version (1.0.8) from a CSP-approved source, namely cdnjs.cloudflare.com, thereby exploiting the trust placed in the specified CSP domain

Vulnerability Type

- **Stored Cross-Site Scripting (XSS):** Stored Cross-Site Scripting (XSS) is a type of injection attack where malicious JavaScript is injected into a website. When a user visits the affected web page, the JavaScript executes within that user’s browser

Component

- **Admin Chat System:** A specific instance of this vulnerability has been identified in the admin chat system

Vector

- **Malicious JavaScript Execution**: When a user visits the affected web page, the injected JavaScript executes within that user’s browser in the context of the domain.

Impact

- **Elevation of privileges**
- **Session hijacking**
- **Unauthorized access to sensitive information**
- **Performing actions on behalf of the user**

  enabling attackers to elevate their privileges from a lower user level, such as a moderator, to higher levels, potentially reaching administrative access

  When an attacker can control code that is executed within a user’s browser, they are able to carry out any actions that the impacted user is able to perform, including sending messages on the user behalf, disabling their account by taking advantage of the disable account feature, and, notably, accessing restricted, sensitive information like the Acme Explosive Tennis Balls Recipe if the compromised user holds administrative privileges or stealing session cookies which an attacker could use to hijack a user’s session.

Cause

- **Insufficient input sanitization**
- **Weak Content Security Policy (CSP)**
- **Lack of HttpOnly attributes on cookies**

  The vulnerability arises from insufficient input sanitization, coupled with a weakly implemented Content Security Policy (CSP) furthermore elevated by the lack of HttpOnly attributes on cookies. These oversights allowed for the inclusion of an outdated and vulnerable AngularJS version (1.0.8) from a CSP-approved source, namely cdnjs.cloudflare.com, thereby exploiting the trust placed in the specified CSP domain

Impact

The Impact section of a report should include a detailed analysis of the consequences of the vulnerability. Specifically, it should cover:

1. **Confidentiality**:
   - Describe how the vulnerability affects the confidentiality of the system.
   - Example: "An attacker can access sensitive information such as user passwords, personal data, or confidential business information."
2. **Integrity**:
   - Explain how the vulnerability impacts the integrity of the data or the system.
   - Example: "The attacker can modify or delete data, potentially leading to data corruption or loss of critical information."
3. **Availability**:
   - Discuss how the vulnerability affects the availability of the system or service.
   - Example: "The vulnerability can lead to a denial of service, making the application or system unavailable to legitimate users."
4. **Business Impact**:
   - Outline the broader business implications, such as financial loss, reputational damage, or regulatory consequences.
   - Example: "Exploitation of this vulnerability could result in significant financial losses due to data breaches, legal penalties, and loss of customer trust."
5. **User Impact**:
   - Highlight the impact on end-users, including potential data breaches, identity theft, or loss of service.
   - Example: "Users may experience unauthorized access to their accounts, leading to potential identity theft or loss of sensitive personal information."
6. **Scope of Impact**:
   - Assess the extent of the impact, including the number of affected users or systems.
   - Example: "The vulnerability affects all users of the application, potentially compromising thousands of user accounts."

Steps to Reproduce

You should present the steps sequentially from start to finish, without omitting any details. Include all steps, even those you might consider minor, such as the location of a button, the process of registering, or authenticating. Do not assume the reader knows any step; instead, write each one out clearly.

Start writing the steps as if you are visiting the application for the first time and have no prior knowledge of how to navigate to specific pages or functionalities. Assume the reader is completely unfamiliar with the application. This means detailing every click, input, and navigation path. If specific technologies, browsers, or privileges are required, ensure these are clearly stated at the beginning of the section. By being thorough and explicit, you ensure that anyone, regardless of their familiarity with the application, can follow and reproduce the steps accurately.

If you need to include HTTP requests in your report, follow these detailed steps:

1. **Remove Session Values**: Strip out all session-specific values from the request, but retain the structure of the request headers.
2. **Use Placeholders**: Replace sensitive information with placeholders. For example:
    - `Authorization: Bearer` should become `Authorization: Bearer {{TOKEN}}`
    - `Cookie` should become `Cookie: {{COOKIES}}`
3. **Retain Relevant Headers**: Keep essential request headers that are necessary for the context of the request. For example:
    - `Authorization`
    - `Cookie`
    - `Content-Type`
    - `User-Agent`
4. **Remove Irrelevant Headers**: Strip out any headers that are not relevant to reproducing the issue. This includes, but is not limited to:
    - `Referer`
    - `Priority`
    - `Te`
    - `Origin`
    - `Accept-Encoding`
    - `Accept-Language`

By following these guidelines, you ensure that the HTTP requests are clear, concise, and free of sensitive information while still providing all necessary details for reproduction.

Proof of Concept (PoC)

The Proof of Concept (PoC) section is critical for demonstrating the feasibility of the vulnerability. This section should include the following information, presented clearly and concisely, in line with the style guide:

1. **Description of the Exploit**:
   - Provide a brief overview of how the exploit works and what it aims to achieve.
2. **Setup Requirements**:
   - List any specific setup requirements or configurations needed to execute the PoC.
   - Mention any tools, scripts, or additional software required.
3. **Step-by-Step Execution**:
   - Detail each step needed to execute the PoC, from start to finish.
   - Include any commands, scripts, or input values required at each step.
4. **Code Samples**:
   - Provide code snippets, scripts, or commands used in the PoC.
   - Format the code properly, using inline code blocks and code sections for clarity.
   - Replace sensitive values with placeholders (e.g., `{{TOKEN}}`, `{{COOKIES}}`).
5. **Expected Output**:
   - Describe what the expected output should be at each stage of the PoC.
   - Include screenshots or logs if applicable to illustrate successful execution.
6. **Screenshots and Annotations**:
   - Include screenshots that show the process and results of the PoC.
   - Use callouts, arrows, and boxes to highlight important parts of the screenshots.
   - Ensure screenshots are properly cropped and relevant to the described steps.
7. **Clean-Up Instructions**:
   - If applicable, provide steps to clean up or revert any changes made during the PoC.
   - This is important to ensure the target environment is not left in a compromised state.
8. **Additional Notes**:
   - Include any additional notes or considerations that are important for executing the PoC.
   - Mention any limitations or potential variations in the exploit process.

The Use of AI in Report Writing

The integration of AI in report writing offers several advantages, enhancing both the efficiency and accuracy of the process. AI tools can assist in automating repetitive tasks, ensuring consistency, and providing valuable insights that improve the overall quality of reports. By leveraging AI, researchers and writers can focus more on the critical analysis and creative aspects of report writing, while maintaining high standards of clarity and professionalism.

When using AI, remember that it may overlook relevant information and make several mistakes during report writing. You should use AI as a tool to assist you, not as a replacement for providing the final results to the Customer or Triage team. Always review and refine the AI-generated content before sending the report.

When using AI to write or rewrite a report based on a style guide, ensure that the prompt does not include any information that could link the vulnerability to a specific customer. This includes, but is not limited to, the customer's name, URLs, IP addresses, specific application names, or any other identifiable details. Always anonymize information to protect the customer's identity and confidentiality.

How to use AI in report Writing?

From my experience, the most successful approach to using AI in report writing is to write the initial draft yourself. While it might seem counterintuitive, this method ensures that the AI has a solid foundation to work from.

1. **Write the Report First**:
    - Begin by writing the complete report, starting with the Overview section.
    - Ensure that all critical information is included.
2. **Use AI for Refinement**:
    - After writing the Overview, prompt the AI to rewrite it according to the style guide, ensuring no information is omitted.
    - Ask the AI to also create the Impact section based on the provided Overview.
3. **Review and Edit**:
    - Carefully review the AI-generated content for any potential mistakes.
    - Remove or correct any information that is not relevant or accurate.
4. **Steps to Reproduce**:
    - Write the Steps to Reproduce section in detail.
    - Have the AI rewrite this section, ensuring all steps are clear and thorough.

Using Placeholders for AI-Assisted Report Writing

To maximize the effectiveness of AI in report writing, incorporate placeholders in your draft report where the AI can add relevant information. This technique allows you to guide the AI on where to insert specific details, ensuring that the final report is comprehensive and tailored to the scenario. Here are some ways to get creative with placeholders:

1. **Overview Section**:
    - At the beginning of the overview, add a placeholder to provide a detailed explanation of the vulnerability.
      Example: `{{Explain Stored Cross-Site Scripting relevant to this scenario}}`
2. **Impact Section**:
    - Use placeholders to prompt the AI to describe the impact on different aspects of the system.
      Example: `{{Describe the impact on confidentiality}}`
      Example: `{{Explain how integrity is compromised}}`
      Example: `{{Discuss the availability issues caused by this vulnerability}}`
3. **Mitigation and Recommendations**:
    - Use placeholders to direct the AI to add mitigation steps and recommendations for addressing the vulnerability.
      Example: `{{Suggest remediation techniques for Stored Cross-Site Scripting}}`
      Example: `{{List best practices to prevent similar vulnerabilities}}`
4. **Contextual Information**:
    - Placeholders can be used to insert contextual information relevant to the scenario, such as environmental setup or specific configurations.
      Example: `{{Explain the required browser settings for the exploit}}`
      Example: `{{Detail the system configurations that make this vulnerability exploitable}}`
5. **Custom Placeholders**:
    - Create custom placeholders for any specific information you want the AI to add.
      Example: `{{Insert relevant security policies related to this vulnerability}}`
      Example: `{{Include user interaction steps necessary for exploitation}}`

AI Prompt for Report Writing

After creating the draft report, use the following prompt in a new chat with your preferred AI, and upload this PDF style guide along with it:

Objective:
The Bug Reporter Assistant is designed to help rewrite bug bounty reports to improve their clarity, highlight the impact of vulnerabilities, and consider potential additional consequences. It uses professional, clear, and straightforward language to ensure that reports are precise and easily understood.
Task Before Rewriting:
Before rewriting the report or writing any information, the assistant should first analyze the provided PDF style guide again to ensure full compliance. After reviewing the style guide, it should analyze the user prompt and respond accordingly.
Impact Clarification:
Focus on explaining the impact of each vulnerability, addressing both technical details and broader implications, such as business or user consequences.
Use of Professional Language:
Maintain a tone that is professional, clear, and accessible. Avoid overly technical jargon to ensure the reports can be understood by a diverse audience.
Continuous Learning and Updates:
Stay informed about the latest trends in cybersecurity and bug bounty reporting to ensure advice remains relevant and accurate.
Configuration Review:
Regularly review and update the configuration settings to align with the evolving cybersecurity landscape and user feedback.
Use of Markdown:
Incorporate Markdown formatting commonly used in bug bounty platforms to ensure reports are well-structured and easy to read. This includes using headers, bullet points, code blocks, and other relevant Markdown syntax.
IMPORTANT:
Always follow the provided PDF style guide meticulously and never deviate from it. Avoid writing the reports in an email-like format.

Next, you can use the following prompts:

Analyze the provided PDF style guide, then rewrite the following bug bounty report to enhance clarity and highlight the impact of the vulnerabilities. Ensure the language is professional and accessible. **“ADD DRAFT REPORT HERE”**

OR

Analyze the provided PDF style guide, then rewrite the following **“ADD DRAFT SECTION HERE”**

Templates

Bug bounty platforms like BugCrowd, HackerOne, and Intigriti offer numerous draft reports for specific vulnerabilities. For HackerOne and Intigriti, you need to be authenticated on the platform to access these draft reports. You can do this by creating a new draft report for any program and selecting the vulnerability type. The platform will then automatically generate the overview and impact sections with draft information you can refine.

BugCrowd provides draft report templates hosted on GitHub, which you can access directly at BugCrowd Templates. These templates offer a structured starting point, helping you ensure that your reports are comprehensive and follow the platform's guidelines.

Thank You

Thank you for reading this style guide. I hope you found it helpful and that following these guidelines will lead to significant improvements in both your bounties and report writing skills. Please feel free to share this guide with your fellow hackers. If you have a few minutes, I would greatly appreciate your feedback on what I could have done better and which other topics you would like to see covered.

Here are my social links and email:

- Linkedin: https://www.linkedin.com/in/fuleki-ioan-503007268/
- Email: polyxena@bugcrowdninja.com
- My BugCrowd Profile: https://bugcrowd.com/Polyxena
- Blog: https://medium.com/@p0lyxena
- GitHub: https://github.com/P0lyxena