BUG BOUNTY TIPS #10

www.indiancybersecuritysolutions.com info@indiancybersecuritysolutions.com 1800-123-500014

MEMBER OF: NASSCOM® Indian Chamber of Commerce Confederation of Indian Industry DSCI PROMOTING DATA PROTECTION CERTIFIED: ISO 27001 ISO 9001

1

LIST OF 24 GOOGLE DORKS FOR BUG BOUNTIES

Here's a set of handy Google dorks for identifying accounts on various third party websites that could be related to our target company:

```
site:codepad.co "company"
site:scribd.com "company"
site:npmjs.com "company"
site:npm.runkit.com "company"
site:libraries.io "company"
site:ycombinator.com "company"
site:coggle.it "company"
site:papaly.com "company"
site:google.com "company"
site:trello.com "company"
site:prezi.com "company"
site:jsdelivr.net "company"
site:codepen.io "company"
site:codeshare.io "company"
site:sharecode.io "company"
site:pastebin.com "company"
site:repl.it "company"
site:productforums.google.com "company"
site:gitter.im "company"
site:bitbucket.org "company"
site:zoom.us inurl:"company"
site:atlassian.net "company"
site:s3.amazonaws.com inurl:"company"
inurl:gitlab "company"
```

Protip: There are number of projects that aim to automate the dorking process. Here are 3 very good ones:

```
https://github.com/A3h1nt/Grawler
https://github.com/Zarcolio/sitedorks
https://github.com/adnane-X-tebbaa/GRecon
```

2

WAF BYPASS DURING EXPLOITATION OF FILE UPLOAD

This is a very curious case of a WAF (Web Application Firewall) bypass discovered during exploitation of a file upload vulnerability. The '.php' file extension was blocked, but the author was able to bypass it using the following trick:

```
/?file=xx.php <-- Blocked
/?file===xx.php <-- Bypassed
```

It is unclear which particular WAF was deployed on the affected target site, but it might have been some kind of a custom solution using only a simple regex rule.

3 TURNING LFI TO RCE IN PHP USING ZIP WRAPPER

Now this is a really cool bug bounty tip for PHP based websites where (1) we can upload zip files and (2) we have found a LFI (Local File Inclusion) vulnerability.

Using PHP ZIP wrapper (zip://) we can leverage the LFI vulnerability and achieve RCE (Remote Code Execution) on the site. Here's how:

1. Create a .php file (rce.php)
2. Compress it to a .zip file (file.zip)
3. Upload your .zip file on the vulnerable web application
4. Trigger your RCE via accessing:

```
https://site.com/index.php?
page=zip://path/file.zip%23rce.php
```

The %23 value is the hash symbol (#), which serves as a delimiter between the archive filename and the filename inside the archive that we want to unzip.

Using this trick we have basically circumvented the file upload restrictions of the web application disallowing us to upload a php file directly.

Note that PHP supports all these compression wrappers:

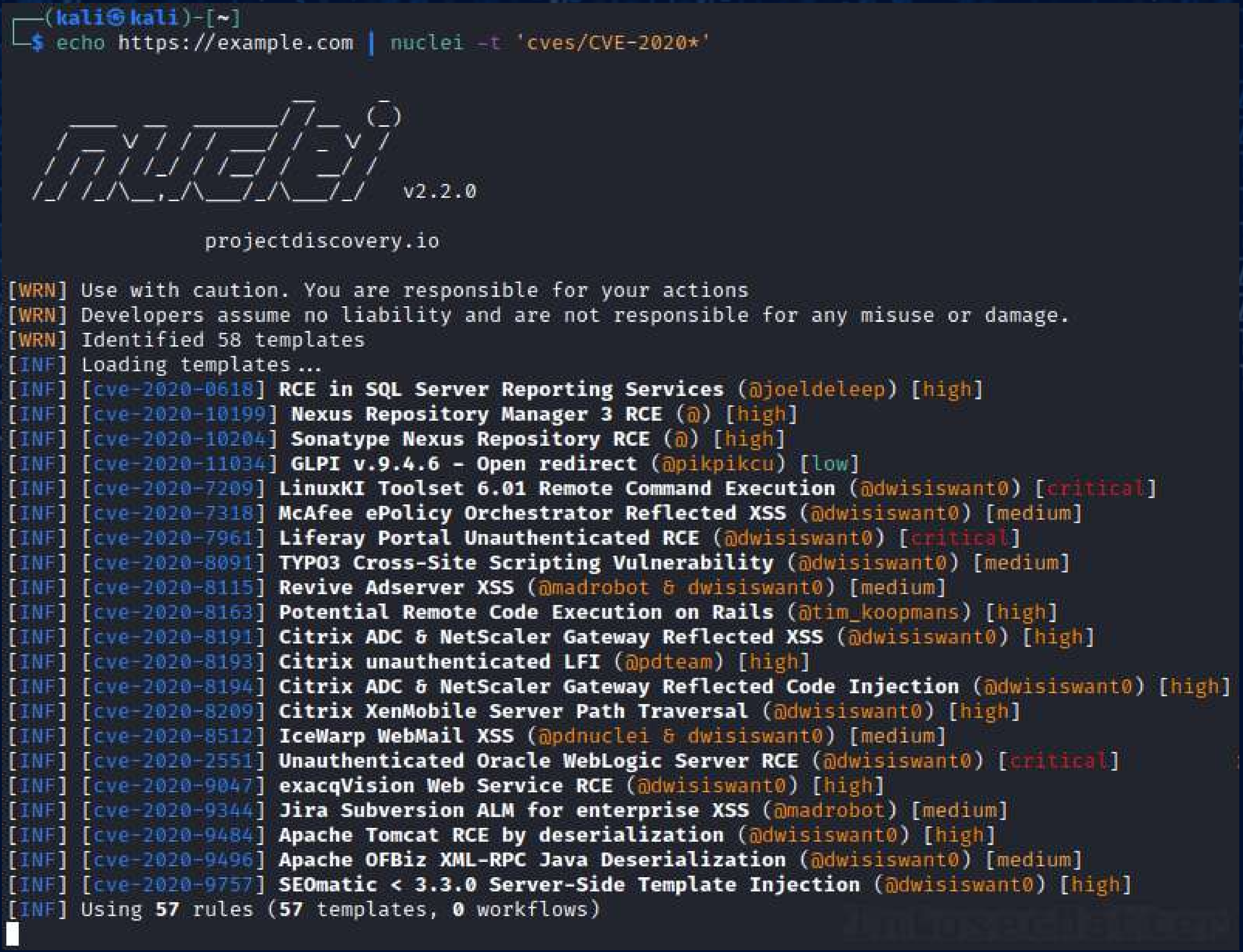
```
zlib:// example: compress.zlib://...
bzip2:// example: compress.bzip2://...
zip:// example: zip://archive.zip#dir/file.txt
rar:// example: rar://<url encoded archive name>[*][#[<url encoded entry name>]]
```

4

SEARCH FOR CVES OF SPECIFIC YEAR WITH NUCLEI

Did you know that you can easily search for particular CVEs with the Nuclei scan engine? Using wildcard glob support in Nuclei, you can quickly scan for CVEs of specific years. For example, the following command will scan for all the CVEs assigned in year 2020:

```
echo https://example.com | nuclei -t 'cves/CVE-2020*'
```



```
┌──(kali㉿kali)-[~]
└─$ echo https://example.com | nuclei -t 'cves/CVE-2020*'

                     __     _
   ____  __  _______/ /__  (_)
  / __ \/ / / / ___/ / _ \/ /
 / / / / /_/ / /__/ /  __/ /
/_/ /_/\__,_/\___/_/\___/_/   v2.2.0

                projectdiscovery.io

[WRN] Use with caution. You are responsible for your actions
[WRN] Developers assume no liability and are not responsible for any misuse or damage.
[WRN] Identified 58 templates
[INF] Loading templates ...
[INF] [cve-2020-0618] RCE in SQL Server Reporting Services (@joeldeleep) [high]
[INF] [cve-2020-10199] Nexus Repository Manager 3 RCE (@) [high]
[INF] [cve-2020-10204] Sonatype Nexus Repository RCE (@) [high]
[INF] [cve-2020-11034] GLPI v.9.4.6 - Open redirect (@pikpikcu) [low]
[INF] [cve-2020-7209] LinuxKI Toolset 6.01 Remote Command Execution (@dwisiswant0) [critical]
[INF] [cve-2020-7318] McAfee ePolicy Orchestrator Reflected XSS (@dwisiswant0) [medium]
[INF] [cve-2020-7961] Liferay Portal Unauthenticated RCE (@dwisiswant0) [critical]
[INF] [cve-2020-8091] TYPO3 Cross-Site Scripting Vulnerability (@dwisiswant0) [medium]
[INF] [cve-2020-8115] Revive Adserver XSS (@madrobot & dwisiswant0) [medium]
[INF] [cve-2020-8163] Potential Remote Code Execution on Rails (@tim_koopmans) [high]
[INF] [cve-2020-8191] Citrix ADC & NetScaler Gateway Reflected XSS (@dwisiswant0) [high]
[INF] [cve-2020-8193] Citrix unauthenticated LFI (@pdteam) [high]
[INF] [cve-2020-8194] Citrix ADC & NetScaler Gateway Reflected Code Injection (@dwisiswant0) [high]
[INF] [cve-2020-8209] Citrix XenMobile Server Path Traversal (@dwisiswant0) [high]
[INF] [cve-2020-8512] IceWarp WebMail XSS (@pdnuclei & dwisiswant0) [medium]
[INF] [cve-2020-2551] Unauthenticated Oracle WebLogic Server RCE (@dwisiswant0) [critical]
[INF] [cve-2020-9047] exacqVision Web Service RCE (@dwisiswant0) [high]
[INF] [cve-2020-9344] Jira Subversion ALM for enterprise XSS (@madrobot) [medium]
[INF] [cve-2020-9484] Apache Tomcat RCE by deserialization (@dwisiswant0) [high]
[INF] [cve-2020-9496] Apache OFBiz XML-RPC Java Deserialization (@dwisiswant0) [medium]
[INF] [cve-2020-9757] SEOmatic < 3.3.0 Server-Side Template Injection (@dwisiswant0) [high]
[INF] Using 57 rules (57 templates, 0 workflows)
█
```

All you need is to get Nuclei from here:

- https://github.com/projectdiscovery/nuclei

And then update the templates:

```
nuclei -update-templates
```

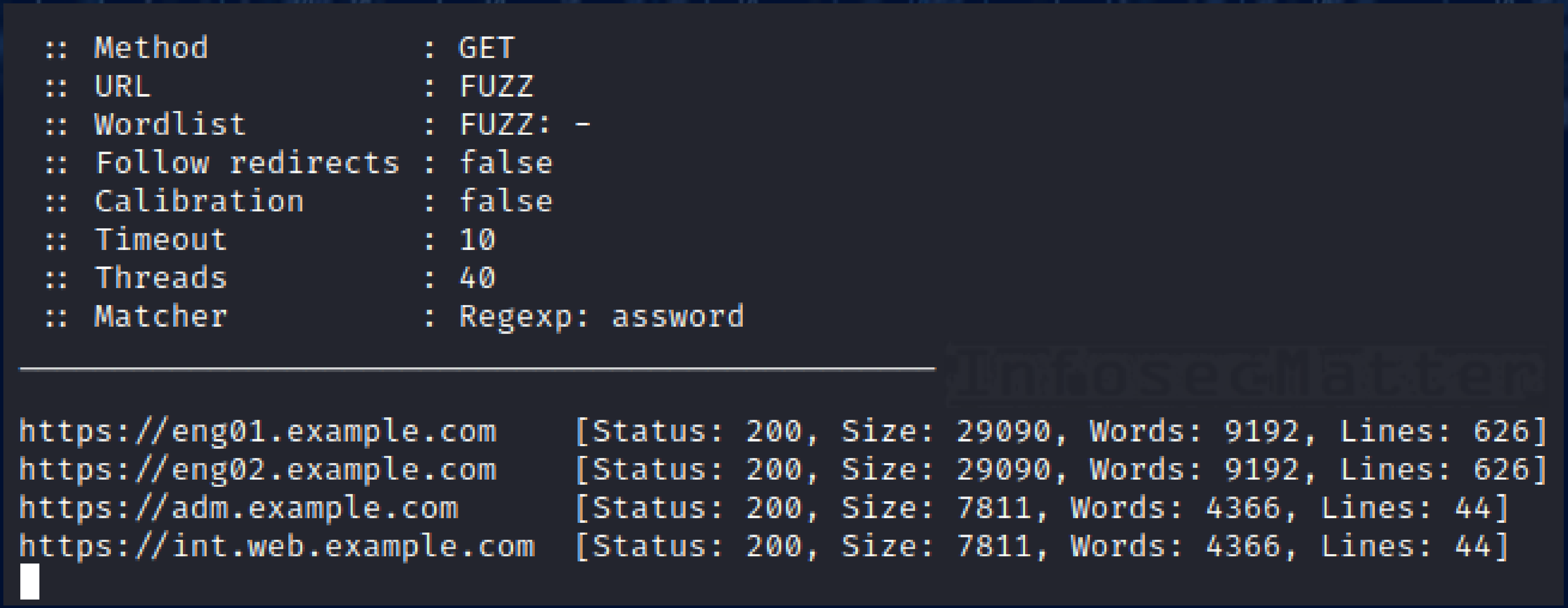
5 SEARCH FOR LOGIN PORTALS AND DEFAULT CREDS

Here's a useful and quick command combo to find actively running web servers from a list of hosts and identify login portals:

```
cat hosts.txt | httprobe -c 300 | ffuf -w - -u FUZZ -mr
"assword"
```

This is what the command does in detail:

1. Produce list of alive URLs from the provided list of hosts (domains or IP addresses)
2. Match URLs containing a login form (containing "assword" string somewhere on the page)



```
 :: Method           : GET
 :: URL              : FUZZ
 :: Wordlist         : FUZZ: -
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Regexp: assword
________________________________________________

https://eng01.example.com    [Status: 200, Size: 29090, Words: 9192, Lines: 626]
https://eng02.example.com    [Status: 200, Size: 29090, Words: 9192, Lines: 626]
https://adm.example.com      [Status: 200, Size: 7811, Words: 4366, Lines: 44]
https://int.web.example.com  [Status: 200, Size: 7811, Words: 4366, Lines: 44]
```

Now to search for default credentials for the identified portals / admin panels, we could use the default-http-login-hunter.sh script, which simply takes a list of URLs as a parameter.

The whole combo would then look like this:

```
cat hosts.txt | httprobe -c 300 | ffuf -w - -u FUZZ -mr
"assword" -s > login_portals.txt
default-http-login-hunter.sh login_portals.txt
```

6 HOW TO FIND ACCESS CONTROL BUGS

The following is practically a golden template on how to find access control bugs in web applications:

1. Firstly I have two users, one with high privileges (admin), and one with low privileges (joe). I log in as the admin first, and use all the functionality.
2. Whenever I do something that should be reserved for an administrator, I send the request over to Repeater.
3. Once I have a stack of them, I get the cookies from the “joe” account and insert them into those requests. I send them and analyze the difference in response to see if it worked. If it did, I report it!
4. Then, I do the same thing but without any cookies or session tokens to see if it works unauthenticated.

You would be amazed at how many applications appear totally secure, and then there’s just one or two endpoints that are vulnerable. This is because many web frameworks kind of suck at implementing roles and permissions. Web frameworks have been very successful at lessening the amount of injection vulnerabilities we see, but permissions still need to be defined by a human, so they’re more prone to errors.

7

AUTOMATED 403 FORBIDDEN BYPASSER TOOLS

There are number of known manual techniques on how to try to bypass 403 Forbidden errors when accessing restricted content (e.g. admin panels, backup folders, config files, logs etc.).
Here's a list of bug bounty tips that has been shared on this topic so far:

- `BBT4-5 – Access Admin panel by tampering with URI`
- `BBT4-6 – Bypass 403 Forbidden by tampering with URI`
- `BBT6-6 – Trick to access admin panel by adding %20`
- `BBT8-11 – Tips on bypassing 403 and 401 errors`
- `BBT9-1 – Bypass 403 errors by traversing deeper`

Now there is a new cool Burp Suite extension called 403Bypasser, which automates bypassing of 403 Forbidden errors.
If you don't have Burp Suite, you can also use shell scripts such as byp4xx.sh or bypass-403.sh, which work very similarly.

8 BYPASS WAF WITH UNICODE CHARACTERS

When you are looking for bugs like SSRF (Server Side Request Forgery), XSS (Cross Site Scripting), Open Redirect etc. and there is a blacklisted character, you can try to bypass it using an equivalent Unicode character.
The author found an Open Redirect vulnerability and found that the dot (.) character was blocked. He was able to bypass it using the Chinese dot (。) character “%E3%80%82”.

- `Here’s a PoC example:`
- `redirect_to=////evil%E3%80%82com`
- `This is equivalent to:`
- `redirect_to=////evil.com`

To find these alternate equivalent Unicode characters, just go to https://compart.com/en/unicode and search for the blacklisted character. You will find various Unicode characters to use for your testing.

9 LIST OF 48 OPEN REDIRECT PARAMETERS FROM HACKERONE

This is a massive list of 48 Open redirect parameters compiled from every disclosed HackerOne report ever, composed in one single wordlist (2020-11-30-open-redirect-params.txt):

```
/[redirect]
?targetOrigin=[redirect]
?fallback=[redirect]
?query=[redirect]
?redirection_url=[redirect]
?next=[redirect]
?ref_url=[redirect]
?state=[redirect]
?l=[redirect]
?redirect_uri=[redirect]
?forum_reg=[redirect]
?return_to=[redirect]
?redirect_url=[redirect]
?return_url=[redirect]
?host=[redirect]
?url=[redirect]
?redirectto=[redirect]
?return=[redirect]
?prejoin_data=[redirect]
?callback_url=[redirect]
?path=[redirect]
?authorize_callback=[redirect]
?email=[redirect]
?origin=[redirect]
?continue=[redirect]
?domain_name=[redirect]
?redir=[redirect]
?wp_http_referer=[redirect]
?endpoint=[redirect]
?shop=[redirect]
?qpt_question_url=[redirect]
?checkout_url=[redirect]
?ref_url=[redirect]
?redirect_to=[redirect]
?succUrl=[redirect]
?file=[redirect]
?link=[redirect]
?referrer=[redirect]
?recipient=[redirect]
?redirect=[redirect]
?u=[redirect]
?hostname=[redirect]
?returnTo=[redirect]
?return_path=[redirect]
?image=[redirect]
?requestTokenAndRedirect=[redirect]
?retURL=[redirect]
?next_url=[redirect]
```

The top 5 most popular are:

```
/[redirect]
?url=[redirect]
?next=[redirect]
?redirect_uri=[redirect]
?return_to=[redirect]
```

10 MASS ACCOUNT TAKEOVER VIA BAC

leveraging BAC (Broken Access Control) and achieving mass account takeover with a little bit of “security by obscurity” twist:

1. Password reset endpoint /api/u/resetPwd takes POST parameter username and sends email containing the password reset link
2. That “u” component in the API endpoint looked weird (u=user).
3. That means that there must exist an admin password reset API endpoint as well.
4. Tried /api/admin/resetPwd , /api/administrator/resetPwd , nothing worked.
5. Tried /api/su/resetPwd -> worked!
6. The above API would set new password as username+ab12*.
7. For example for username = admin, the password after reset would be adminab12*.
8. Done! Mass account takeover.

The biggest issue here was that the author discovered and admin API endpoint for resetting the password, which lacked any authorization checks.