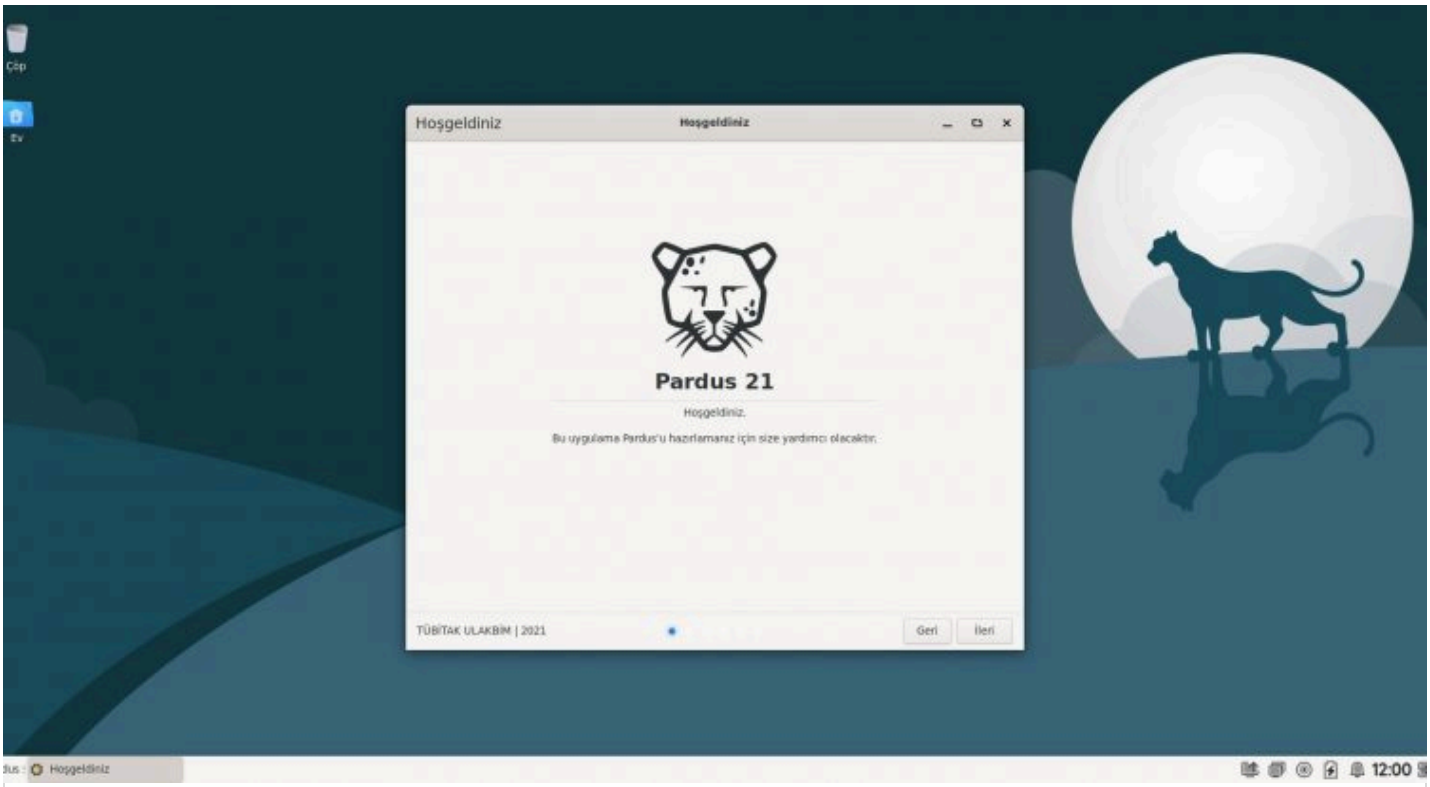


Pardus 21 Linux Distro – Remote Code Execution 0day 2021 CVE-2021-3806

September 13, 2021 (https://pentest.blog/pardus-21-linux-distro-remote-code-execution-0day-2021/) Mehmet Ince (https://pentest.blog/author/mehmet-ince/) Research (https://pentest.blog/category/research/)

A couple of days ago, I came up with news that Pardus will organize a report-bug contest. I love to contribute to open-source projects. So that was a pretty good chance to revisit one of my old friends, Pardus, and uncover security and/or privacy issues.

What is Pardus ?

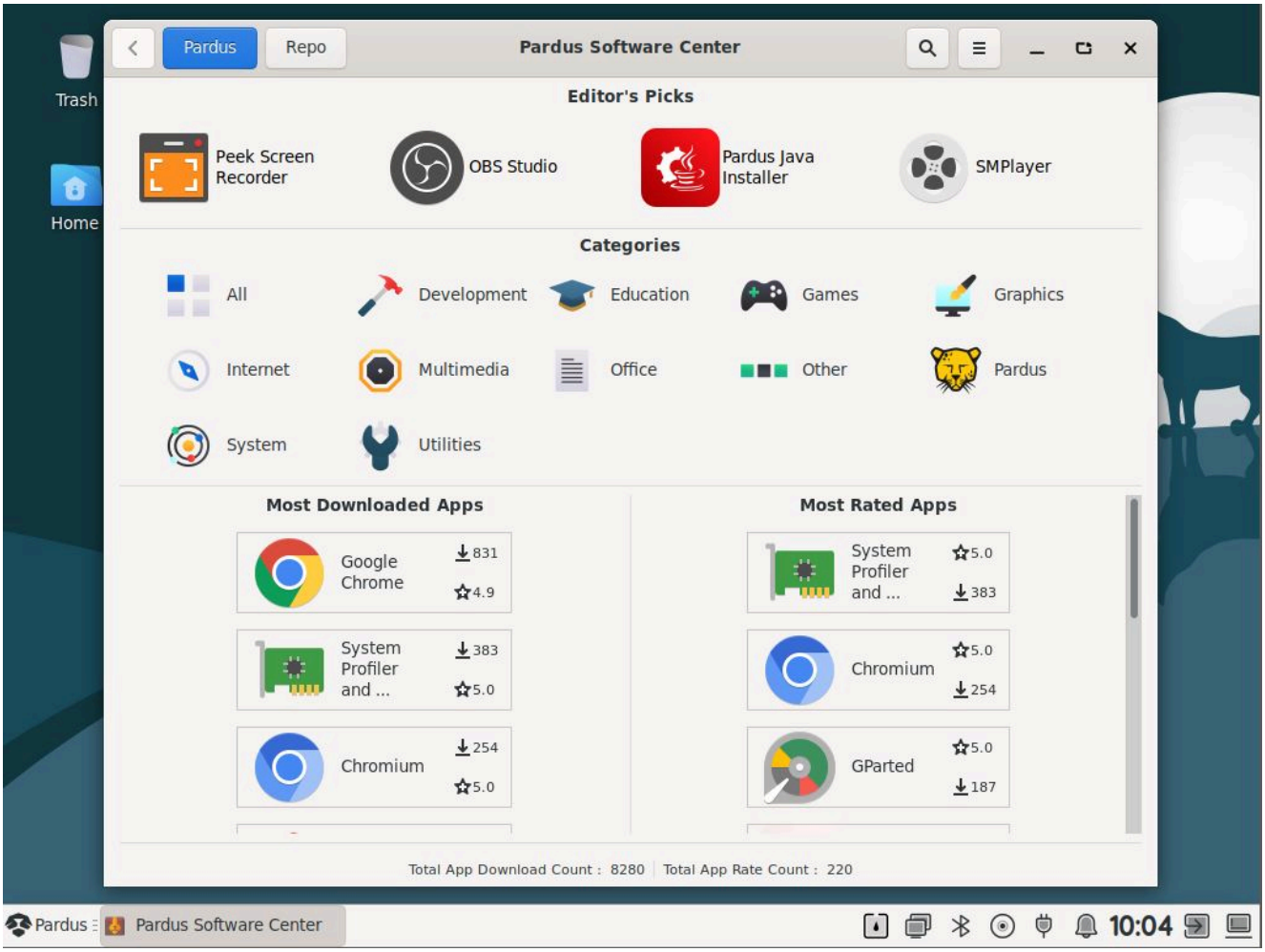
Pardus is a Linux distribution (https://en.wikipedia.org/wiki/Linux_distribution) developed with support from the government of Turkey (https://en.wikipedia.org/wiki/Turkey). Pardus' main focus is office-related work including use in Turkish government agencies (https://en.wikipedia.org/wiki/Government_agency). Despite that, Pardus ships in several languages. Its ease of use and availability free of charge has spawned numerous communities throughout the world.[1] (https://en.wikipedia.org/wiki/Pardus_(operating_system))

It started its life as a Gentoo-based project before developing its own unique identity. Since late 2012 the distribution is based on Debian GNU/Linux.

It's Debian Based Distro. So, where to Target?

I used to be one of the volunteers of Pardus back in 2009. It was an excellent and promising project back in those days. But Pardus has chosen to shift their Linux base to Debian. As far as I know, ever since 2012, they mainly develop packages around new core.

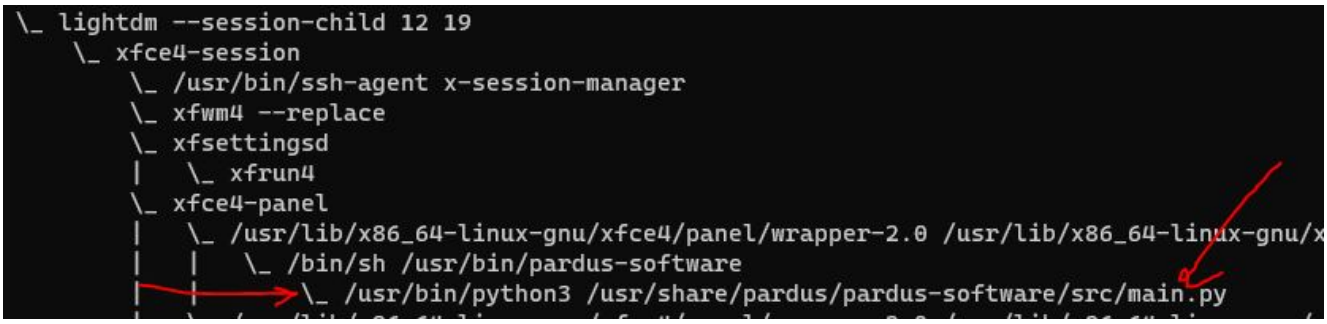
For that reason, I decided to focus on the GUI of the package manager without having a second thought.



Vulnerability Research

Installation of the Pardus 21 is pretty straightforward. But it's somewhat complicated to find the source-code of the packages developed by the Pardus team on the internet. I think they are still working on these types of issues while they work hard on releases.

Let's find where the `Pardus Software Center` . Click on `Pardus` icon on the bottom-left and then click on `Pardus Software Center` shortcut. That will initiate the process. That process has to be child process of the xfce4-session. We can easily locate path and the program with `ps auxf` .



As a next step, copy all the files under the target directory to the main host. So that we can start offensive code review.

```
┌──(root💀DESKTOP-I30B071)-[~/pardus-software]
└─# ls -al
total 196
drwxr-xr-x 2 root root   4096 Sep 13 09:18 .
drwx------ 4 root root   4096 Sep 13 09:18 ..
-rwxr-xr-x 1 root root   2707 Sep 13 09:18 Actions.py
-rw-r--r-- 1 root root   1719 Sep 13 09:18 AppDetail.py
-rw-r--r-- 1 root root   1674 Sep 13 09:18 AppImage.py
-rw-r--r-- 1 root root   1723 Sep 13 09:18 AppRequest.py
-rw-r--r-- 1 root root   1999 Sep 13 09:18 CellRendererButton.py
-rw-r--r-- 1 root root   2091 Sep 13 09:18 GnomeComment.py
-rw-r--r-- 1 root root   1094 Sep 13 09:18 GnomeRatingServer.py
-rw-r--r-- 1 root root 132924 Sep 13 09:18 MainWindow.py
-rw-r--r-- 1 root root   4974 Sep 13 09:18 Package.py
-rw-r--r-- 1 root root   7744 Sep 13 09:18 Server.py
-rw-r--r-- 1 root root   1869 Sep 13 09:18 UserSettings.py
-rw-r--r-- 1 root root     11 Sep 13 09:18 __version__
-rwxr-xr-x 1 root root    539 Sep 13 09:18 main.py
100% 1094     3.1MB/s
```

Vulnerability : Insecure Tar Extraction

While I was reviewing the project, one question has raised in my mind. Where are these app icons coming from? It is not possible to ship all icons via one Pardus package, is it?

Following function is taken from `MainWindows.py` file, between lines 706-712. So it looks like icons are coming from server.

```
    def getIcons(self):
        if self.Server.connection:
            # self.splashbar.pulse()
            print("Getting icons from server")
            self.splashlabel.set_markup("<b>{}</b>".format(_("Getting icons from server")))
            self.serverappicons = self.Server.getAppIcons()
            self.servercaticons = self.Server.getCategoryIcons()
```

Let’s see how does it fetch and use the icons. `getAppIcons()` is defined within Server.py file.

```
    def getAppIcons(self):
        if not self.isExists(self.cachedir + self.serverappicons):
            print("trying to downlad " + self.serverappicons)
            try:
```

```
5.                  response = requests.get(self.serverurl + self.serverfiles +
    self.serverappicons + self.serverarchive)
6.              except:
7.                  print(
8.                      "server error getting " + self.serverurl + self.serverfiles +
    self.serverappicons + self.serverarchive)
9.                  return False
10.             if response.status_code == 200:
11.                 if self.createDir(self.cachedir):
12.                     with open(self.cachedir + self.serverappicons +
    self.serverarchive, "wb") as file:
13.                         file.write(response.content)
14.                     if self.extractArchive(self.cachedir + self.serverappicons +
    self.serverarchive,
15.                                            self.serverappicons):
16.                         return True
17.                 return False
18.             else:
19.                 print("{} : {}".format("error getting app icons, status code",
    response.status_code))
20.                 return False
21.         else:
22.             return True
```

In a nutshell, Pardus Software Center fetches icons as an archive and extracts them on line 15. That sounds okay. But two questions I must ask:

1 – Any kind of network-level encryption ?

2 – How does it extract archives?

Let’s have a look at the class variables of the Server, to answer the first question. Actually, that shouldn’t have been a problem. Because it fetch a innocent archive file that contains lots of images.

```
1.  class Server(object):
2.      def __init__(self):
3.          self.serverurl = "http://store.pardus.org.tr"
4.          self.serverapps = "/api/v2/apps/"
5.          self.servercats = "/api/v2/cats/"
6.          self.serverhomepage = "/api/v2/homepage"
7.          self.serversendrate = "/api/v2/rate"
8.          self.serversenddownload = "/api/v2/download"
9.          self.serversendsuggestapp = "/api/v2/suggestapp"
10.         self.serverfiles = "/files/"
11.         self.serverappicons = "appicons"
12.         self.servercaticons = "categoryicons"
13.         self.serverarchive = ".tar.gz"
14.         self.serversettings = "/api/v2/settings"
15.         self.settingsfile = "serversettings.ini"
```

Let’s see how does it extract the archive file. The following function is taken from Server.py file, lines between 163-173.

```
1.      def extractArchive(self, archive, type):
2.          if not Path(self.cachedir + type).exists():
3.              try:
```

```
                tar = tarfile.open(archive)
                tar.extractall(path=self.cachedir)
                tar.close()
                return True
            except:
                print("tarfile error")
                return False
        return True
```

I do NOT see any validation on files names within the archive file. extractall() method of tarfile package does NOT restrict the file extraction to the different paths when the file has a directory traversal destination. (E.g: ../../../backdoor.sh) . The archive file is taken over HTTP traffic. You see where I'm going..

Exploitation

1 – Perform ARP poisoning attack in order to become Man In The Middle.

2 – When the Pardus user open Pardus Software Center application

3 – Following HTTP request will be generated by the app (http://store.pardus.org.tr/files/appicons.tar.gz (http://store.pardus.org.tr/files/appicons.tar.gz))

4 – Intercept the tar.gz file and replace it with your own tar.gz file that contains ../../.ssh/authorized_keys with a public key of your own ssh file.

5- extractAll() method will be executed with our own tar.gz file.

6 – When the insecure tar extraction is being done, your public ssh key will be stored under the user's home folder. That will give us the chance to ssh to the target !

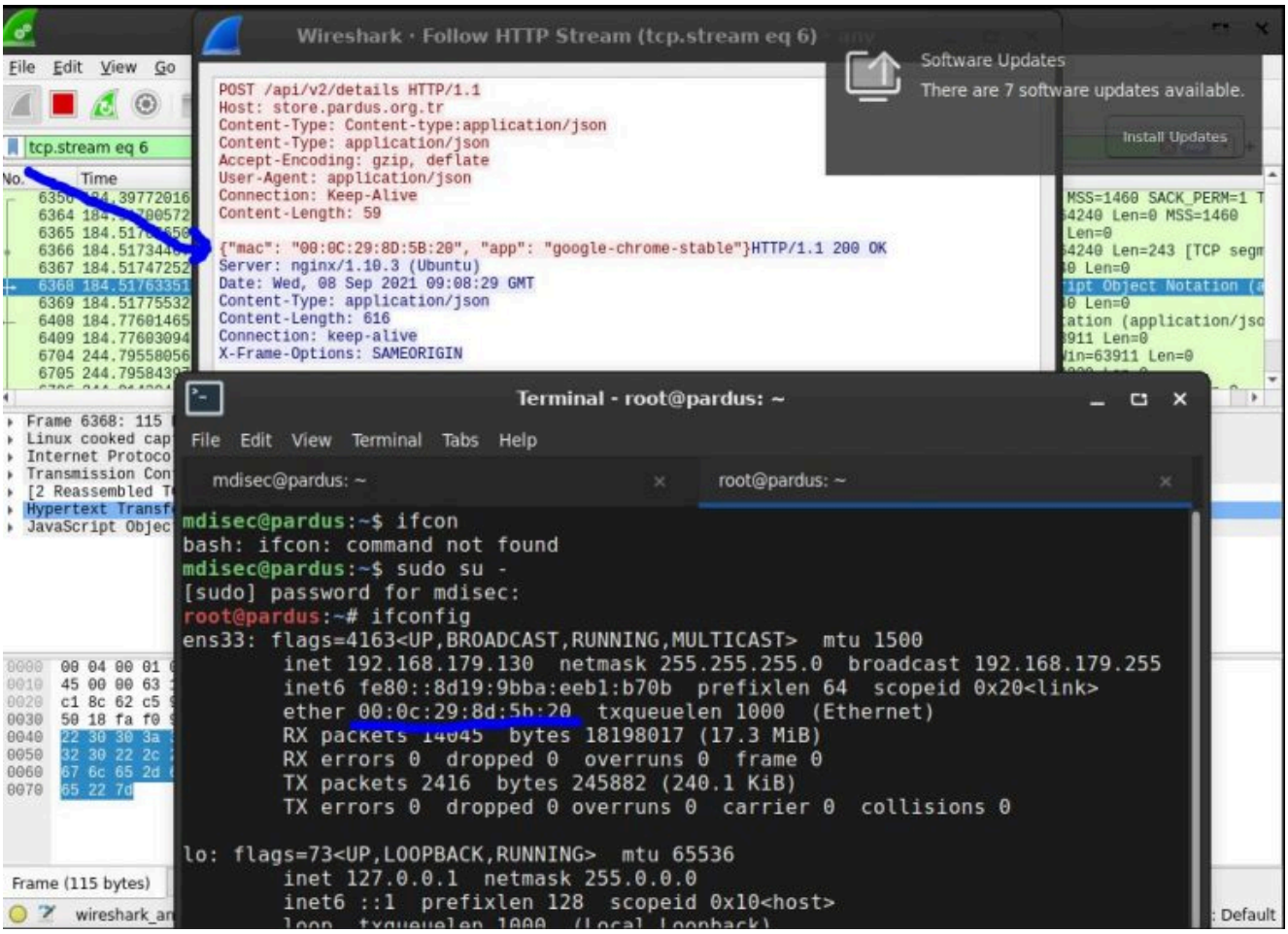
7 – SSH to the box your public key.

Proof of Concept

I've implemented a bash script, which automates all steps.

Bonus Finding: Privacy Issue : Mac Adress Leakage over HTTP

During the research, I've found that the Pardus Software Center is sending an HTTP POST request (to store.pardus.org.tr) containing the computer's MAC address when the user clicks on an app icon.



That issue also has been fixed along with the vulnerability.

Timeline

07 Sep 2021 01:15 AM – Research started

07 Sep 2021 02:10 AM – Vulnerabilities found.

07 Sep 2021 03:37 AM – PoC implemented and video record.

08 Sep 2021 12:00 AM – Report it to the Pardus team.

08 Sep 2021 13:15 PM – Vulnerability validated by the vendor.

09 Sep 2021 7:17 AM – Patch released.

(https://pentest.blog/tag/0day/) (https://pentest.blog/tag/advisory/)

(https://pentest.blog/tag/exploit/) (https://pentest.blog/tag/linux/)

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

Master Ninja @ Prodaft / INVICTUS Europe.

‹ 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/)

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

Search...

PRODAFT CYBER INTELLIGENCE AND CYBER SECURITY SERVICES

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

RECENT POSTS

Advisory | NetModule Router Software Race Condition Leads to Remote Code Execution (https://pentest.blog/advisory-netmodule-router-software-race-condition-leads-to-remote-code-execution/)

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/)

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

(https://pentest.blog/tag/0day/) (https://pentest.blog/tag/1day/)

(https://pentest.blog/tag/advisory/) (https://pentest.blog/tag/alienvault/)

(https://pentest.blog/tag/android/) (https://pentest.blog/tag/application/)

(https://pentest.blog/tag/assembly/) (https://pentest.blog/tag/bof/)

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

AWARDED TOP 15 PENTEST BLOG

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