Sri Lanka Institute of Information Technology

Assignment 1

M. P. D. M. Dias

IT19165530

MLB_WD_Y2S1_13.1

Point to Point Protocol Daemon RCE Vulnerability (CVE-2020-8597)

Systems and Network Programming– IE2012

Content

Introduction

The Point-to-Point Protocol (PPP) is a full-duplex protocol that allows simple data to be encapsulated and distributed through Layer 2 or data-link infrastructure spanning from dial-up connectivity through DSL broadband to virtual private networks (VPNs) incorporating SSL encryption. As these protocols do not allow point-to-point communications, PPP is also used to enforce IP and TCP over two directly connected nodes. Pppd is a daemon used on Unix-like operating systems to manage the establishment of PPP sessions and the termination of sessions between two nodes.

PPP is the protocol used to create internet connections over dial-up modems, DSL connections, and several other forms of point-to-peer connections via Virtual Private Networks (VPN), such as Peer to Point Tunneling Protocol (PPTP). The pppd program can also authenticate a peer connecting to the network and/or provide the peer with authentication details utilizing various authentication protocols like EAP.

Due to a flaw in the Point-to-Point Protocol Daemon (pppd) handling of the Extensible Authentication Protocol (EAP) packet, an unauthenticated remote attacker may cause a stack buffer overflow which may allow arbitrary execution of code on the target system. This weakness is triggered by an mistake in validating the input size before copying the data supplied to memory. Provided that the validity of the data size is wrong, random data may be copied into memory which can trigger file leakage contributing to unintended code execution.

The weakness lies in the eap parsing code logic, explicitly in the eap request) (and eap response) (functions in eap.c that a network input handler calls. These functions, using the first byte as a type, take a pointer and length as input. If the form is EAPT MD5CHAP(4), then it looks at an embedded area of 1-byte length. The rationale in this code is intended to make sure embedded duration is less than the total length of the packet. Following this verification, it tries to copy the provided data (hostname), which is located in a local stack buffer after the embedded length field. This boundary check is incorrect and allows memory copying to occur with an arbitrary data length.

An additional logical error causes the eap input) (function not to test whether EAP has been resolved during the Line Control Protocol (LCP) process. This allows an unauthenticated attacker to send an EAP packet even if ppp refused to negotiate authentication due to lack of support for EAP or due to a non-compliance with an agreed pre-shared passphrase in the LCP stage. In eap input the insecure pppd code must still process the EAP packet and cause the stack buffer overflow. This unverified, unknown-size data can be used to compromise goal device memory. The pppd also runs with high privileges (system or root), and operates with kernel drivers in tandem.

The pppd program is also used with the LWIP (lightweight IP) project to provide pppd capability with tiny computers. The default lwIP installation and installations are not susceptible to this buffer overload. However, if used the lwIP source code and modified it explicitly to allow EAP at compile time, the program can be susceptible to buffer overflow.

CVE-2020-8597 is a buffer overflow bug in pppd owing to a conceptual defect in the Extensible Authentication Protocol (EAP) packet processor. An unauthorized remote attacker who sends a specially crafted EAP packet to a vulnerable PPP client or server may cause a denial of service condition or an arbitrary code execution. As pppd operates in tandem with kernel drivers and also has high privileges, such as device or even core, any code execution may also be performed with the same privileges.

Referenced on who found the vulnerability

Discovered by IOActive Protection Researcher Ilja Van Sprundel, the crucial problem is a stack buffer overflow flaw that occurs due to a logical error in the Extensible Authentication Protocol (EAP) module parser of pppd applications, an enhancement that offers support for additional authentication methods in PPP connections.

The weakness, monitored as CVE-2020-8597 with CVSS Score 9.8, can be abused by unauthorized attackers to remotely execute arbitrary code on affected devices and gain complete control of them.

How it was found

This weakness is attributed to an mistake in validating the size of the input before transferring the data to memory. Because data size validation is wrong, random data can be copied to memory and can trigger database fragmentation, likely contributing to the execution of unauthorized code.

The vulnerability is found in the logic of the eap parsing code, specifically in eap request) (and eap response) (functions in eap.c, which are called by the network input handler.

It is wrong to conclude that pppd is not insecure if EAP is not allowed or if EAP has not been initiated by a remote peer using a password or passphrase. This is because an authenticated intruder may always be able to submit an unsolicited EAP packet to induce a buffer overflow.

The vulnerability has been identified in the Point-to-Point Protocol (PPP) daemon, or pppd. PPP is a Layer 2 protocol used to establish connections over dial-up modems, DSL connections, and many other physical networks, including mobile networks. PPP has been included in and expanded to include additional protocols, such as the Point-to-Point Tunneling Protocol (PPTP) that is used in Virtual Private Networks (VPNs) to provide encrypted connections.

Throughout this situation, the SEI CERT collaboration team partnered with protection analyst Ilja Van Sprundel (IOActive) who found this flaw and software developer Paul Mackerras (OZlabs) who manages the source code to easily examine the issue and find a workaround. The issue involved a buffer overflow in the pppd source code owing to a basic buffer overflow in the Boolean expression and the implementation of the conditional statements that resulted. The sentence below can be fooled into allowing unknown duration feedback and copying it to a stack buffer. That is commonly referred to as frame overload or stack buffer overflow.

if (vallen >= len + sizeof(rhostname)) { // Copy to buffer rhostname

The fix for the vulnerability was to simply change the above statement to the Boolean logic below.

if (len-vallen >= sizeof(rhostname)) { // Copy to buffer rhostname

Paul issued CVE-2020-8597 for this flaw and continued to repair it to the source code he managed. The system update required to correct the bug is a minor one that requires just a few lines of code. Nevertheless, this insecure technology resides in thousands of libraries of software projects. It has been adopted by more than 100 companies offering network access devices spanning from home routers to business network hardware. As this weakness affects all PPP clients and servers, it also affects Internet Service Providers (ISPs).

When it was found

On 2020 March 4, researchers at the CERT Coordination Center (CERT / CC) published vulnerability note # 782301 for critical vulnerability in Point-to-Point Protocol Daemon (pppd) versions 2.4.2 through 2.4.8, with disclosure credited to IOActive's Ilja van Sprundel.

What is the damage that it can cause

Through submitting an unsolicited EAP packet to a vulnerable ppp client or server, an unauthorized remote intruder can trigger memory corruption in the pppd mechanism, which may require arbitrary code execution.

According to the researcher, Point-to-Point Protocol Daemon versions 2.4.2 to 2.4.8—all versions published in the last 17 years — are susceptible to this new remote code execution bug.
Any of the commonly used, successful Linux distributions mentioned below have already been reported impacted, and several other projects are most likely affected as well.

Debian
Ubuntu
SUSE Linux
Fedora
NetBSD
Red Hat Enterprise Linux

In addition, the number of other susceptible applications and devices (some of which are mentioned below) that ship pppd apps is also likely to be vast, providing a broad attack surface for hackers.

Cisco CallManager
TP-LINK products
OpenWRT Embedded OS
Synology products

Due to a flaw in the Extensible Authentication Protocol (EAP) packet processing in the Point-to-Point Protocol Daemon (pppd), an unauthenticated remote attacker may be able to cause a stack buffer overflow, which may allow arbitrary code execution on the target system. This vulnerability is due to an error in validating the size of the input before copying the supplied data into memory. As the validation of the data size is incorrect, arbitrary data can be copied into memory and cause memory corruption possibly leading to execution of unwanted code.

What are exploitation techniques

The critical issue is a stack buffer overflow vulnerability that exists due to a logical error in the Extensible Authentication Protocol (EAP) packet parser of the pppd software, an extension that provides support for additional authentication methods in PPP connections.

For this, all an attacker needs to do is to send an unsolicited malformed EAP packet to a vulnerable ppp client or a server over a direct serial link, ISDN, Ethernet, SSH, socket CAT, PPTP, GPRS, or ATM networks.
Additionally, since pppd often runs with high privileges and works in conjunction with kernel drivers, the flaw could allow attackers to potentially execute malicious code with the system or root-level privileges.

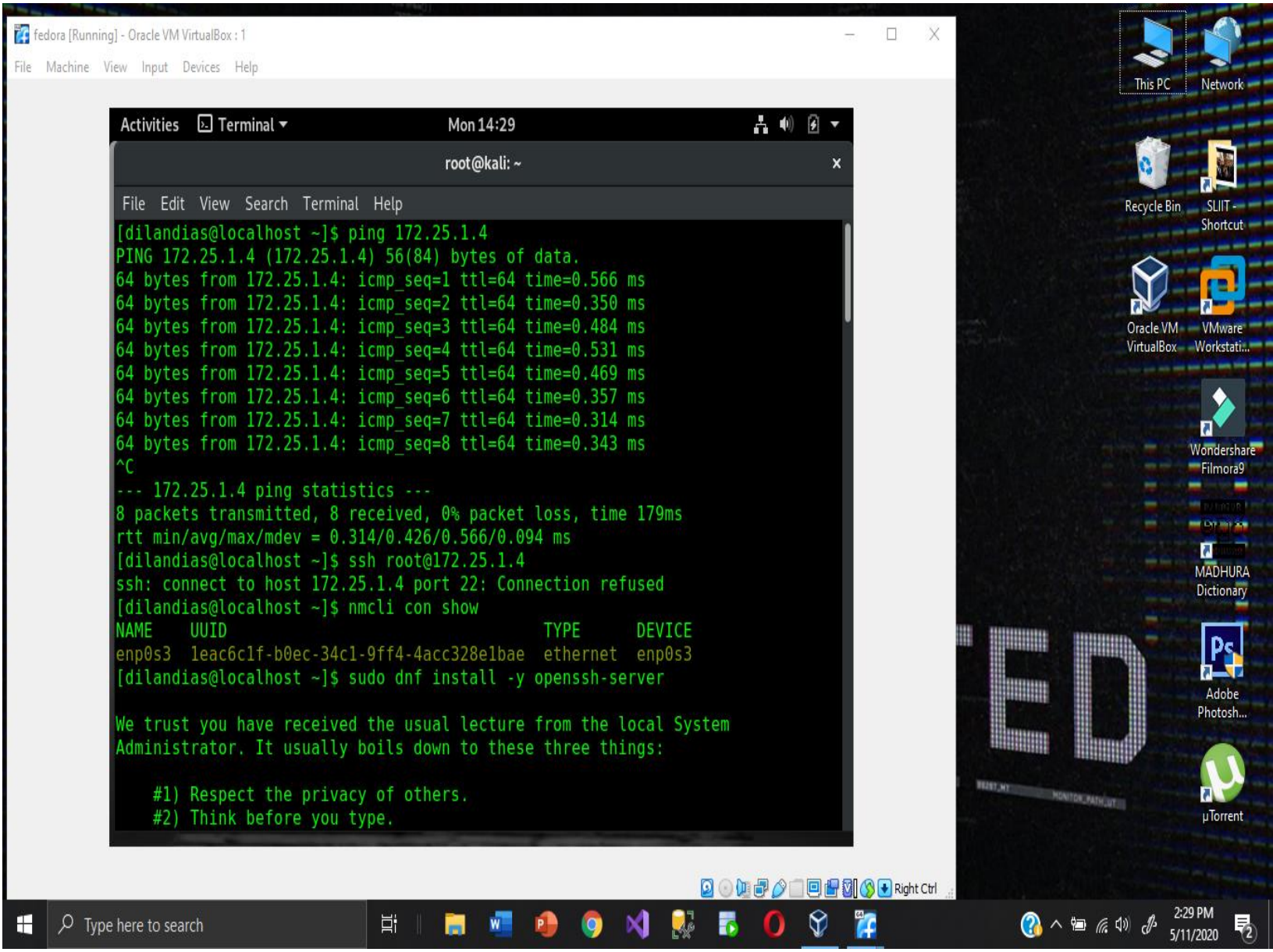
What exploit method you chose

Remote code execution method use to exploit the vulnerable client. Remote code execution (RCE) relates to the capacity of a cyber intruder to enter and make modifications to a device controlled by someone, without permission and unaware of where the machine is situated. RCE allows an attacker to take over a computer or server by running arbitrary malware (malware) software.

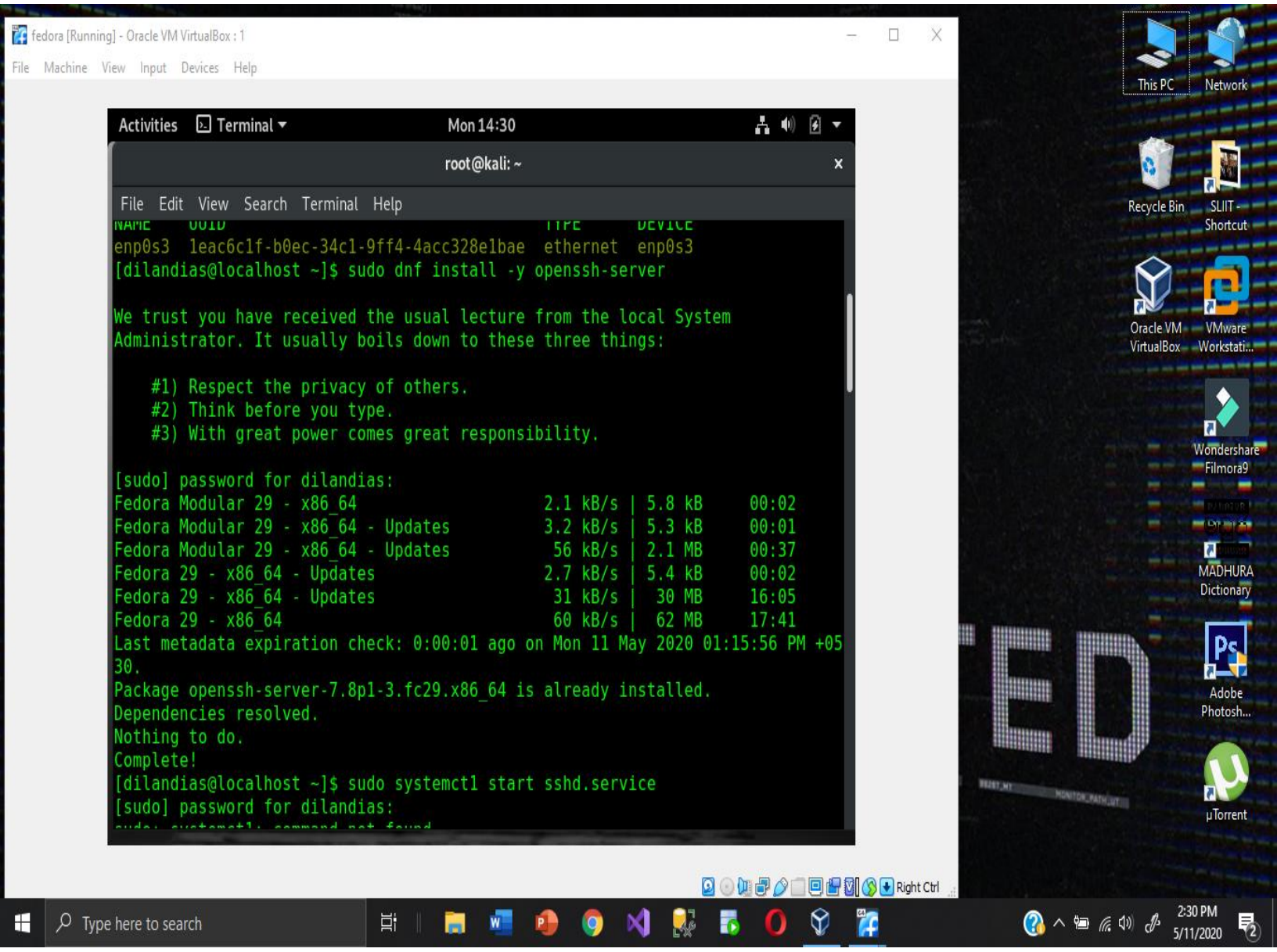
I use two virtual machines to test on the same computer. One as server and one as client. To connect virtual machines I install ssh. Using their ip addresses I connect Fedora 29 virtual machine as the server side and Kali Linux virtual machine as the vulnerable client side. As per the instructions I set up a pppoe-server. After open the debug mode and set log file and add the following to the log file '/etc/ppp/pppoe-server-options'. Then set up a pppoe-client by enter 'sudo pppoeconf'. Finally using the python code can exploit the vulnerable client. Also by sending an unsolicited EAP packet to a vulnerable ppp client , remote attacker could cause memory corruption in the pppd process, which may allow for arbitrary code execution.

Screenshots of the exploit

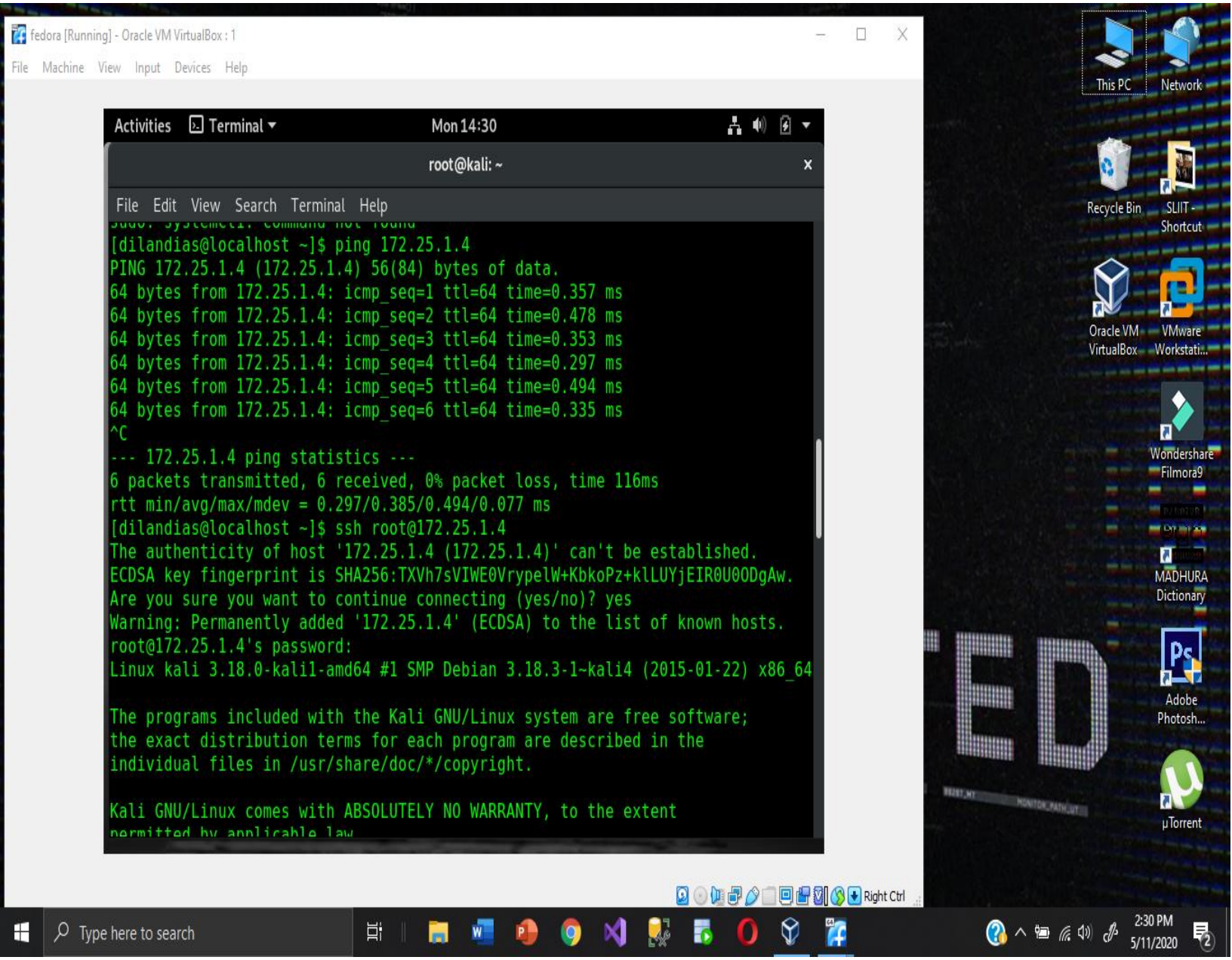
- **Ping with vulnerable client**



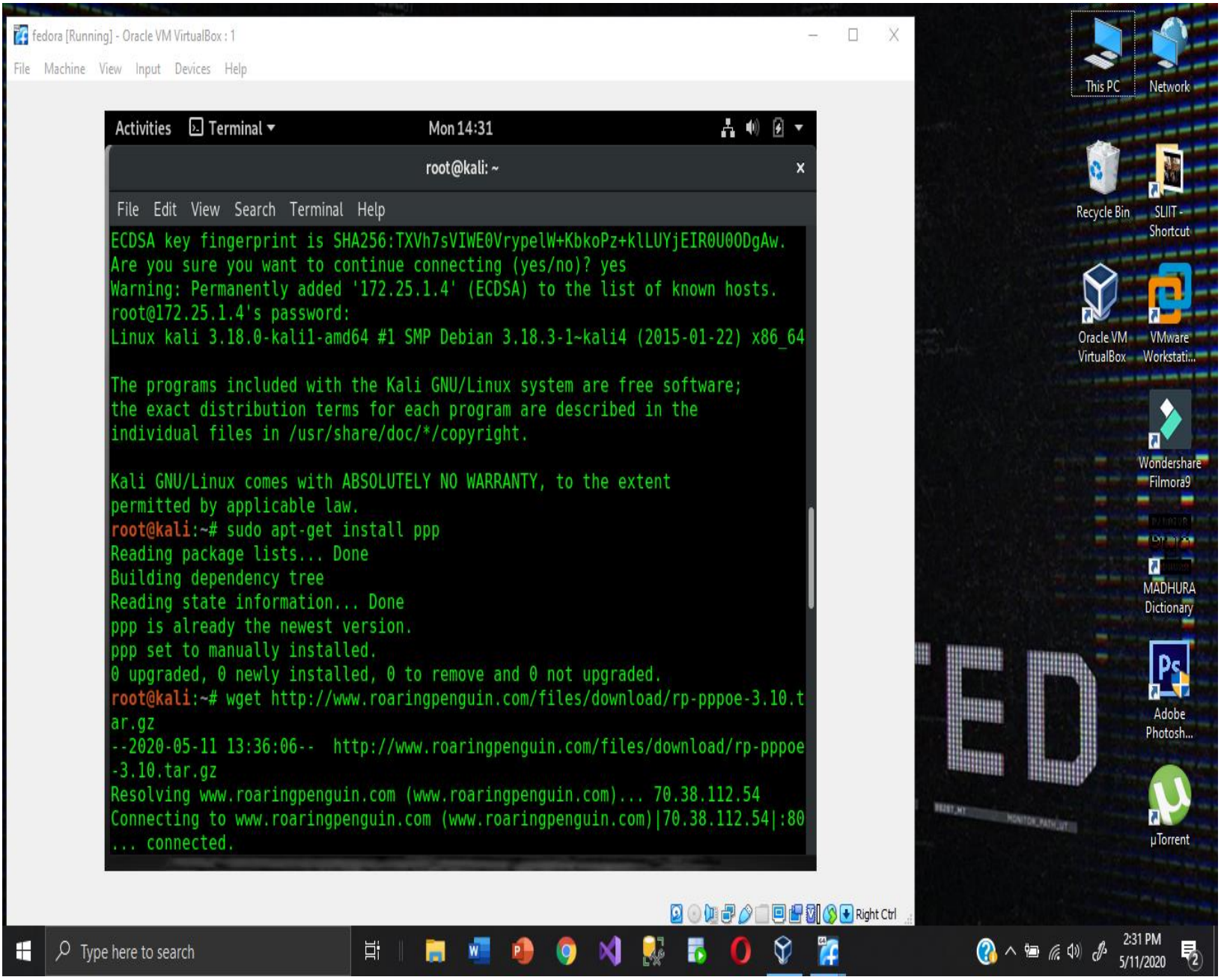
- **Install SSH to the server**



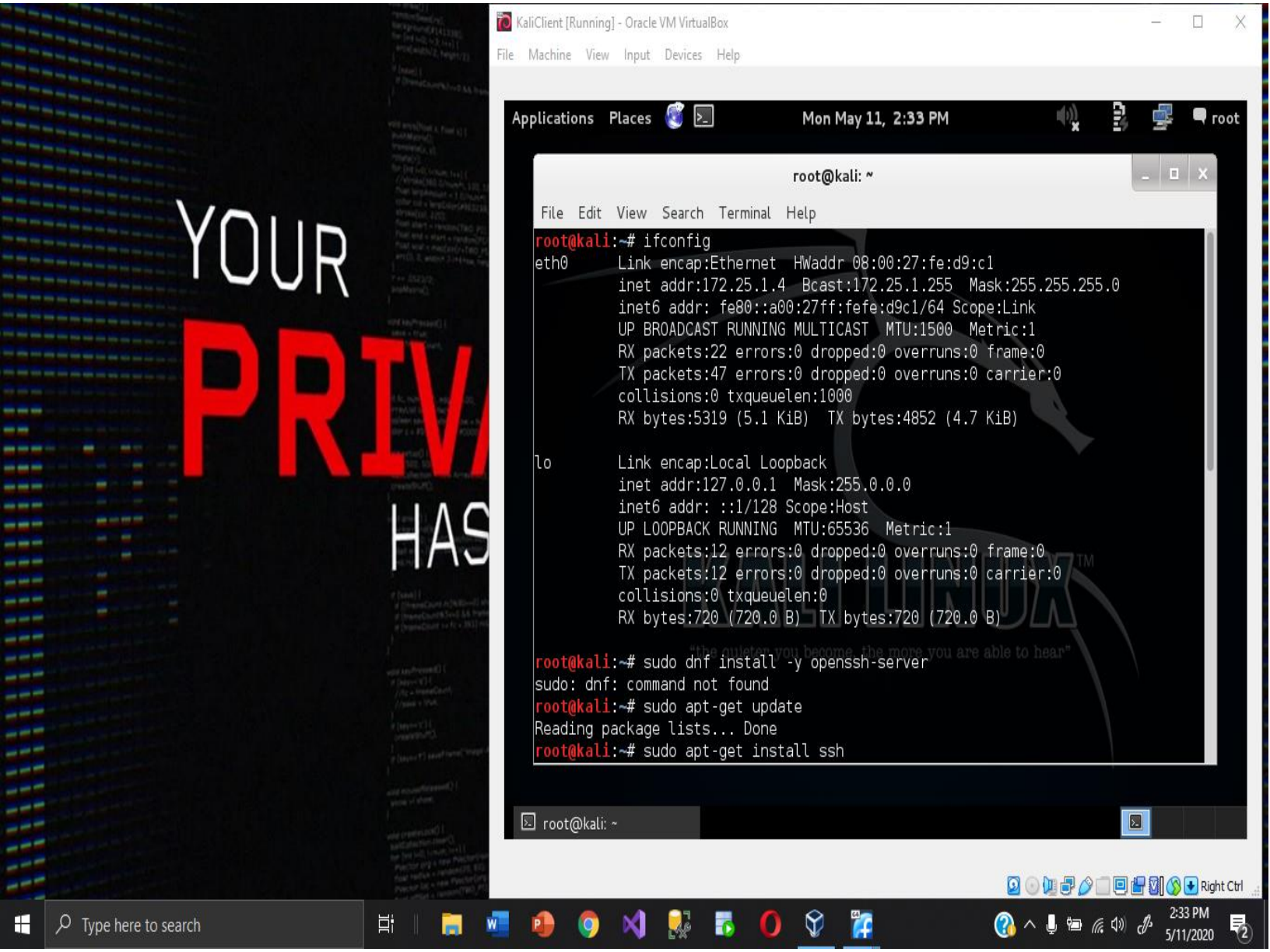
- **Getting root access to the client side**

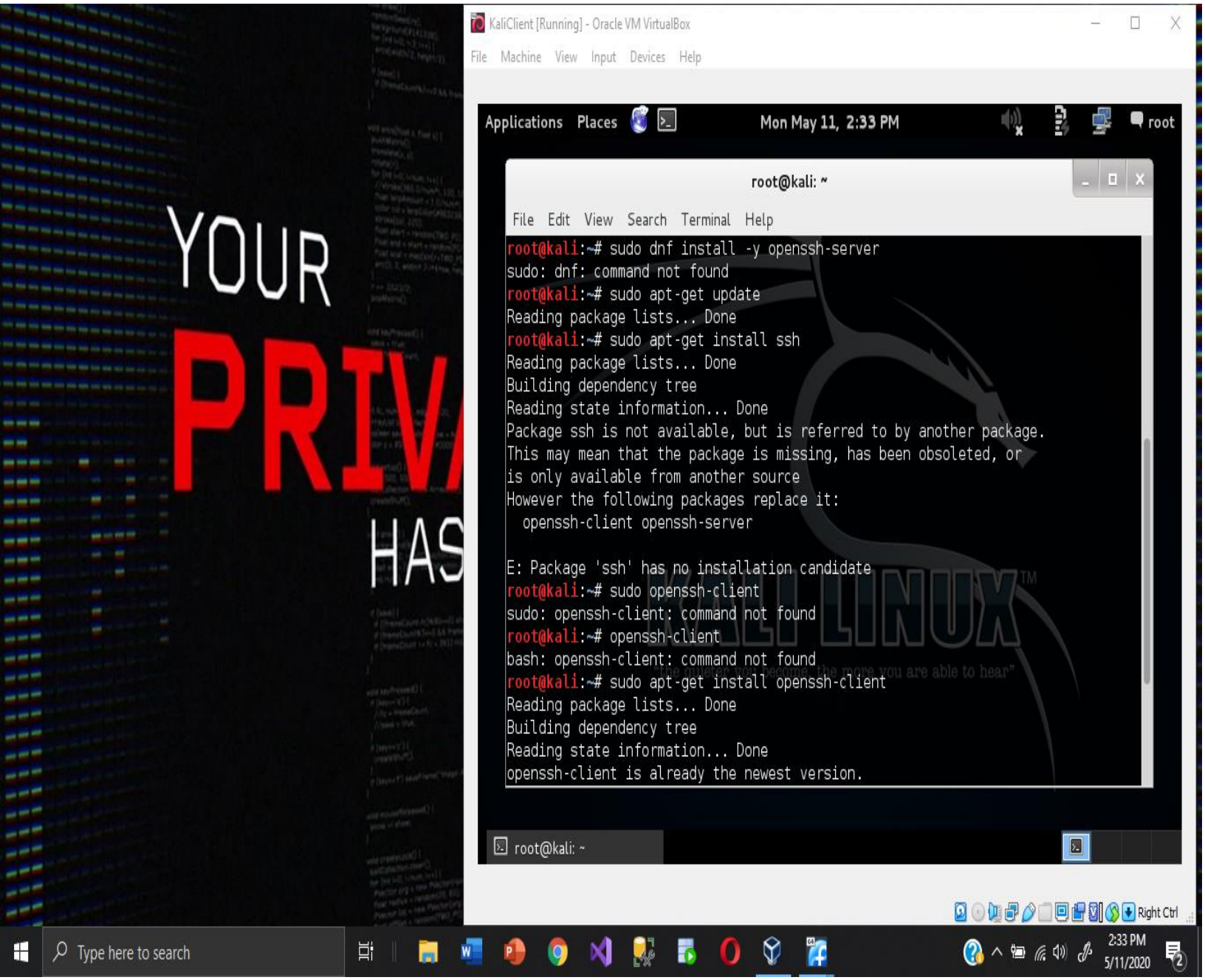


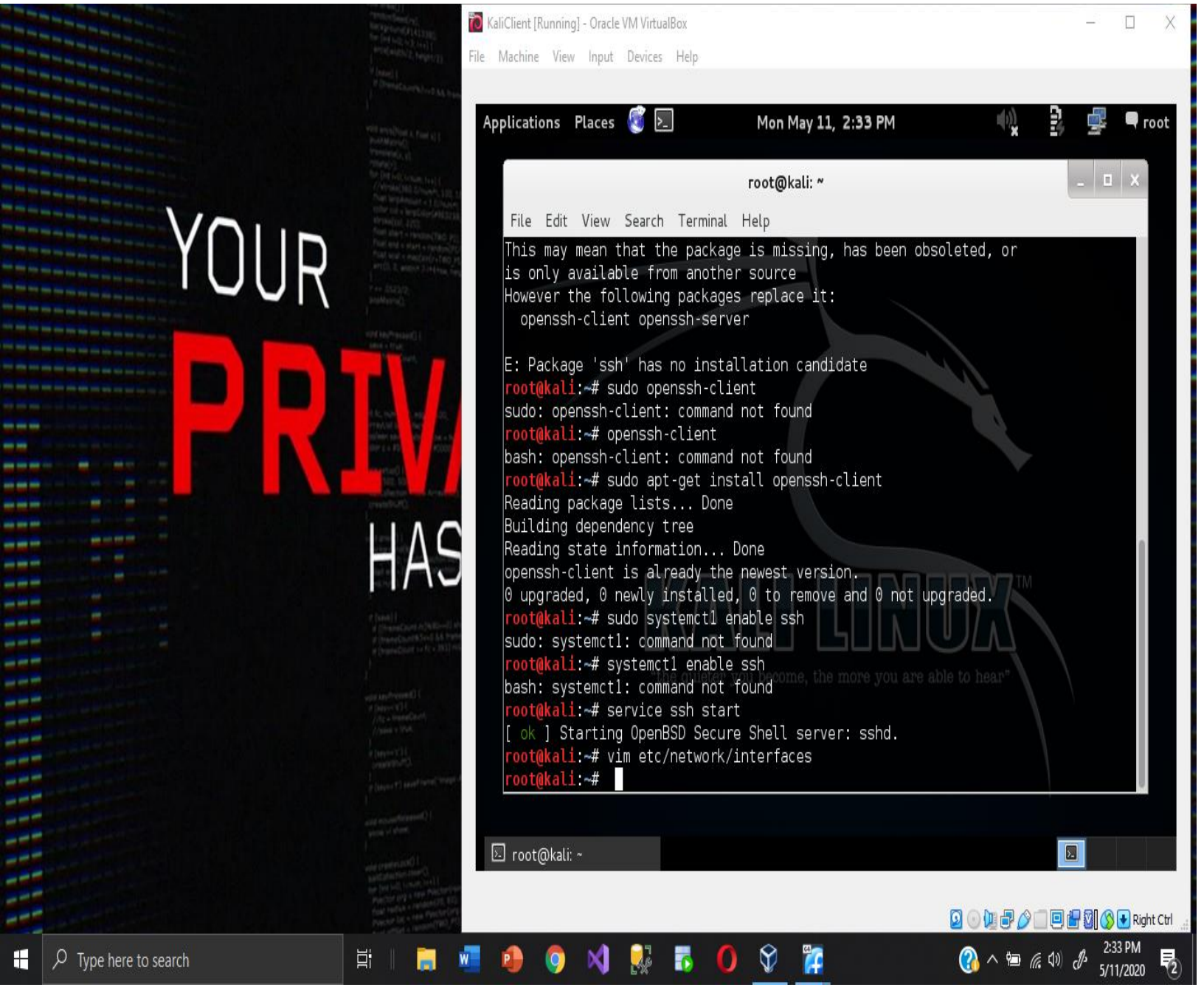
- **After getting client's root access**



- **Enable SSH in client side**




YOUR
PRIV
HAS
KaliClient [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places
Mon May 11, 2:33 PM
root
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# sudo dnf install -y openssh-server
sudo: dnf: command not found
root@kali:~# sudo apt-get update
Reading package lists... Done
root@kali:~# sudo apt-get install ssh
Reading package lists... Done
Building dependency tree
Reading state information... Done
Package ssh is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source
However the following packages replace it:
openssh-client openssh-server
E: Package 'ssh' has no installation candidate
root@kali:~# sudo openssh-client
sudo: openssh-client: command not found
root@kali:~# openssh-client
bash: openssh-client: command not found
root@kali:~# sudo apt-get install openssh-client
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-client is already the newest version.
root@kali: ~
Right Ctrl
Type here to search
2:33 PM
5/11/2020

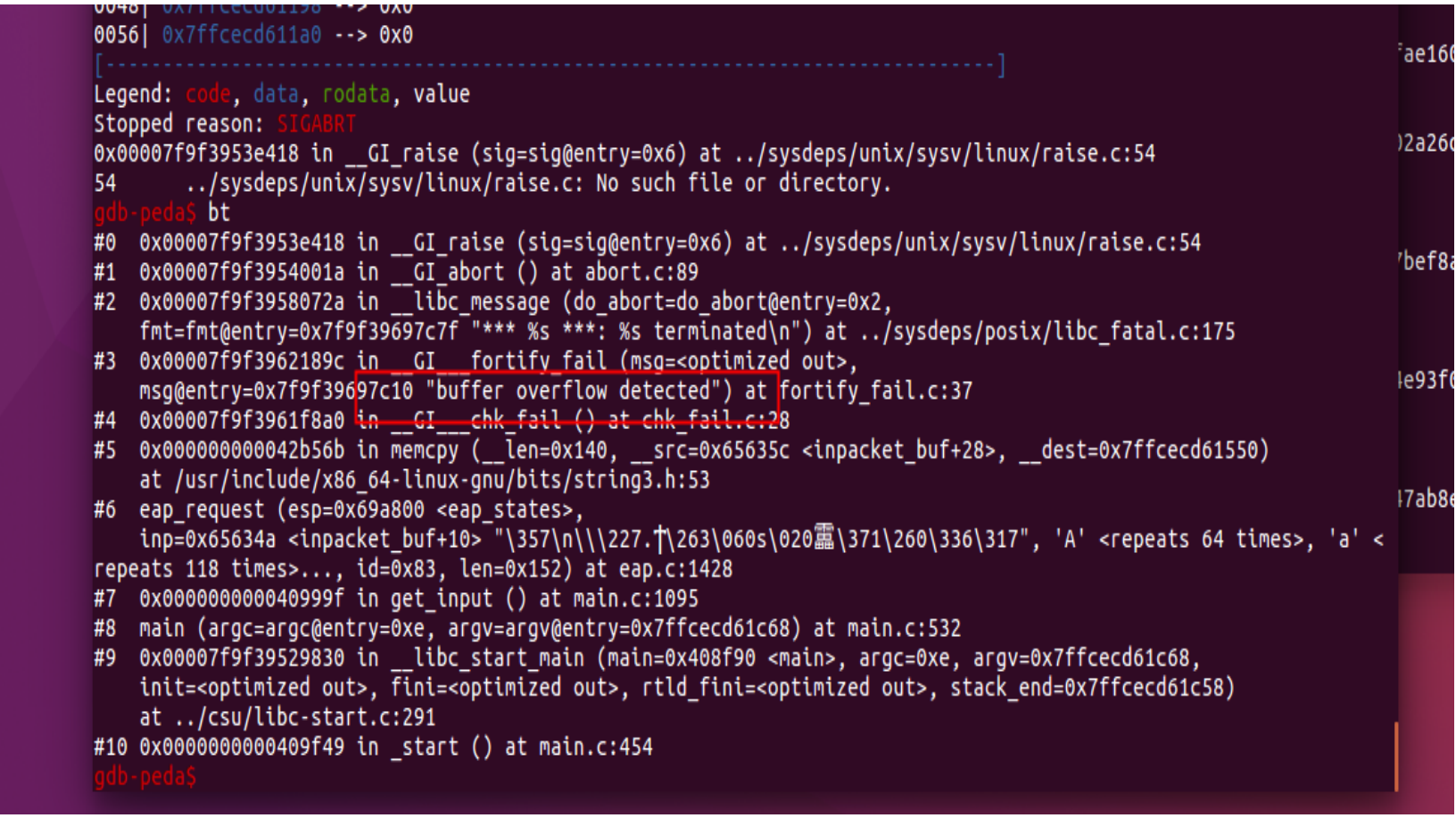

YOUR
PRIV
HAS
KaliClient [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places
Mon May 11, 2:33 PM
root
root@kali: ~
File Edit View Search Terminal Help
This may mean that the package is missing, has been obsoleted, or
is only available from another source
However the following packages replace it:
openssh-client openssh-server
E: Package 'ssh' has no installation candidate
root@kali:~# sudo openssh-client
sudo: openssh-client: command not found
root@kali:~# openssh-client
bash: openssh-client: command not found
root@kali:~# sudo apt-get install openssh-client
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-client is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@kali:~# sudo systemctl enable ssh
sudo: systemctl: command not found
root@kali:~# systemctl enable ssh
bash: systemctl: command not found
root@kali:~# service ssh start
[ok] Starting OpenBSD Secure Shell server: sshd.
root@kali:~# vim etc/network/interfaces
root@kali:~#
root@kali: ~
Right Ctrl
Type here to search
2:33 PM
5/11/2020

- ***Crash***

```
rcvd [CCP ConfAck id=0x2]
rcvd [IPCP ConfAck id=0x2 <addr 192.168.1.254>]
not replacing existing default route via 192.168.134.2 with metric -1
local  IP address 192.168.1.254
remote IP address 192.168.1.1
Script /etc/ppp/ip-up started (pid 88518)
Script /etc/ppp/ip-up finished (pid 88518), status = 0x0

rcvd [EAP Request id=0x83 MD5-Challenge <Value ef 0a 5c 97 2e cf ae b3 30 73 10 e9 9d 81 f9 b0 de cf> <Name "AAAAAAAAAAAAAAAAAAAAA
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
*** buffer overflow detected ***: pppd terminated
======= Backtrace: =========
/lib/x86_64-linux-gnu/libc.so.6(+0x777e5)[0x7f6d3c8cc7e5]
/lib/x86_64-linux-gnu/libc.so.6(__fortify_fail+0x5c)[0x7f6d3c96e15c]
/lib/x86_64-linux-gnu/libc.so.6(+0x117160)[0x7f6d3c96c160]
pppd[0x42b56b]
pppd(main+0xa0f)[0x40999f]
/lib/x86_64-linux-gnu/libc.so.6(__libc_start_main+0xf0)[0x7f6d3c875830]
pppd(_start+0x29)[0x409f49]
======= Memory map: ========
00400000-00449000 r-xp 00000000 08:01 1457155                            /usr/sbin/pppd
00648000-00649000 r--p 00048000 08:01 1457155                            /usr/sbin/pppd
00649000-0064f000 rw-p 00049000 08:01 1457155                            /usr/sbin/pppd
0064f000-0069b000 rw-p 00000000 00:00 0
01dd9000-01dfa000 rw-p 00000000 00:00 0                                  [heap]
7f6d3bdff000-7f6d3be15000 r-xp 00000000 08:01 1316027                    /lib/x86_64-linux-gnu/libgcc_s.so.1
7f6d3be15000-7f6d3c014000 ---p 00016000 08:01 1316027                    /lib/x86_64-linux-gnu/libgcc_s.so.1
7f6d3c014000-7f6d3c015000 rw-p 00015000 08:01 1316027                    /lib/x86_64-linux-gnu/libgcc_s.so.1
7f6d3c015000-7f6d3c020000 r-xp 00000000 08:01 1316327                    /lib/x86_64-linux-gnu/libnss_files-2.23.so
7f6d3c020000-7f6d3c21f000 ---p 0000b000 08:01 1316327                    /lib/x86_64-linux-gnu/libnss_files-2.23.so
7f6d3c21f000-7f6d3c220000 r--p 0000a000 08:01 1316327                    /lib/x86_64-linux-gnu/libnss_files-2.23.so
7f6d3c220000-7f6d3c221000 rw-p 0000b000 08:01 1316327                    /lib/x86_64-linux-gnu/libnss_files-2.23.so
7f6d3c221000-7f6d3c227000 rw-p 00000000 00:00 0
7f6d3c227000-7f6d3c232000 r-xp 00000000 08:01 1316331                    /lib/x86_64-linux-gnu/libnss_nis-2.23.so
7f6d3c232000-7f6d3c431000 ---p 0000b000 08:01 1316331                    /lib/x86_64-linux-gnu/libnss_nis-2.23.so
7f6d3c431000-7f6d3c432000 r--p 0000a000 08:01 1316331                    /lib/x86_64-linux-gnu/libnss_nis-2.23.so
7f6d3c432000-7f6d3c433000 rw-p 0000b000 08:01 1316331                    /lib/x86_64-linux-gnu/libnss_nis-2.23.so
7f6d3c433000-7f6d3c449000 r-xp 00000000 08:01 1316302                    /lib/x86_64-linux-gnu/libnsl-2.23.so
7f6d3c449000-7f6d3c648000 ---p 00016000 08:01 1316302                    /lib/x86_64-linux-gnu/libnsl-2.23.so
7f6d3c648000-7f6d3c649000 r--p 00015000 08:01 1316302                    /lib/x86_64-linux-gnu/libnsl-2.23.so
7f6d3c649000-7f6d3c64a000 rw-p 00016000 08:01 1316302                    /lib/x86_64-linux-gnu/libnsl-2.23.so
7f6d3c64a000-7f6d3c64c000 rw-p 00000000 00:00 0
7f6d3c64c000-7f6d3c654000 r-xp 00000000 08:01 1316322                    /lib/x86_64-linux-gnu/libnss_compat-2.23.so
7f6d3c654000-7f6d3c853000 ---p 00008000 08:01 1316322                    /lib/x86_64-linux-gnu/libnss_compat-2.23.so
7f6d3c853000-7f6d3c854000 r--p 00007000 08:01 1316322                    /lib/x86_64-linux-gnu/libnss_compat-2.23.so
7f6d3c854000-7f6d3c855000 rw-p 00008000 08:01 1316322                    /lib/x86_64-linux-gnu/libnss_compat-2.23.so
7f6d3c855000-7f6d3ca15000 r-xp 00000000 08:01 1316314                    /lib/x86_64-linux-gnu/libc-2.23.so
7f6d3ca15000-7f6d3cc15000 ---p 001c0000 08:01 1316314                    /lib/x86_64-linux-gnu/libc-2.23.so
7f6d3cc15000-7f6d3cc19000 r--p 001c0000 08:01 1316314                    /lib/x86_64-linux-gnu/libc-2.23.so
7f6d3cc19000-7f6d3cc1b000 rw-p 001c4000 08:01 1316314                    /lib/x86_64-linux-gnu/libc-2.23.so
7f6d3cc1b000-7f6d3cc1f000 rw-p 00000000 00:00 0
7f6d3cc1f000-7f6d3cc22000 r-xp 00000000 08:01 1316316                    /lib/x86_64-linux-gnu/libdl-2.23.so
7f6d3cc22000-7f6d3ce21000 ---p 00003000 08:01 1316316                    /lib/x86_64-linux-gnu/libdl-2.23.so
7f6d3ce21000-7f6d3ce22000 r--p 00002000 08:01 1316316                    /lib/x86_64-linux-gnu/libdl-2.23.so
7f6d3ce22000-7f6d3ce23000 rw-p 00003000 08:01 1316316                    /lib/x86_64-linux-gnu/libdl-2.23.so
7f6d3ce23000-7f6d3d03e000 r-xp 00000000 08:01 1316179                    /lib/x86_64-linux-gnu/libcrypto.so.1.0.0
7f6d3d03e000-7f6d3d23d000 ---p 0021b000 08:01 1316179                    /lib/x86_64-linux-gnu/libcrypto.so.1.0.0
7f6d3d23d000-7f6d3d259000 r--p 0021a000 08:01 1316179                    /lib/x86_64-linux-gnu/libcrypto.so.1.0.0
7f6d3d259000-7f6d3d265000 rw-p 00236000 08:01 1316179                    /lib/x86_64-linux-gnu/libcrypto.so.1.0.0
7f6d3d265000-7f6d3d268000 rw-p 00000000 00:00 0
7f6d3d268000-7f6d3d26a000 r-xp 00000000 08:01 1316321                    /lib/x86_64-linux-gnu/libutil-2.23.so
7f6d3d26a000-7f6d3d469000 ---p 00002000 08:01 1316321                    /lib/x86_64-linux-gnu/libutil-2.23.so
7f6d3d469000-7f6d3d46a000 r--p 00001000 08:01 1316321                    /lib/x86_64-linux-gnu/libutil-2.23.so
7f6d3d46a000-7f6d3d46b000 rw-p 00002000 08:01 1316321                    /lib/x86_64-linux-gnu/libutil-2.23.so
7f6d3d46b000-7f6d3d474000 r-xp 00000000 08:01 1316333                    /lib/x86_64-linux-gnu/libcrypt-2.23.so
7f6d3d474000-7f6d3d673000 ---p 00009000 08:01 1316333                    /lib/x86_64-linux-gnu/libcrypt-2.23.so
7f6d3d673000-7f6d3d674000 r--p 00008000 08:01 1316333                    /lib/x86_64-linux-gnu/libcrypt-2.23.so
7f6d3d674000-7f6d3d675000 rw-p 00009000 08:01 1316333                    /lib/x86_64-linux-gnu/libcrypt-2.23.so
7f6d3d675000-7f6d3d6a3000 rw-p 00000000 00:00 0
7f6d3d6a3000-7f6d3d6c9000 r-xp 00000000 08:01 1316312                    /lib/x86_64-linux-gnu/ld-2.23.so
7f6d3d8ad000-7f6d3d8b2000 rw-p 00000000 00:00 0
```

- **Result**


0056| 0x7ffcecd611a0 --> 0x0
Legend: code, data, rodata, value
Stopped reason: SIGABRT
0x00007f9f3953e418 in __GI_raise (sig=sig@entry=0x6) at ../sysdeps/unix/sysv/linux/raise.c:54
54 ../sysdeps/unix/sysv/linux/raise.c: No such file or directory.
gdb-peda$ bt
#0 0x00007f9f3953e418 in __GI_raise (sig=sig@entry=0x6) at ../sysdeps/unix/sysv/linux/raise.c:54
#1 0x00007f9f3954001a in __GI_abort () at abort.c:89
#2 0x00007f9f3958072a in __libc_message (do_abort=do_abort@entry=0x2,
fmt=fmt@entry=0x7f9f39697c7f "*** %s ***: %s terminated\n") at ../sysdeps/posix/libc_fatal.c:175
#3 0x00007f9f3962189c in __GI___fortify_fail (msg=<optimized out>,
msg@entry=0x7f9f39697c10 "buffer overflow detected") at fortify_fail.c:37
#4 0x00007f9f3961f8a0 in __GI___chk_fail () at chk_fail.c:28
#5 0x000000000042b56b in memcpy (__len=0x140, __src=0x65635c <inpacket_buf+28>, __dest=0x7ffcecd61550)
at /usr/include/x86_64-linux-gnu/bits/string3.h:53
#6 eap_request (esp=0x69a800 <eap_states>,
inp=0x65634a <inpacket_buf+10> "\357\n\\\227.†\263\060s\020\371\260\336\317", 'A' <repeats 64 times>, 'a' <
repeats 118 times>..., id=0x83, len=0x152) at eap.c:1428
#7 0x000000000040999f in get_input () at main.c:1095
#8 main (argc=argc@entry=0xe, argv=argv@entry=0x7ffcecd61c68) at main.c:532
#9 0x00007f9f39529830 in __libc_start_main (main=0x408f90 <main>, argc=0xe, argv=0x7ffcecd61c68,
init=<optimized out>, fini=<optimized out>, rtld_fini=<optimized out>, stack_end=0x7ffcecd61c58)
at ../csu/libc-start.c:291
#10 0x0000000000409f49 in _start () at main.c:454
gdb-peda$

Conclusion

GitHub Security Lab to the Rescue

While Vijay Sarvepalli researched GitHub's latest protection initiative, Vijay Sarvepalli wanted to identify opportunities to leverage GitHub's API and CodeQL solutions to address the issue, and he used code to propose the fix to software repository users. He has contacted our Government Protection Leader, Allan Friedman (Director of Cyber Security Initiatives at the National Telecommunications and Information Administration (NTIA) in the United States Department of Commerce), which has put together a range of organisations, including GitHub, to create a Software Bill of Materials (SBOM). Allan introduced Vijay Sarvepalli to the people at GitHub who are dedicated to confidentiality, and they put me in contact with the head of the GitHub Security Lab, Nico Waisman.

Nico and his international team at GitHub Security Lab quickly found a way to adapt their security patching mechanism to this problem. They triggered an automatic "robot" technology that reached out to the owners of all repositories impacted by this bug. Repository owners just had to take a few quick measures to fix and secure their copied or forked version of the pppd program, fixing the bug. This community initiative took us to the stage where tech was being fixed. This offered a modular, timely approach to implement improvements to the source code in order to enhance protection. Within four days of GitHub Security Lab's automatic updates, 1,896 repository owners provided specifics of the bug and were granted the option to fix it in a few clicks. At least 42 of these repository owners have approved an automatic patch; 13 more have confirmed that the problem has already been fixed. Without automation, it will take several days to contact the affected repository owners to fix their apps.

DoD's Challenge and CERT's Role in the Future of Software

As a federally funded research and development center (FFRDC), Carnegie Mellon University's Software Engineering Institute (SEI) and its CERT Division are constantly confronted by the challenges the U.S. Department of Defense (DoD) faces in cyberspace. DoD CIO Terry Halvorsen, a renowned cybersecurity evangelist, said that "cyber defensive actions and counteractions will occur in milliseconds" in his address at the AFCEA conference. These desired cyber defensive actions cannot be done manually or through cumbersome communication processes. They must be delivered via software and automated as much as possible to limit issues with current human-in-the-loop patching models.

Throughout potential threat management activities, we expect to identify situations where we can take advantage of incentives (such as this partnership with GitHub Security Lab) to accelerate source code patching against information security vulnerabilities. Although we understand that this will not fix any issue of information protection and is not a substitution for good coding methods, we do realize that bugs can tend to be found in information after it has been published. When information is pervasive in our everyday lives, information can only be protected by prompt identification of vulnerabilities — and, if possible, by automating both identification and answer.

References

- https://www.kb.cert.org/vuls/id/782301/
- https://thehackernews.com/2020/03/ppp-daemon-vulnerability.html
- https://www.tenable.com/blog/cve-2020-8597-buffer-overflow-vulnerability-in-point-to-point-protocol-daemon-pppd
- https://insights.sei.cmu.edu/cert/2020/03/security-automation-should-begin-at-the-source.html
- https://packetstormsecurity.com/files/156802/pppd-2.4.8-Buffer-Overflow.html
- https://www.drizgroup.com/driz_group_blog/what-is-remote-code-execution-attack-how-to-prevent-this-type-of-cyberattack
- https://github.com/WinMin/CVE-2020-8597
- http://www.howtodoityourself.org/pppoe-server-how-to-do-it-yourself.html