# Expanding Chainsaw

Tool: <https://github.com/countercept/chainsaw>

Documentation: <https://github.com/countercept/chainsaw/tree/documentation_improvements>

Description:

Chainsaw provides a powerful ‘first-response’ capability to quickly identify threats within Windows event logs. It offers a generic and fast method of searching through event logs for keywords, and by identifying threats using built-in detection logic and via support for Sigma detection rules.

Following a recent request, the developer of Chainsaw provided some documentation to help users expand on the tool. This included how the Chainsaw mappings worked and how to link them to sigma rules and also how to create your own mappings and associate them with rules of interest (either preexisting or bespoke self creations)

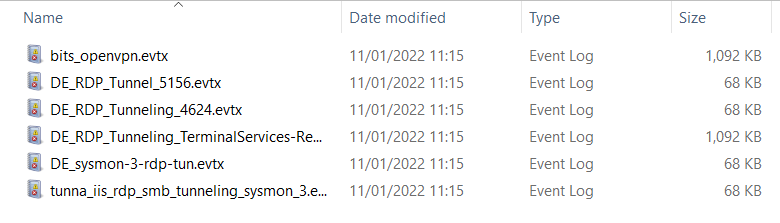
To help support this new guidance documentation and help understand the mappings better myself, I have run some tests with some new mappings and sigma rules I have created and thought I’d write up and step by step as to how I achieved this.

Firstly, I created a very simple rule that simply looks at my own workstations Security.evtx for a 4624 login from my username and local loopback address as I knew it existed and wanted to see if it appeared in the chainsaw standard out in the terminal amongst it’s usual built in logic returns.

This was relatively straightforward and I was pleased to see, was successful. I also tested using the –csv switch and successfully outputted the results .csv including my test rule.

Following my success with this somewhat easy and useless rule, I decided I would use one of Chainsaw’s provided EVTX attack examples to create the necessary mappings and rules to detect all of the known IOCs from these provided .evtx files. The attack example I chose was:

<https://github.com/sbousseaden/EVTX-ATTACK-SAMPLES/tree/b947ed80cd67a7413f6689d032eb1151f85f9df6/Command%20and%20Control>

The original evtx files were all contained within a folder with a pcap, log files and a README.md. To save the frustration of potential conflicts or errors when pointing chainsaw at this folder, I moved the evtx files into their own directory leaving me with:

I then focused on the 4 DE\_RDP evtx files for the purposes of my testing.

The README.md provided the following information:

Connection Proxy:

1. RDP Tunneling via SSH - eventid 4624 - Logon Type 10 and Source IP eq to loopback IP address

2. RDP Tunneling via SSH - eventid 1149 - TerminalServices-RemoteConnectionManagerOperational - RDP source IP loopback IP address

3. RDP Tunneling via SSH - Sysmon eventid 3 - local port forwarding to/from loopback IP (svchost.exe <-> plink.exe)

4. RDP Tunneling via SSH - eventid 5156 - local port forwarding to/from loopback IP to 3389 rdp port

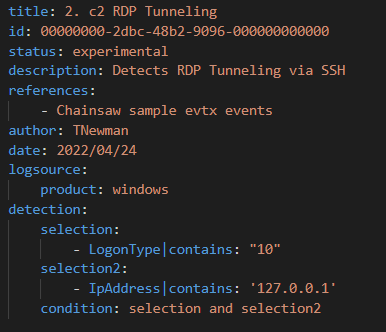
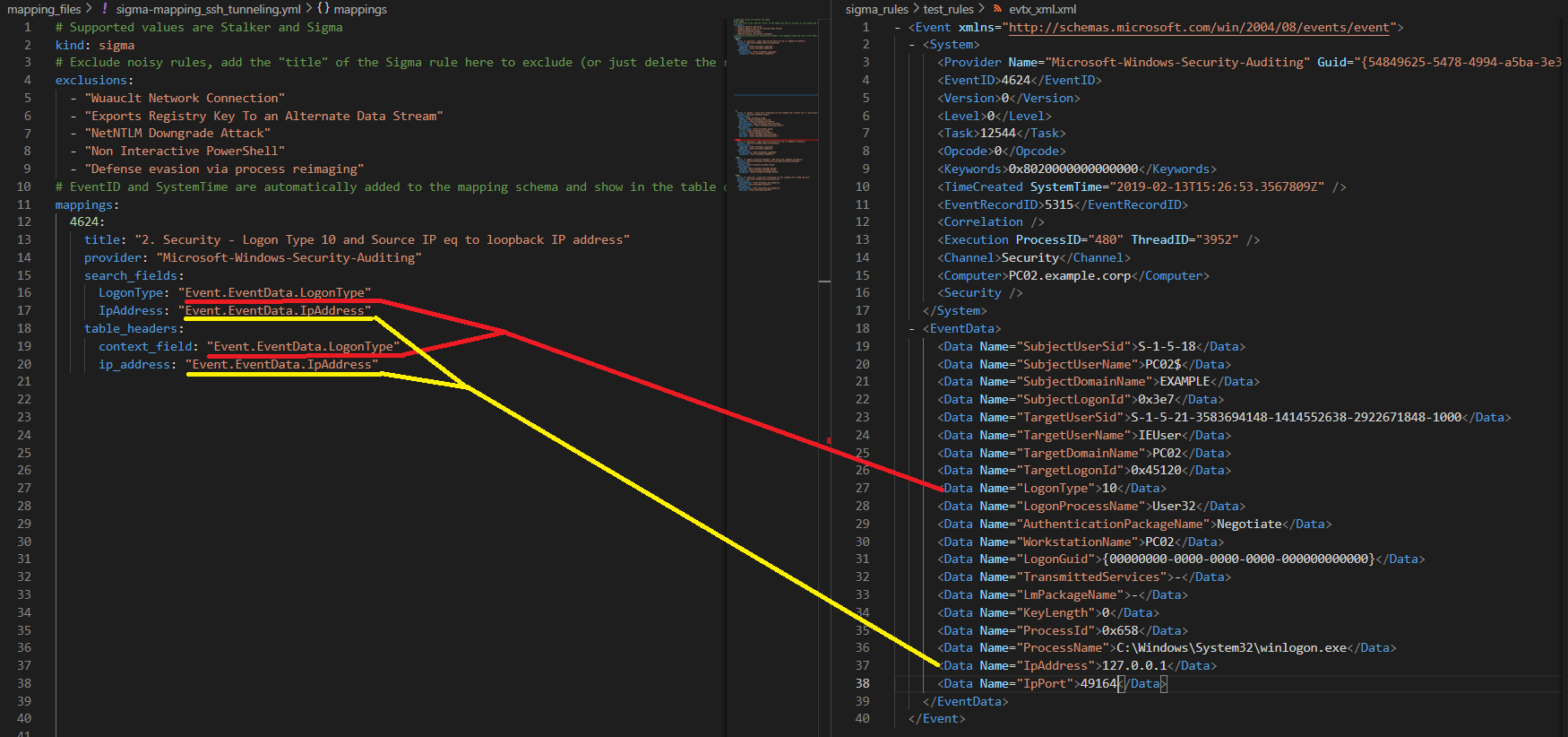
*NOTE: I’m using these .evtx samples simply as a reference point to create the mappings and rules. This is not meant to represent the best or definitive way for writing these mappings/rules for this specific IOC. If you would like to learn more about RDP tunneling, here is a great article:* [*https://www.real-sec.com/2019/04/bypassing-network-restrictions-through-rdp-tunneling/*](https://www.real-sec.com/2019/04/bypassing-network-restrictions-through-rdp-tunneling/)

## STEP 1 - EventID 4634

RDP Tunneling via SSH - eventid 4624 - Logon Type 10 and Source IP eq to loopback IP address

I needed to create a rule that would look for EventID 4624 for Logon Type 10 with the Source IP as the loopback IP

**LEFT: mappings file** **RIGHT: XML of EventID 4624 (Security.evtx)**



**LEFT: Sigma Rule** created to look for our specific IOCs:

detection>selection>LogonType

detection>selection2>IpAddress

^^This is where we apply our actual IOCs (Logon Type 10 &

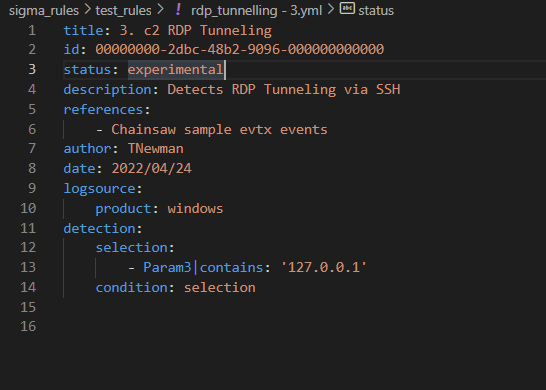
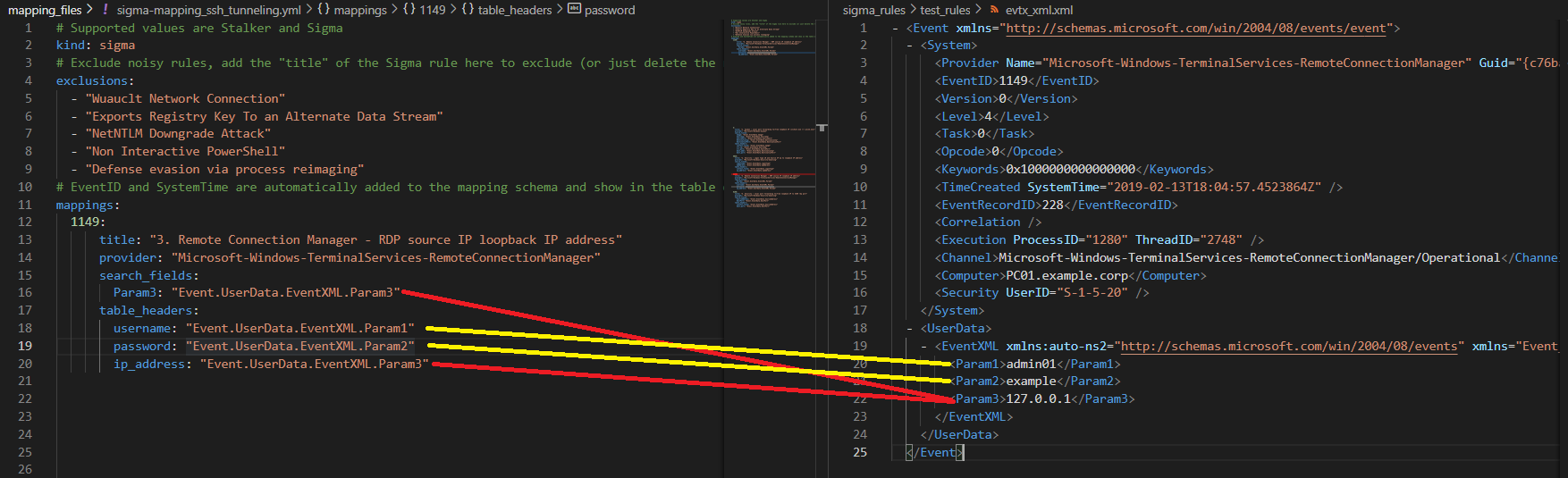
loopback IP)

Condition == rule must match both detections together

## STEP 2 - EventID 1149

2. RDP Tunneling via SSH - eventid 1149 - TerminalServices-RemoteConnectionManagerOperational - RDP source IP loopback IP address

Next is EventID 1149 from the Remote Connection Manager evtx looking for the local loopback address



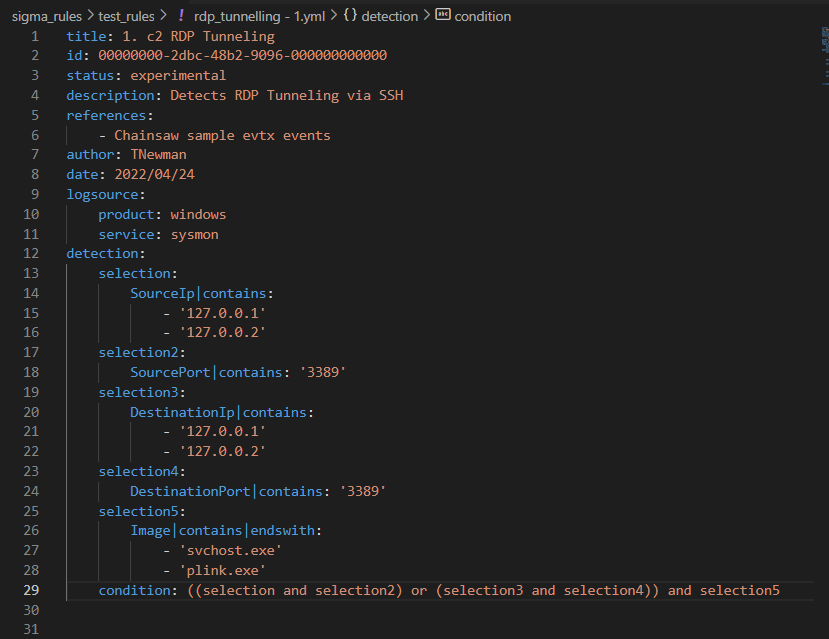
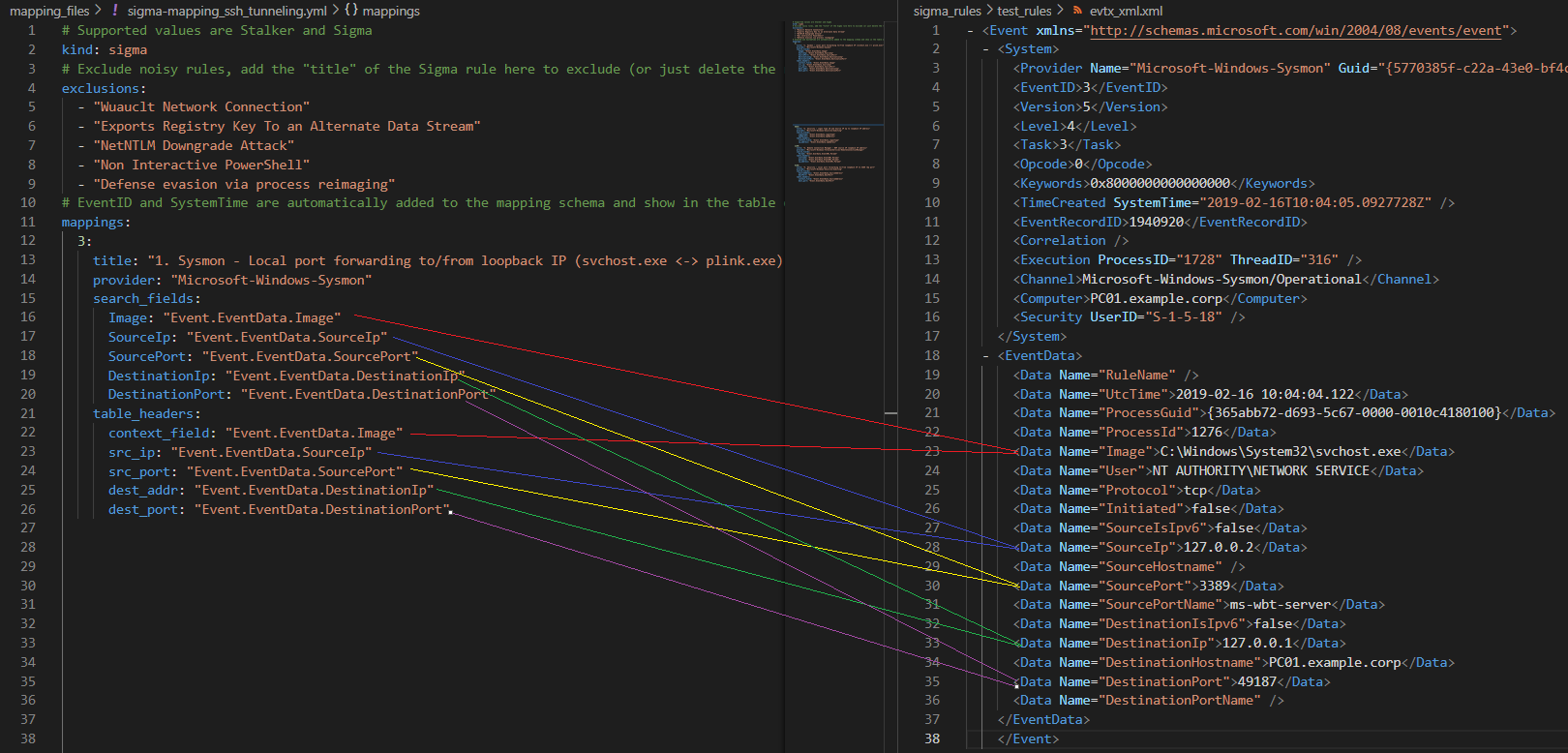
You’ll notice with this mapping (top left), I have used additional parameters under the table\_headers. These are the values we want chainsaw to output to the terminal and in this case I asked for the **username/password** as well as our searchable parameter, the **loopback IP**

The sigma rule only requires our searchable fields from the mapping file.

## STEP 3 - EventID 3

3. RDP Tunneling via SSH - Sysmon eventid 3 - local port forwarding to/from loopback IP (svchost.exe <-> plink.exe)

Sysmon EventID 3 is next. This is looking for port forwarding to and from our loopback IP between svchost and plink.exe

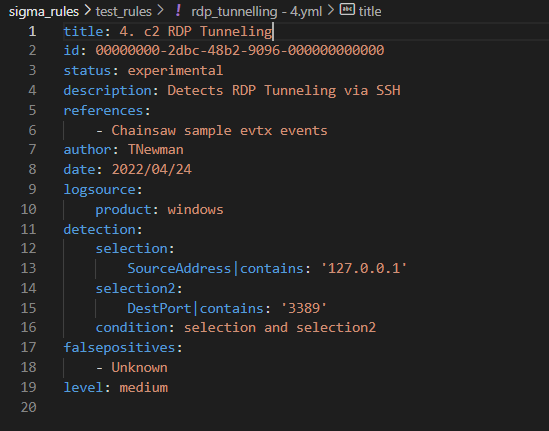
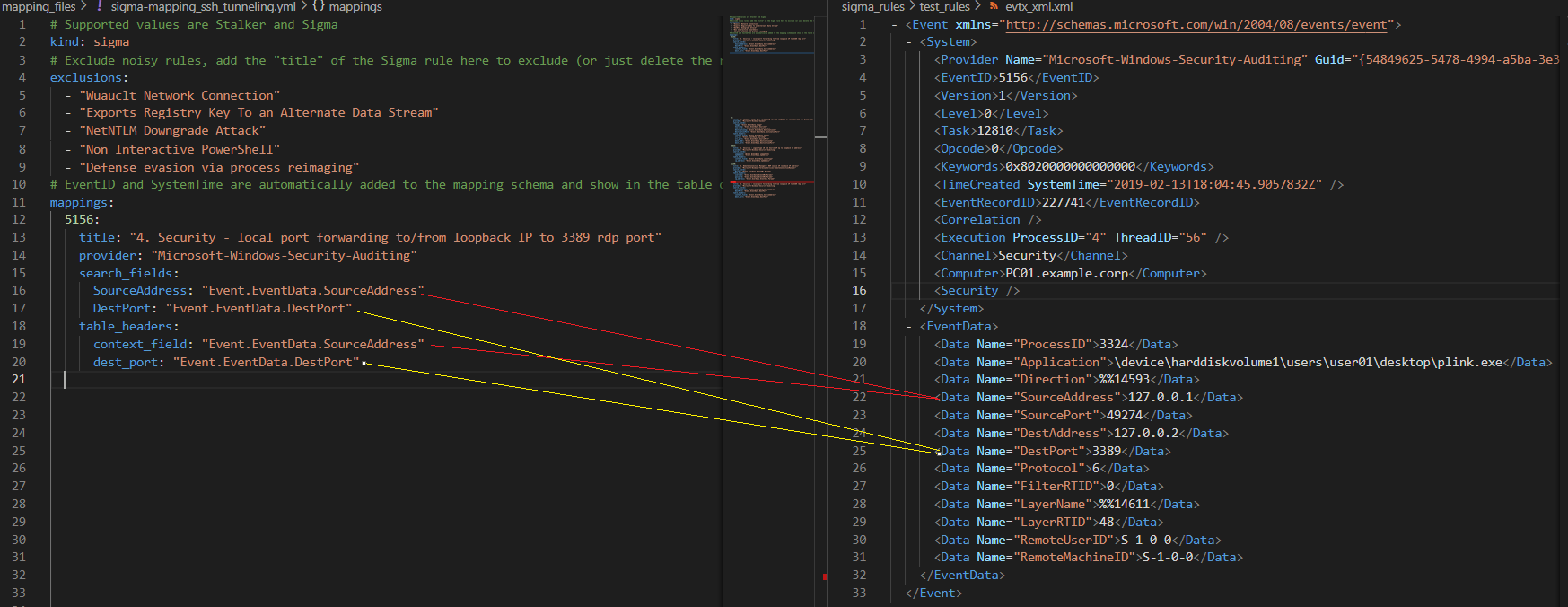


Having a quick look at the sysmon.evtx file showed that both loopback address 172.0.0.1 & \*.2 was being used to demonstrate the port forwarding between the executables. This is reflected in the sigma rule

## STEP 4 - EventID 5156

4. RDP Tunneling via SSH - eventid 5156 - local port forwarding to/from loopback IP to 3389 rdp port

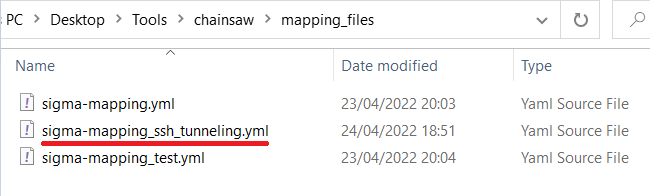
Finally, EventID 5156 from Security.evtx is similar to Sysmon EventID 3 but with less information so as to more easily differentiate between the two I’m only looking for plink.exe outbound traffic to port 3389



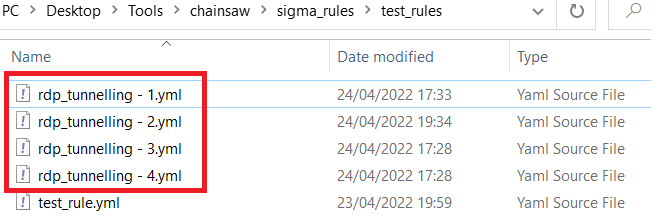
## STEP 5 - Layout

The mappings and rules were placed in the following paths:

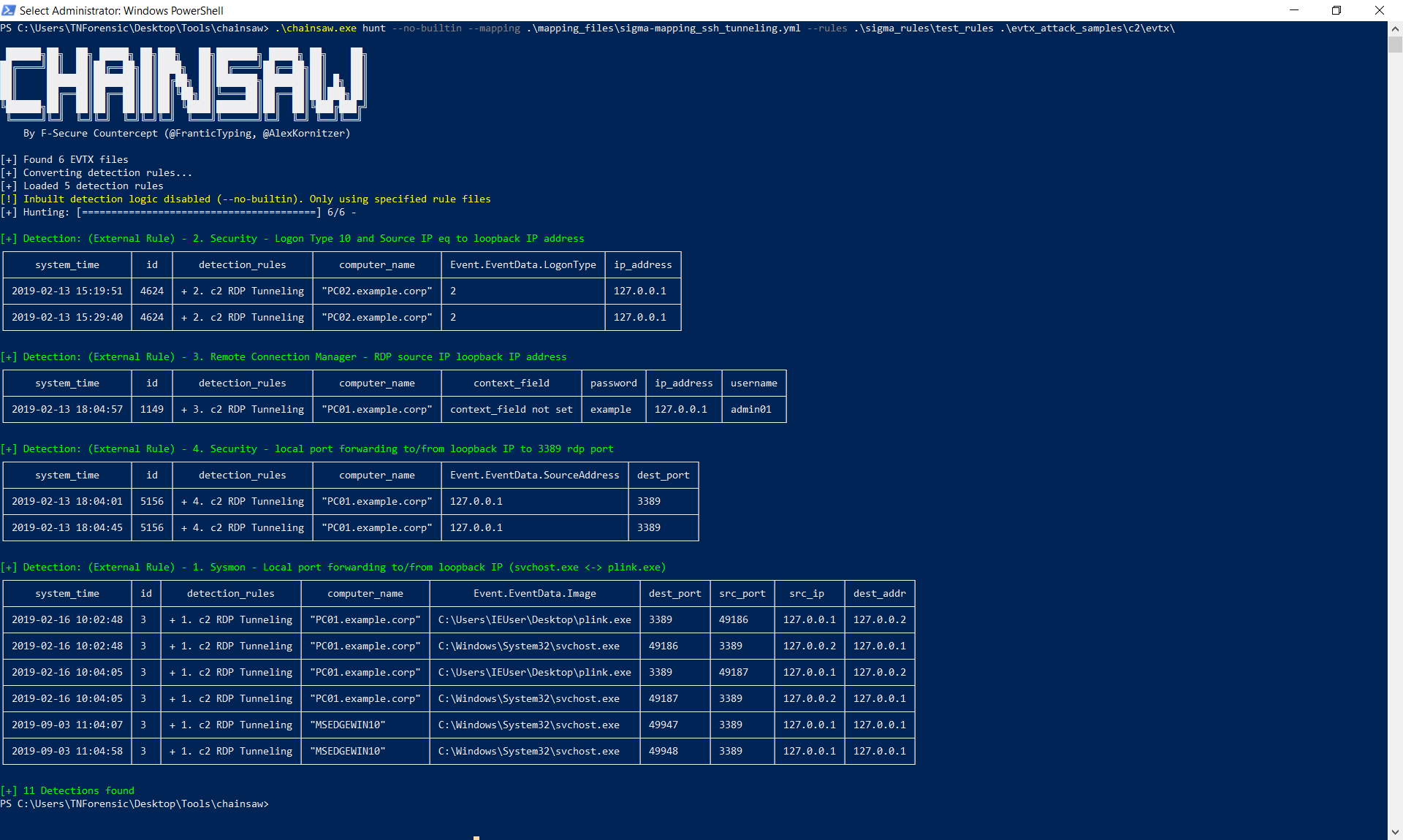
<path\_to>\chainsaw\mapping\_files



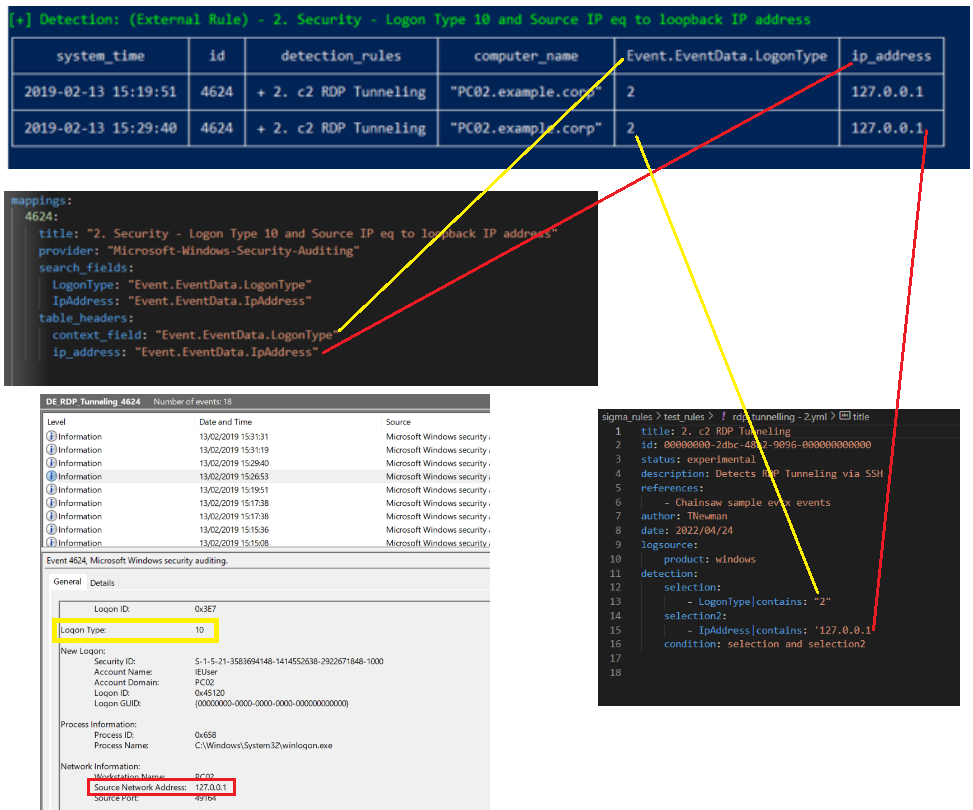
<path\_to>\chainsaw\sigma-rules\test\_rules



# Test Drive



Perfect!

Looking at our Security.evtx Logon Type 10 results a bit closer…

Note - The system\_time, id, detection\_rules and computer\_name are all parsed and displayed automatically by chainsaw

I also tested these rules using the –csv switch and they were all parsed across with no errors.

# Conclusion

What is great about the functionality of chainsaw’s mapping tool is that you can either:

1. Add to the native mappings - this could be used for event log types that you want extracted every time you run chainsaw (adding to their Suspicious Logons, Suspicious Process creation etc…)
2. Create an entirely separate mapping file bespoke to your investigation (as I have done with the RDP Tunneling) which could easily be shared and added to as an investigation progresses.

Being able to create individual mapping files with associated sigma rules will be a fantastic way for forensic examiners or threat intelligence analysts to build out packages for specific malware or threat actor TTPs/IOCs that are discovered in evtx logs and share with the community/industry as a whole.