# MALWARE DEVELOPMENT

## Hacker vs Developer

Writing code is an absolute art and most developers aren’t good at it. Hackers on the other side are not developers. In my opinion, hacker is someone who takes multiple tools and somehow put them together. I am not saying thats a bad thing. I am just saying that hackers don’t have the skills to write everything from the scratch. At the same time they don’t have the discipline. In security industry there are very few hackers that are good developers as well. Those hackers / good coders write malware like Stuxnet, flame, duqu etc.

## Is coding mandatory?

One doesn't have to be a developer to be in the cyber security industry, and thats a fact. Look around you, most of the cyber security guys that you work with, aren’t really developers. Even though they know how to write some python / ruby etc and script things together. So if you aren’t a developer and you are in the field of cyber security, you are still legit :)

## Should you learn coding?

Learning anything is a plus. If you are in the cyber security industry and you know how to code, that will give you an edge. So learning some coding is not a bad idea.

## Programming language for malware development

Plain old **C** all they way. Less compilation overhead. Plug-n-play executables. Smaller payload etc. I guess I am old! If you want to go with Python and convert it into a large binary, go for it. .Net isn't bad as well. Chose whatever you feel like and whatever is easy for you.

Here is an example of a malware, written in python

**[http://udurrani.com/0fff/py.html](http://udurrani.com/0fff/py.html)**

## Malware flow

Most malicious payloads follow a certain flow. Hacker who puts malware together, relies on functions like CreateProcess(), ShellExecute(), system(), exec() etc. The challenge with this approach is that the malware leaves a lot of evidence behind. Any EDR solutions, these days can detect a child or sub process very easily. E.g. if a payload calls netsh command, netsh becomes a child process of that payload. An EDR solution will pick on this, report it and also push an IOC or IOA against such flow.

## So how does a malicious flow look like?

Well, thats a tough one. Some flows are easy to challenge but some look very legit. Let's say that CMD.exe is spawning powershell.exe or cscript.exe, or vice versa. Its not easy to challenge this flow as it looks legit. But the real question is: What if the attacker doesn't use any child process????? and develops a function or uses an API to carry on all the tasks. How would EDR react to that? In most cases, EDR won't. EDR can't hook on every call. Let me give you an example.

Here is a binary that runs arp -a command by using arp.exe command (**Password**: foo)

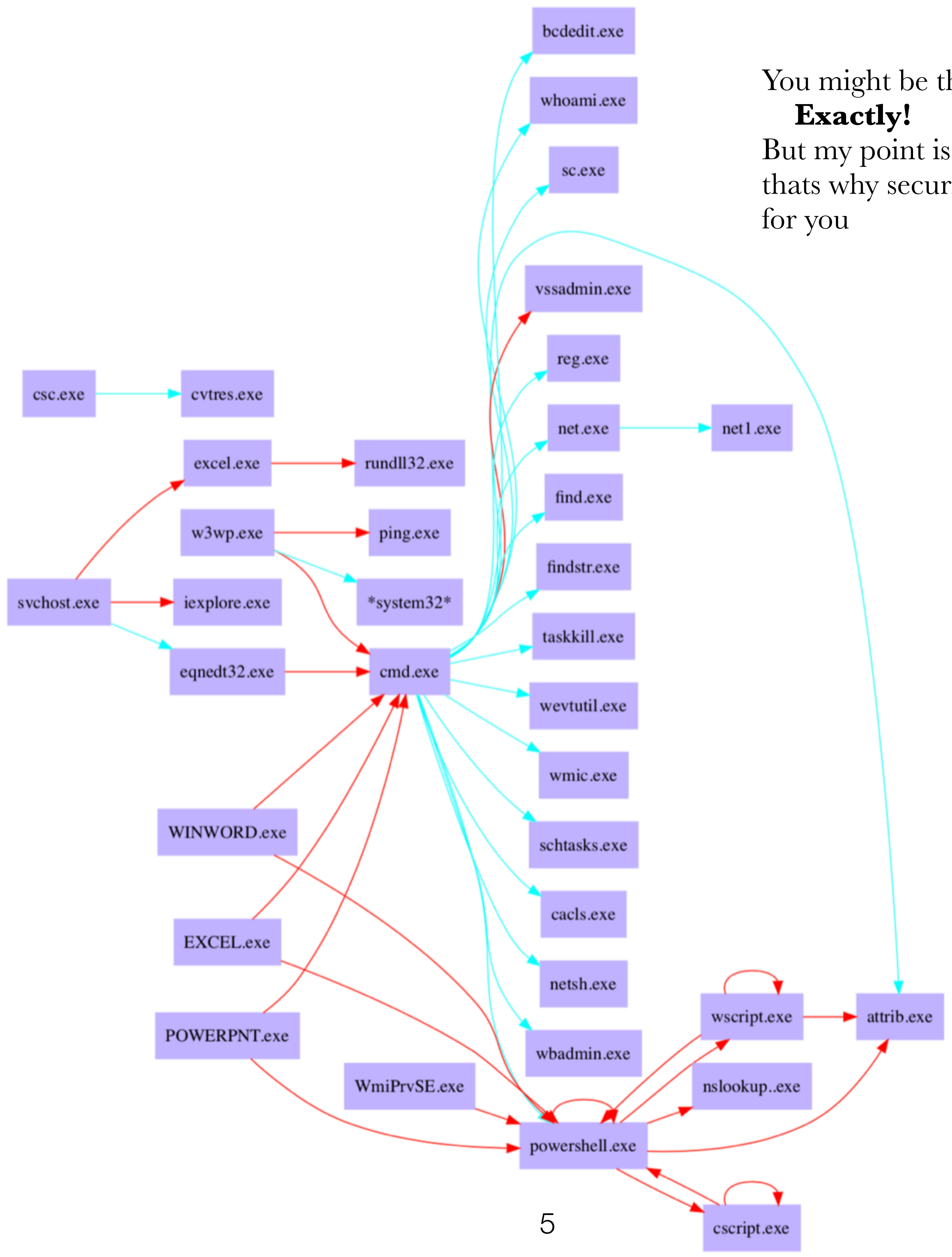
http://udurrani.com/0fff/a1.zip

Here is a binary that runs the same functionality without using arp.exe command (**Password**: foo)

http://udurrani.com/0fff/a2.zip

Test it your self and find out how an EDR would react to it.

## Let's look at some malicious flows



You might be thinking that some of this looks legit?
**Exactly!**
But my point is: Its easy to detect on such behavior and thats why security solutions can get this triage spelled out for you

## Then why do malware use such commands?

Many reasons:

- Ease of use
- Most hackers are great system guys and understand system much better then coding
- Avoiding system commands could delay their hacking project / scheme
- Developers don't really have to write clean code or care about vulnerabilities. They are in a win-win situation. If it lands, ***GREAT***! if it doesn't, lets put a new payload together.

## Yet another example!

Let me give you one more example, where no system commands are being used:

- Payload is initiated as normal user
- Payload follows multitasking
- It locks everything down
- And scans few IP addresses at the same time
- Once the scan is complete, it informs the payload and it panics
- If payload runs as admin on windows 7, blue screen of death
- Windows 10 will panic any way and go to the famous blue screen
- So all that is done at the same time with proper IPC and without using any system commands.

In this situation an EDR may pick on the socket() functionality but other stuff, not so much. This payload won't hurt your machine but run it on a VM anyway. Before starting, run a packet capture as well.

**http://themalware.com/no_command.zip**

## Malware must by-pass

Malware's job is to by-pass existing security barriers and it's not that difficult. No product is 100% secure, thats why they keep on adding new features E.g. with rise of ransomware, some AV's added a feature to prevent file modification based on certain behavior(s). Kaspersky added a backup model, where they would backup modified files and if the behavior is suspicious, the payload is stopped and files are restored. Sophos hada similar feature, I think it was Hitman (later acquired by Sophos).

To by-pass this I developed a ransomware that will use a subProcess to encrypt. What does that mean:

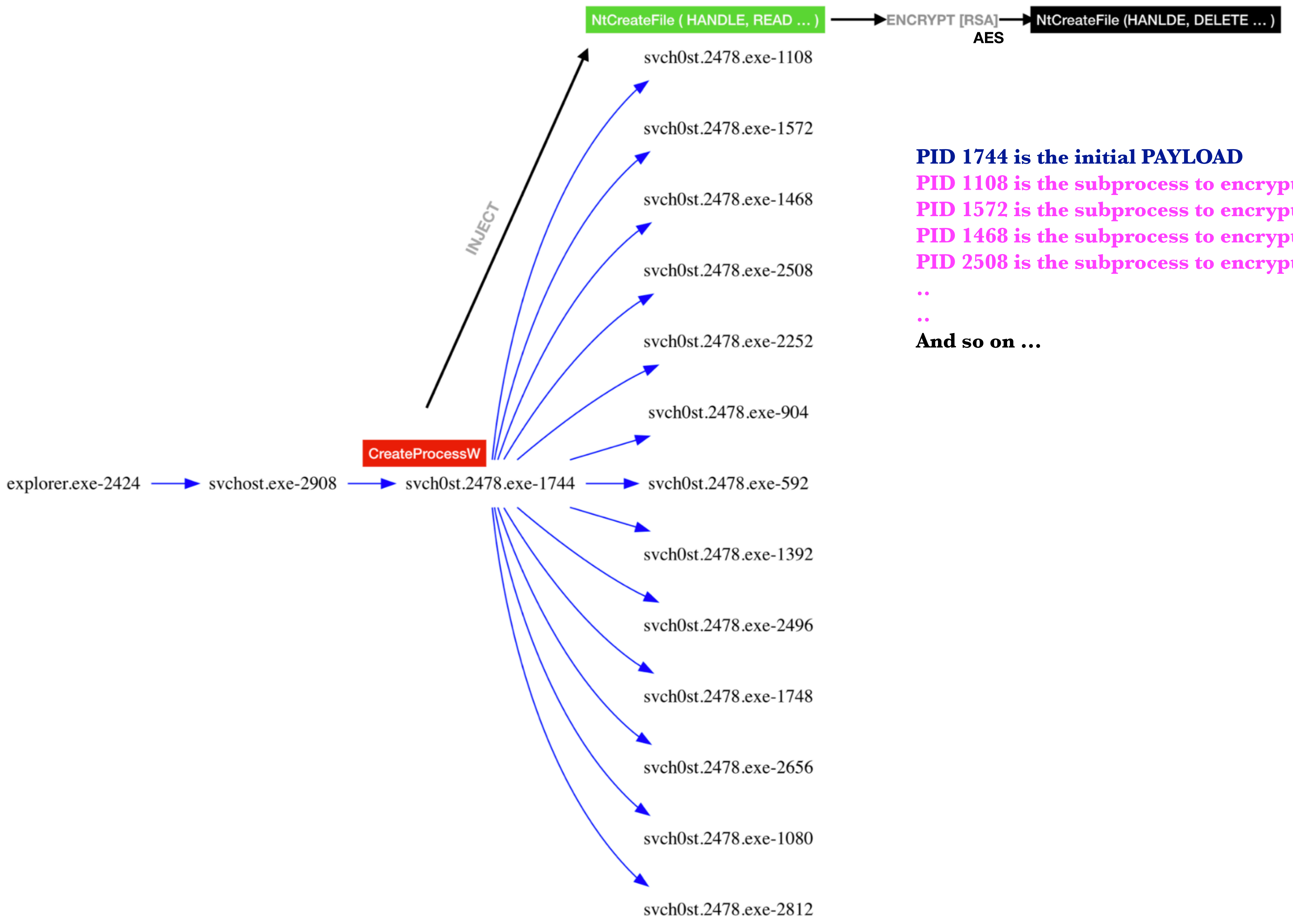
- Payload has 100 worker threads
- Each worker thread spawns the main payload to encrypt a single file
- There is an IPC between the threads so we don't end up encrypting same file twice
- This way I can control how many files are encrypted / subProcess
- And the encryption begins

Here is the test link:

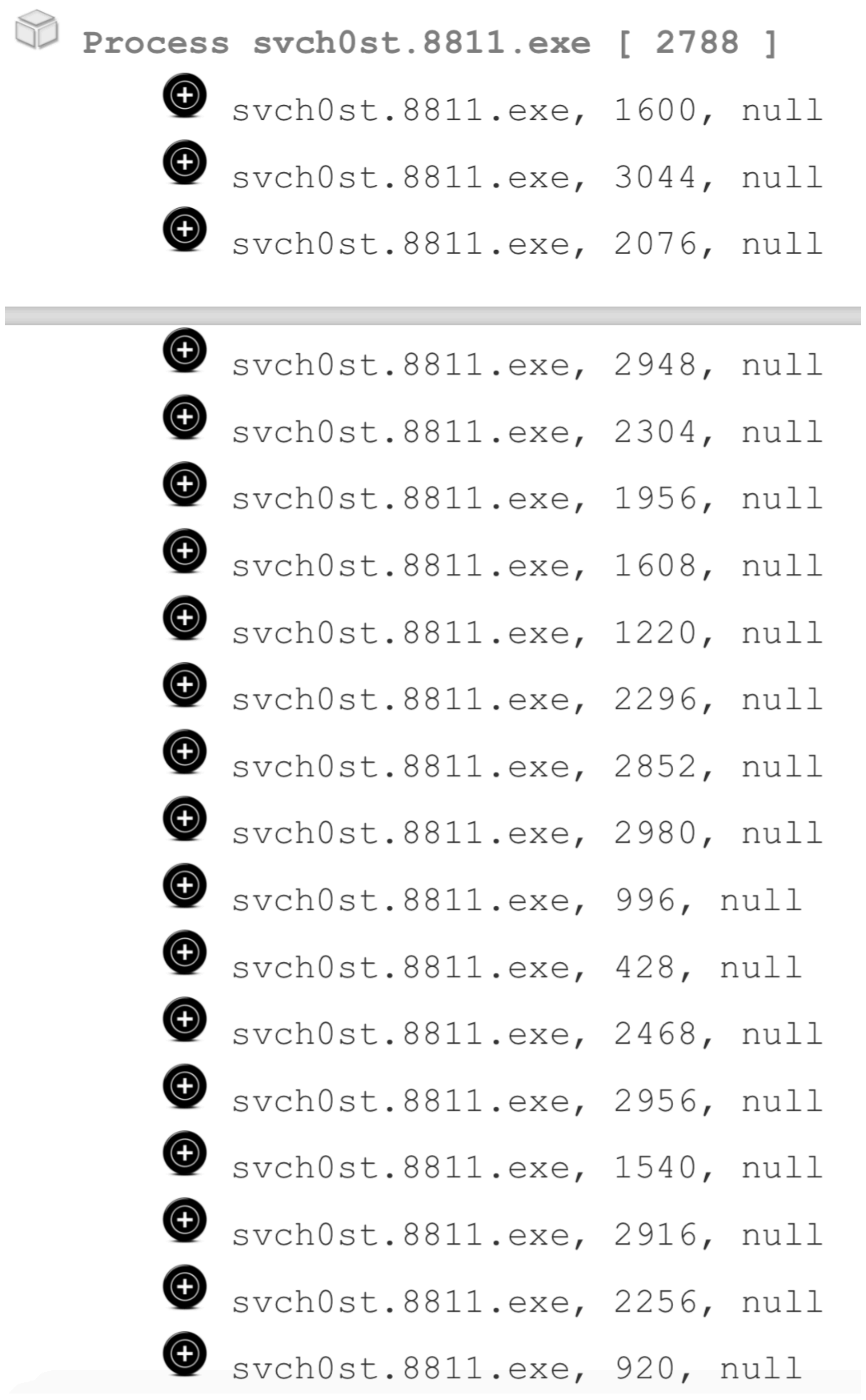
**https://www.youtube.com/watch?v=whPeRFMBhzY**

Recently I noticed a similar technique in a real campaign, where the payload uses a subProcess to encrypt a file. It's using one process / file. This means: If there are 5K files, initial payload will have to spawn 5000 subprocesses to encrypt all files. All sequential though. This means my technique was more efficient :)

## Here is the flow of the malware.



**PID 1744 is the initial PAYLOAD**
**PID 1108 is the subprocess to encrypt a single file**
**PID 1572 is the subprocess to encrypt a single file**
**PID 1468 is the subprocess to encrypt a single file**
**PID 2508 is the subprocess to encrypt a single file**
**..**
**..**
**And so on ...**



```
Process svch0st.8811.exe [ 2788 ]
        svch0st.8811.exe, 1600, null
        svch0st.8811.exe, 3044, null
        svch0st.8811.exe, 2076, null

        svch0st.8811.exe, 2948, null
        svch0st.8811.exe, 2304, null
        svch0st.8811.exe, 1956, null
        svch0st.8811.exe, 1608, null
        svch0st.8811.exe, 1220, null
        svch0st.8811.exe, 2296, null
        svch0st.8811.exe, 2852, null
        svch0st.8811.exe, 2980, null
        svch0st.8811.exe, 996, null
        svch0st.8811.exe, 428, null
        svch0st.8811.exe, 2468, null
        svch0st.8811.exe, 2956, null
        svch0st.8811.exe, 1540, null
        svch0st.8811.exe, 2916, null
        svch0st.8811.exe, 2256, null
        svch0st.8811.exe, 920, null
```

**Another example of the same malware.**
**PID 2788 spawning PID 1600, 3044, 2076, 2948, 1956 …**

## Tools

You always need good tools to develop or analyze malware. Some fun tools you can download from the following link.

**http://udurrani.com/0fff/ff.html**

Another tool for testing reasons

**http://udurrani.com/0fff/wbtool/upurs.zip**

Put the binary on a 64bit machine and execute by running the following command:

**webe.exe <portNumber>** **//** Make sure you have a folder called ud (its a prerequisit)

***E.g. webe.exe 6000***

Now upload an executable from linux, mac or windows machine remotely.

```
curl -F “ud=@fileName" ipAddressofServer:6000/upload
```

This will upload and execute the binary file on remote system (Where webe.exe is running).

You can **ONLY** upload **PE files**.

If you want to launch other file formats, let me know or simply call

1-800-OTHERFILEFORMATS

## Exploitation

Even though malware and exploits are very different, its always a good idea to know about exploits

**http://udurrani.com/exp0/memory_exploitation.pdf**

Thats it for now, this is just an intro. I am going to write more on this topic and cover multiple malicious techniques and topics on malware development