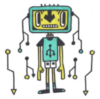
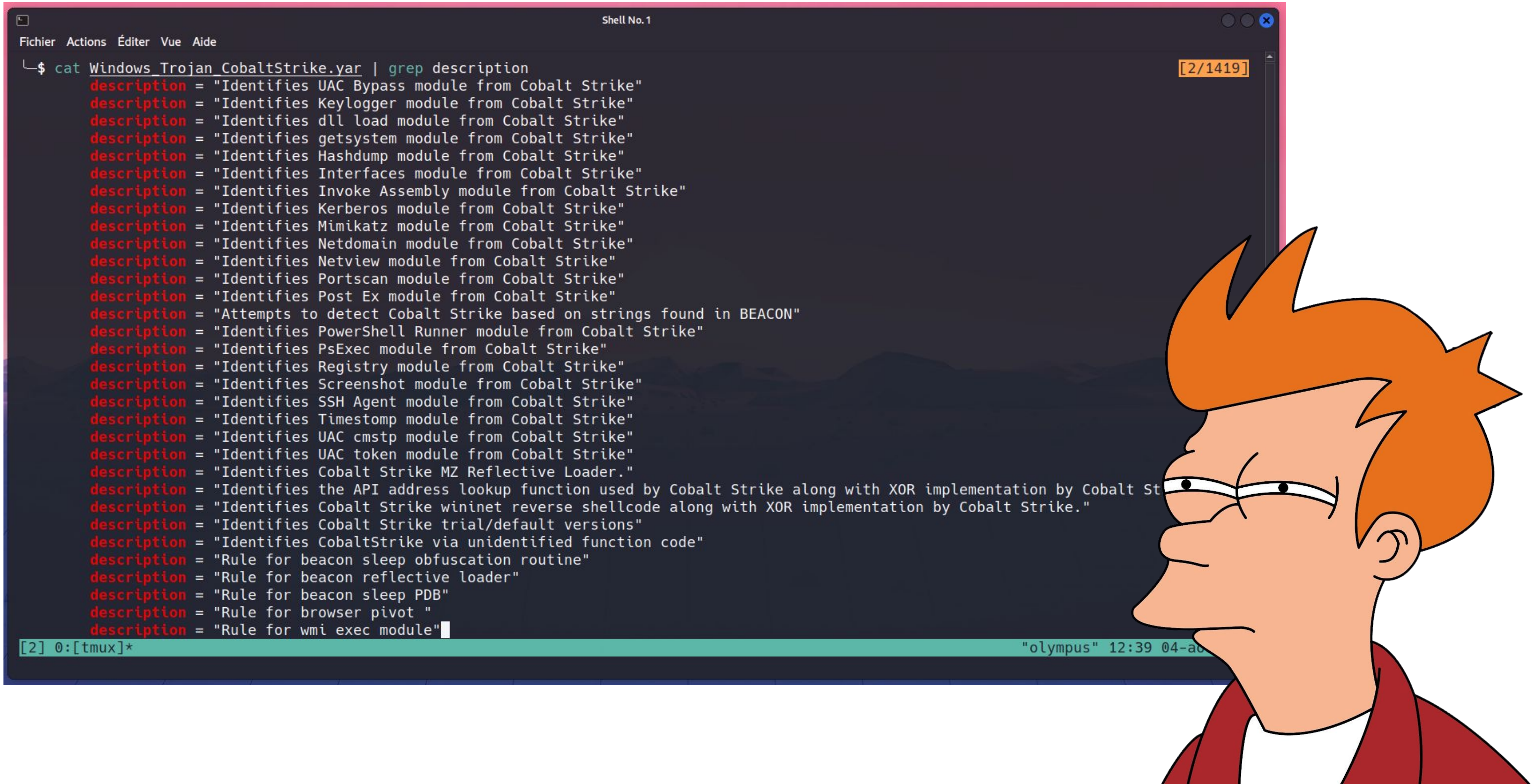
How to develop your own Cobalt Strike

@pentest_swissky
Maxime



@pentest_soka
Romain

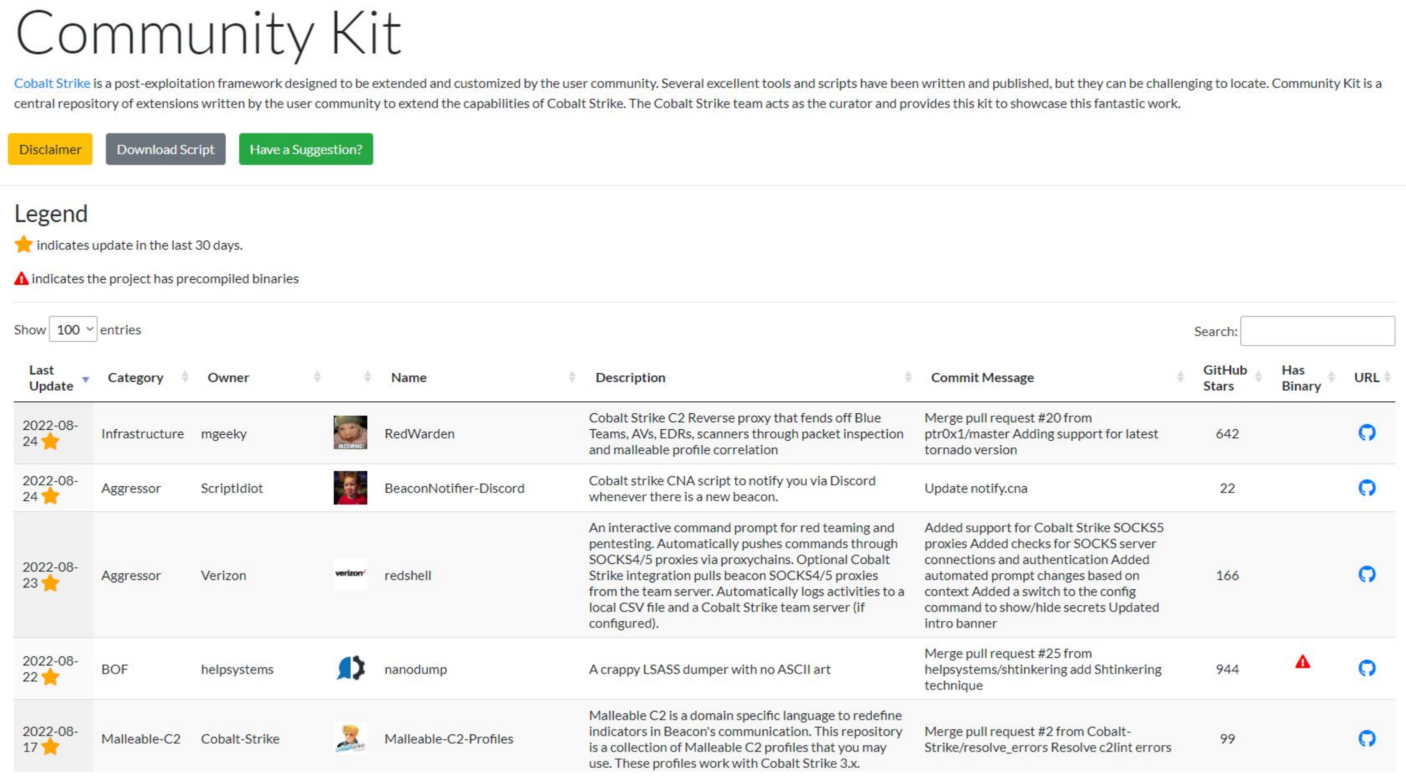
Cobalt Strike: too much IOCs



```
└─$ cat Windows_Trojan_CobaltStrike.yar | grep description                                          [2/1419]
        description = "Identifies UAC Bypass module from Cobalt Strike"
        description = "Identifies Keylogger module from Cobalt Strike"
        description = "Identifies dll load module from Cobalt Strike"
        description = "Identifies getsystem module from Cobalt Strike"
        description = "Identifies Hashdump module from Cobalt Strike"
        description = "Identifies Interfaces module from Cobalt Strike"
        description = "Identifies Invoke Assembly module from Cobalt Strike"
        description = "Identifies Kerberos module from Cobalt Strike"
        description = "Identifies Mimikatz module from Cobalt Strike"
        description = "Identifies Netdomain module from Cobalt Strike"
        description = "Identifies Netview module from Cobalt Strike"
        description = "Identifies Portscan module from Cobalt Strike"
        description = "Identifies Post Ex module from Cobalt Strike"
        description = "Attempts to detect Cobalt Strike based on strings found in BEACON"
        description = "Identifies PowerShell Runner module from Cobalt Strike"
        description = "Identifies PsExec module from Cobalt Strike"
        description = "Identifies Registry module from Cobalt Strike"
        description = "Identifies Screenshot module from Cobalt Strike"
        description = "Identifies SSH Agent module from Cobalt Strike"
        description = "Identifies Timestomp module from Cobalt Strike"
        description = "Identifies UAC cmstp module from Cobalt Strike"
        description = "Identifies UAC token module from Cobalt Strike"
        description = "Identifies Cobalt Strike MZ Reflective Loader."
        description = "Identifies the API address lookup function used by Cobalt Strike along with XOR implementation by Cobalt St
        description = "Identifies Cobalt Strike wininet reverse shellcode along with XOR implementation by Cobalt Strike."
        description = "Identifies Cobalt Strike trial/default versions"
        description = "Identifies CobaltStrike via unidentified function code"
        description = "Rule for beacon sleep obfuscation routine"
        description = "Rule for beacon reflective loader"
        description = "Rule for beacon sleep PDB"
        description = "Rule for browser pivot "
        description = "Rule for wmi exec module"
[2] 0:[tmux]*                                                                              "olympus" 12:39 04-a
```

Cobalt Strike: current bypasses

- Artifact kit
- Stack spoofing
- Sleep mask kit
- UDRL kit
- Custom loaders
  - Powershell
  - .NET
  - Beacon Object File



Community Kit

Cobalt Strike is a post-exploitation framework designed to be extended and customized by the user community. Several excellent tools and scripts have been written and published, but they can be challenging to locate. Community Kit is a central repository of extensions written by the user community to extend the capabilities of Cobalt Strike. The Cobalt Strike team acts as the curator and provides this kit to showcase this fantastic work.

Legend

★ indicates update in the last 30 days.

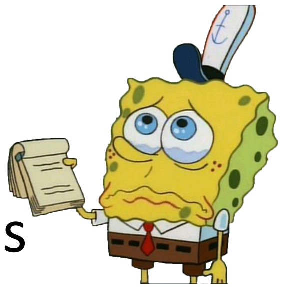
⚠ indicates the project has precompiled binaries

Show 100 entries

Search:

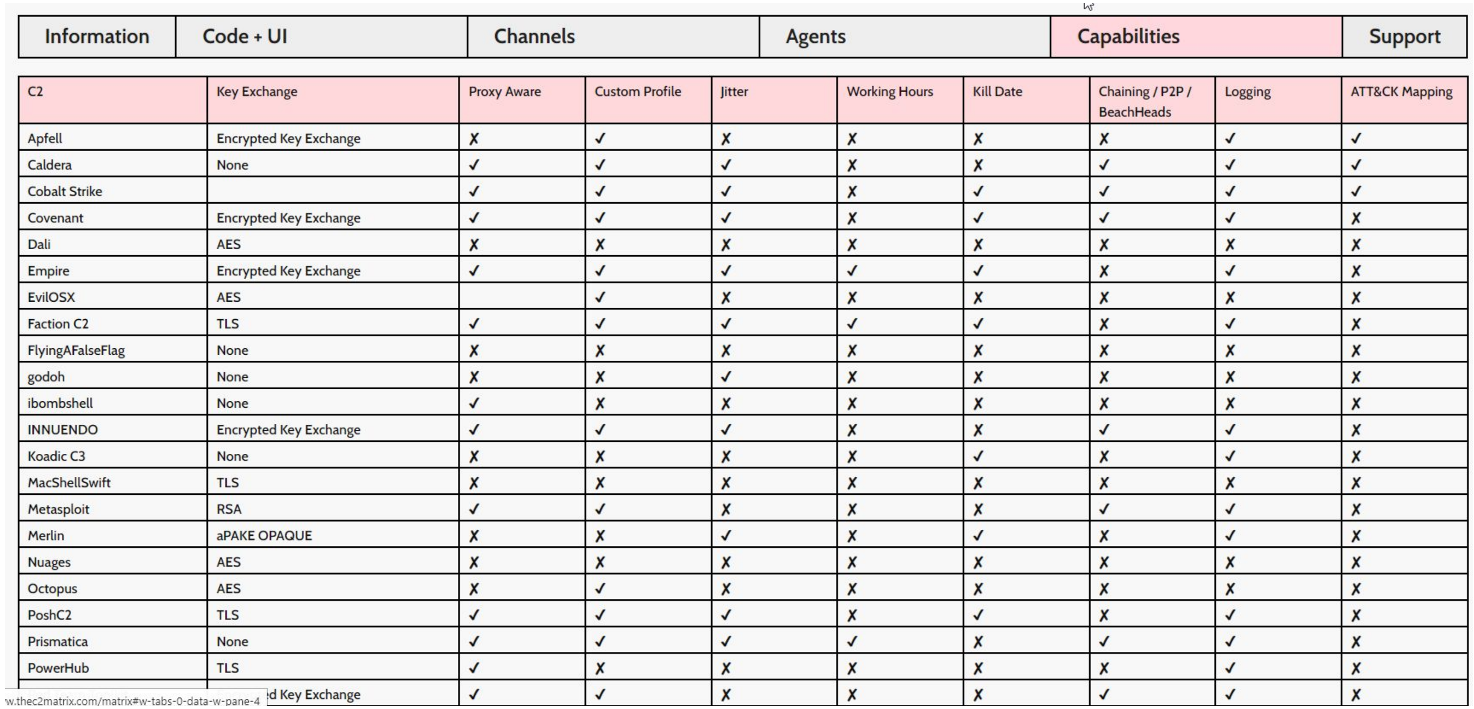
Last Update	Category	Owner		Name	Description	Commit Message	GitHub Stars	Has Binary	URL
2022-08-24 ★	Infrastructure	mgeeky		RedWarden	Cobalt Strike C2 Reverse proxy that fends off Blue Teams, AVs, EDRs, scanners through packet inspection and malleable profile correlation	Merge pull request #20 from ptr0x1/master Adding support for latest tornado version	642		
2022-08-24 ★	Aggressor	ScriptIdiot		BeaconNotifier-Discord	Cobalt strike CNA script to notify you via Discord whenever there is a new beacon.	Update notify.cna	22		
2022-08-23 ★	Aggressor	Verizon		redshell	An interactive command prompt for red teaming and pentesting. Automatically pushes commands through SOCKS4/5 proxies via proxychains. Optional Cobalt Strike integration pulls beacon SOCKS4/5 proxies from the team server. Automatically logs activities to a local CSV file and a Cobalt Strike team server (if configured).	Added support for Cobalt Strike SOCKS5 proxies Added checks for SOCKS server connections and authentication Added automated prompt changes based on context Added a switch to the config command to show/hide secrets Updated intro banner	166		
2022-08-22 ★	BOF	helpsystems		nanodump	A crappy LSASS dumper with no ASCII art	Merge pull request #25 from helpsystems/shtinkering add Shtinkering technique	944	⚠	
2022-08-17 ★	Malleable-C2	Cobalt-Strike		Malleable-C2-Profiles	Malleable C2 is a domain specific language to redefine indicators in Beacon's communication. This repository is a collection of Malleable C2 profiles that you may use. These profiles work with Cobalt Strike 3.x.	Merge pull request #2 from Cobalt-Strike/resolve_errors Resolve c2lint errors	99		

https://cobalt-strike.github.io/community_kit/



… yet still detected sometimes

Others C&C



Information | Code + UI | Channels | Agents | Capabilities | Support

C2	Key Exchange	Proxy Aware	Custom Profile	Jitter	Working Hours	Kill Date	Chaining / P2P / BeachHeads	Logging	ATT&CK Mapping
Apfell	Encrypted Key Exchange	✗	✓	✗	✗	✗	✗	✓	✓
Caldera	None	✓	✓	✓	✗	✗	✓	✓	✓
Cobalt Strike		✓	✓	✓	✗	✓	✓	✓	✓
Covenant	Encrypted Key Exchange	✓	✓	✓	✗	✓	✓	✓	✗
Dali	AES	✗	✗	✗	✗	✗	✗	✗	✗
Empire	Encrypted Key Exchange	✓	✓	✓	✓	✓	✗	✓	✗
EvilOSX	AES		✓	✗	✗	✗	✗	✗	✗
Faction C2	TLS	✓	✓	✓	✓	✓	✗	✓	✗
FlyingAFalseFlag	None	✗	✗	✗	✗	✗	✗	✗	✗
godoh	None	✗	✗	✓	✗	✗	✗	✗	✗
ibombshell	None	✓	✗	✗	✗	✗	✗	✗	✗
INNUENDO	Encrypted Key Exchange	✓	✓	✓	✗	✗	✓	✓	✗
Koadic C3	None	✗	✗	✗	✗	✓	✗	✓	✗
MacShellSwift	TLS	✗	✗	✗	✗	✗	✗	✗	✗
Metasploit	RSA	✓	✓	✗	✗	✗	✓	✓	✗
Merlin	aPAKE OPAQUE	✗	✗	✓	✗	✓	✗	✓	✗
Nuages	AES	✗	✗	✗	✗	✗	✗	✗	✗
Octopus	AES	✗	✓	✗	✗	✗	✗	✗	✗
PoshC2	TLS	✓	✓	✓	✗	✓	✗	✓	✗
Prismatica	None	✓	✓	✓	✓	✗	✓	✓	✗
PowerHub	TLS	✓	✗	✗	✗	✗	✗	✓	✗
w.thec2matrix.com/matrix#w-tabs-0-data-w-pane-4	d Key Exchange	✓	✓	✗	✗	✗	✓	✓	✗

https://www.thec2matrix.com/

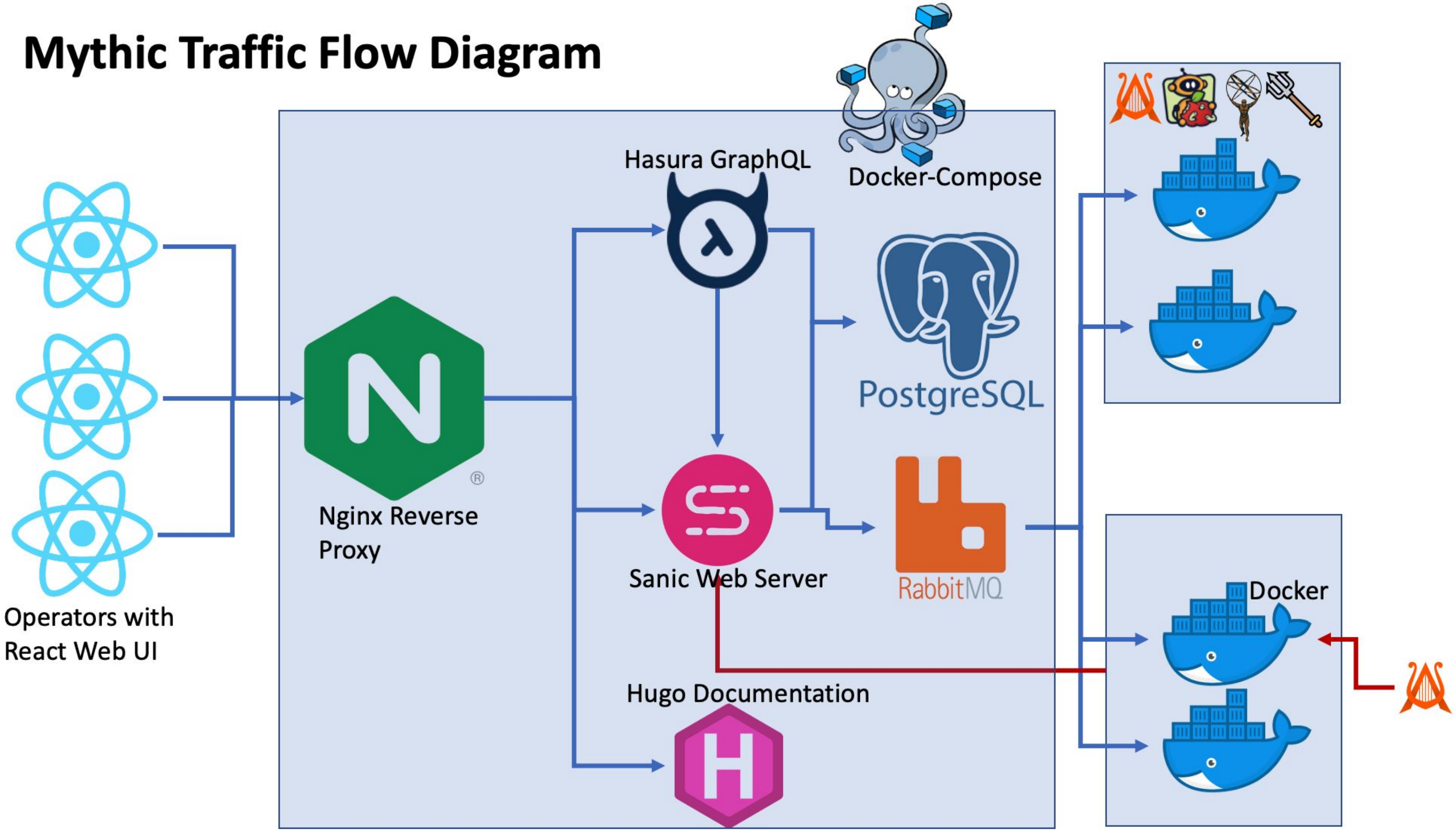
- Licensed
  - NightHawk
  - Brute Ratel (BRC4)

  Minimum price: 3500$/user

- Opensource
  - Covenant
  - Sliver
  - Empire
  - Merlin
  - Shad0w

  Low customization

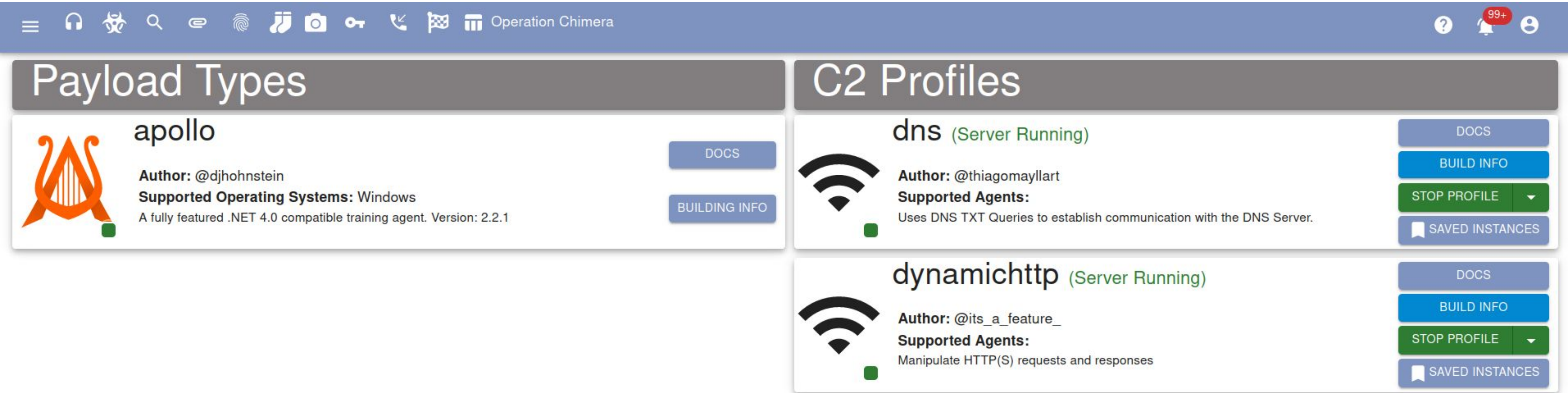
Mythic C2



Documentation is well-written (https://docs.mythic-c2.net/)

Mythic C2

- User interface and APIs are already dev



Operation Chimera

Payload Types

apollo

Author: @djhohnstein
Supported Operating Systems: Windows
A fully featured .NET 4.0 compatible training agent. Version: 2.2.1

C2 Profiles

dns (Server Running)

Author: @thiagomayllart
Supported Agents:
Uses DNS TXT Queries to establish communication with the DNS Server.

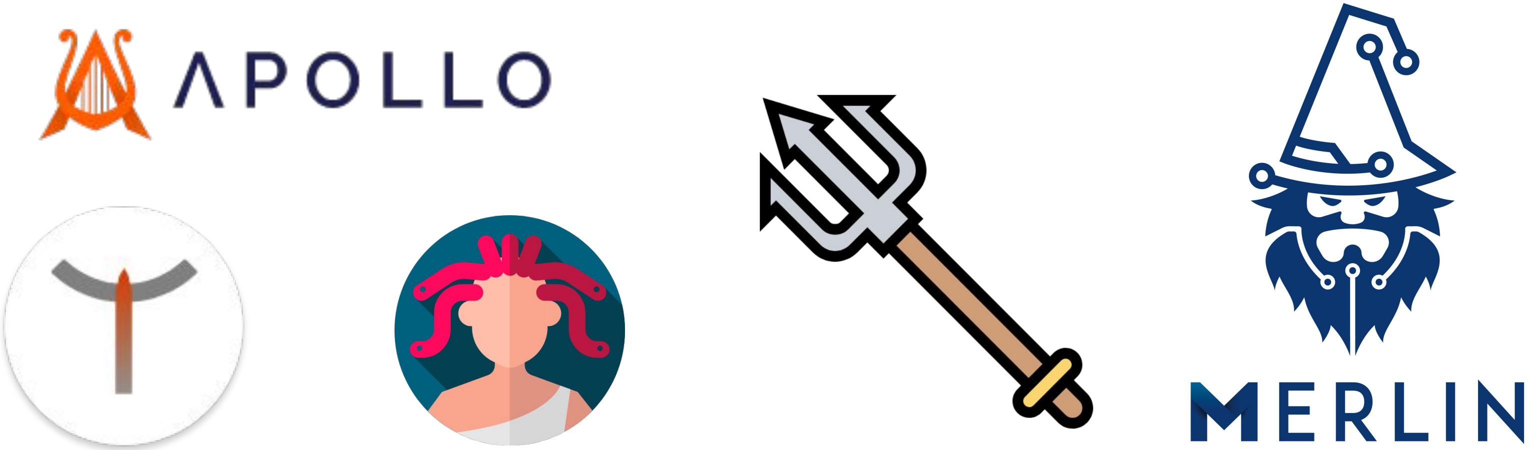
dynamichttp (Server Running)

Author: @its_a_feature_
Supported Agents:
Manipulate HTTP(S) requests and responses

Mythic C2: bring your own implant

Name	Supported Operating System	Language
Apollo	Windows	C#
Athena	Windows, Mac & Linux	.NET Core
NimPlant	Windows, Mac & Linux	Nim
Merlin*	Windows	Go
Tetanus	Windows, Mac & Linux	Rust
Medusa	Mac & Linux	Python

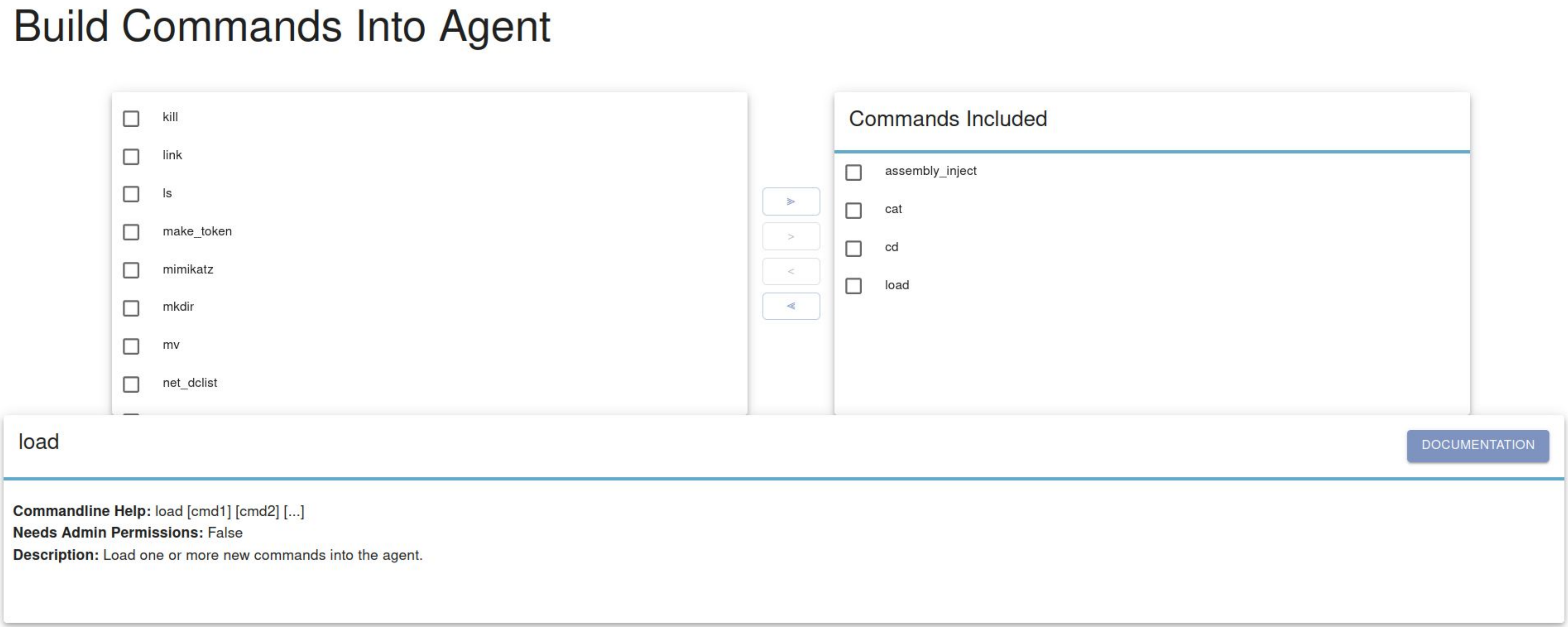
+ your implant



Mythic C2 - Capabilities

- Support several transports: HTTP, Websocket, SMB, TCP, DNS
- SOCKS5 proxy
- Credentials management
- File upload and download
- Translators

Mythic C2: modules import



Build Commands Into Agent

□ kill

□ link

□ ls

□ make_token

□ mimikatz

□ mkdir

□ mv

□ net_dclist

Commands Included

□ assembly_inject

□ cat

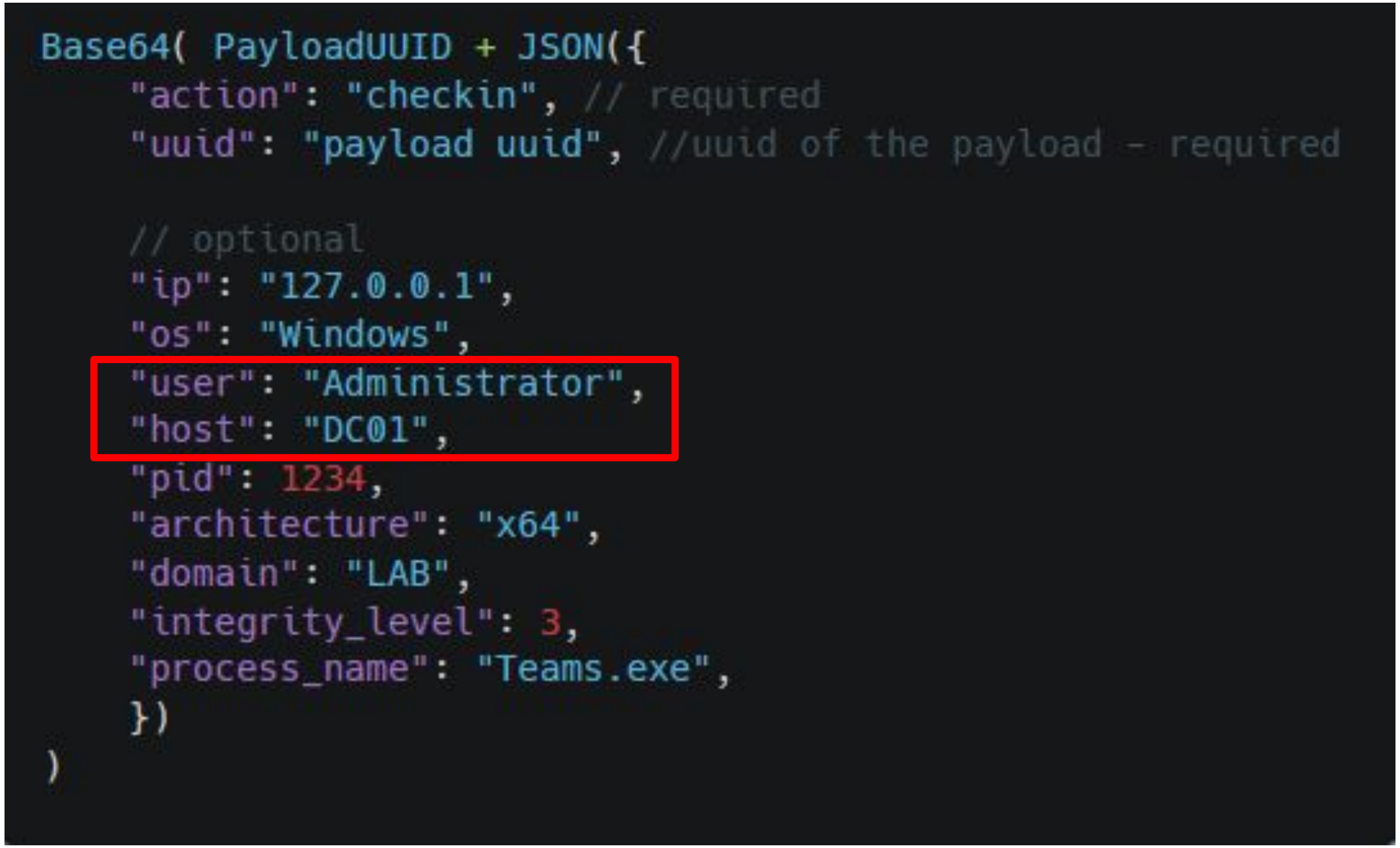
□ cd

□ load

load

Commandline Help: load [cmd1] [cmd2] [...]
Needs Admin Permissions: False
Description: Load one or more new commands into the agent.

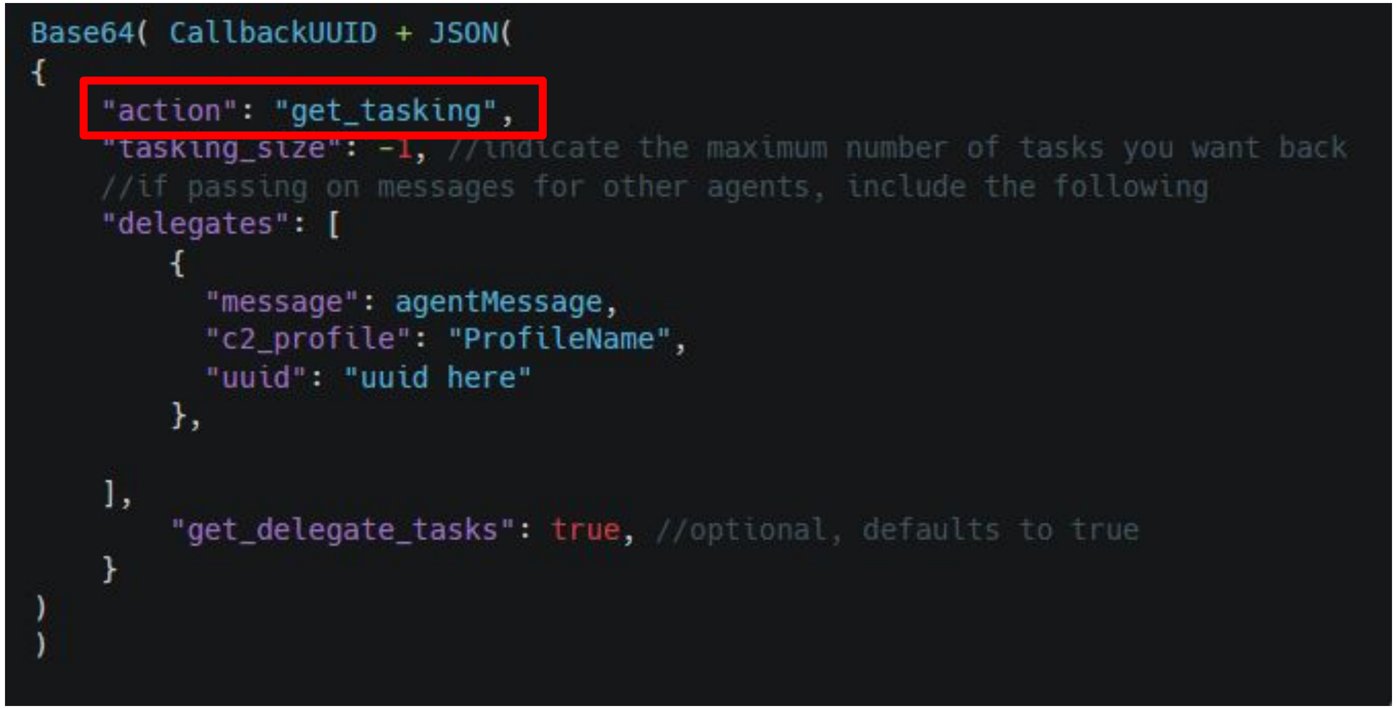
Mythic C2: how the transport looks like



```
Base64( PayloadUUID + JSON({
    "action": "checkin", // required
    "uuid": "payload uuid", //uuid of the payload - required

    // optional
    "ip": "127.0.0.1",
    "os": "Windows",
    "user": "Administrator",
    "host": "DC01",
    "pid": 1234,
    "architecture": "x64",
    "domain": "LAB",
    "integrity_level": 3,
    "process_name": "Teams.exe",
    })
)
```

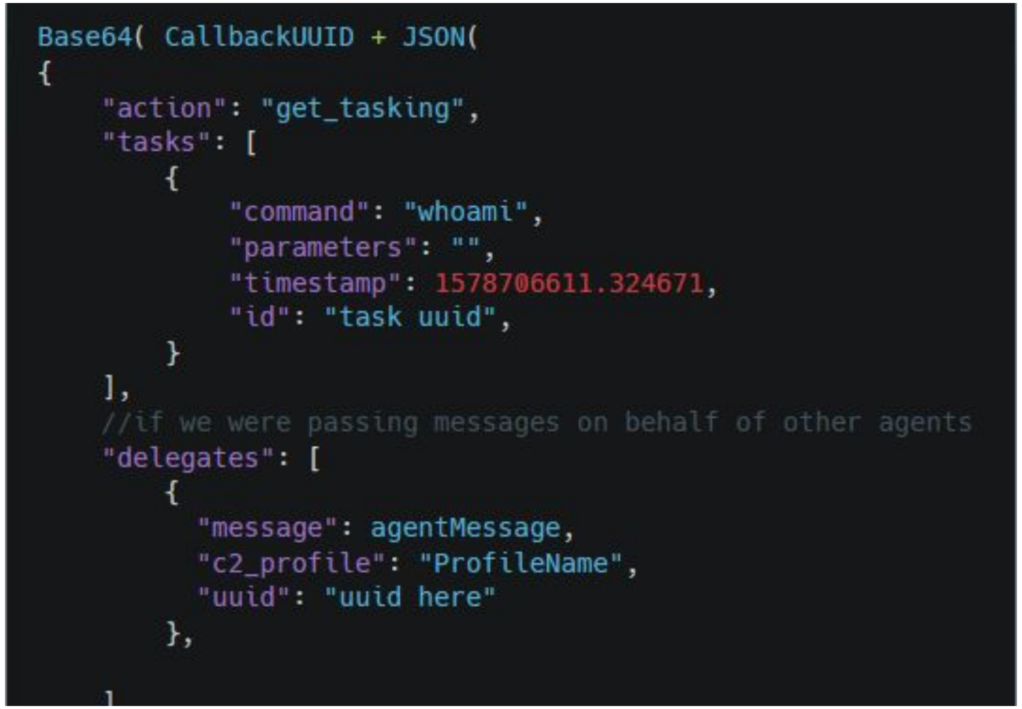
Agent checking format

Mythic C2: how the transport looks like



```
Base64( CallbackUUID + JSON(
{
    "action": "get_tasking",
    "tasking_size": -1, //indicate the maximum number of tasks you want back
    //if passing on messages for other agents, include the following
    "delegates": [
        {
          "message": agentMessage,
          "c2_profile": "ProfileName",
          "uuid": "uuid here"
        },

    ],
        "get_delegate_tasks": true, //optional, defaults to true
    }
)
)
```

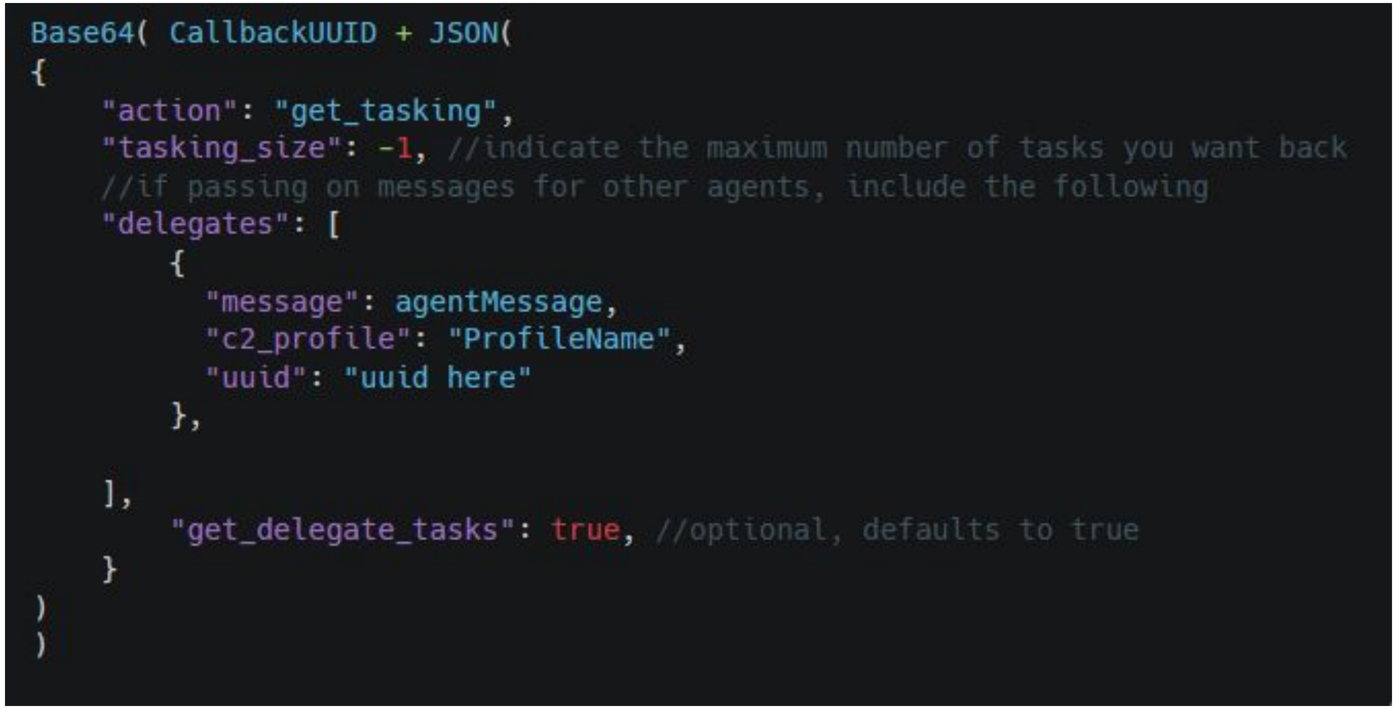
Get Tasking - Request



```
Base64( CallbackUUID + JSON(
{
    "action": "get_tasking",
    "tasks": [
        {
            "command": "whoami",
            "parameters": "",
            "timestamp": 1578706611.324671,
            "id": "task uuid",
        }
    ],
    //if we were passing messages on behalf of other agents
    "delegates": [
        {
          "message": agentMessage,
          "c2_profile": "ProfileName",
          "uuid": "uuid here"
        },

    ]
```

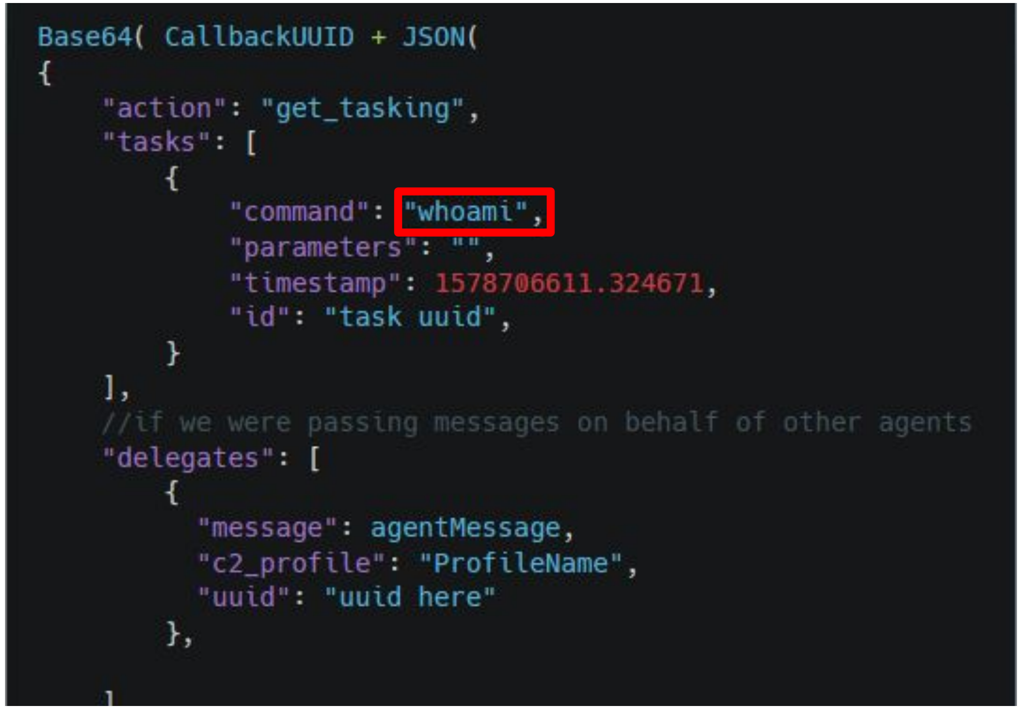
Get Tasking - Response

Mythic C2: how the transport looks like



```
Base64( CallbackUUID + JSON(
{
    "action": "get_tasking",
    "tasking_size": -1, //indicate the maximum number of tasks you want back
    //if passing on messages for other agents, include the following
    "delegates": [
        {
          "message": agentMessage,
          "c2_profile": "ProfileName",
          "uuid": "uuid here"
        },

    ],
        "get_delegate_tasks": true, //optional, defaults to true
    }
)
)
```

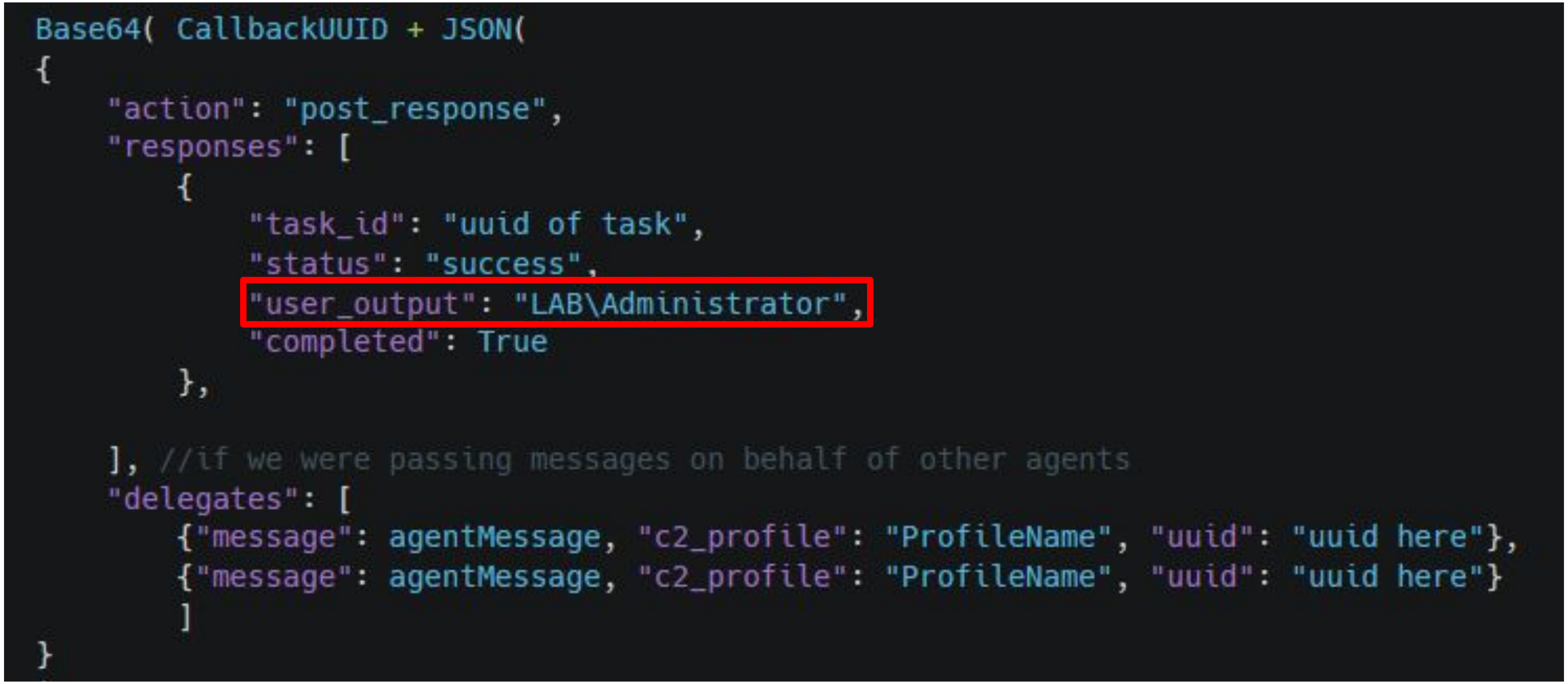
Get Tasking - Request



```
Base64( CallbackUUID + JSON(
{
    "action": "get_tasking",
    "tasks": [
        {
            "command": "whoami",
            "parameters": "",
            "timestamp": 1578706611.324671,
            "id": "task uuid",
        }
    ],
    //if we were passing messages on behalf of other agents
    "delegates": [
        {
          "message": agentMessage,
          "c2_profile": "ProfileName",
          "uuid": "uuid here"
        },

    ]
```

Get Tasking - Response

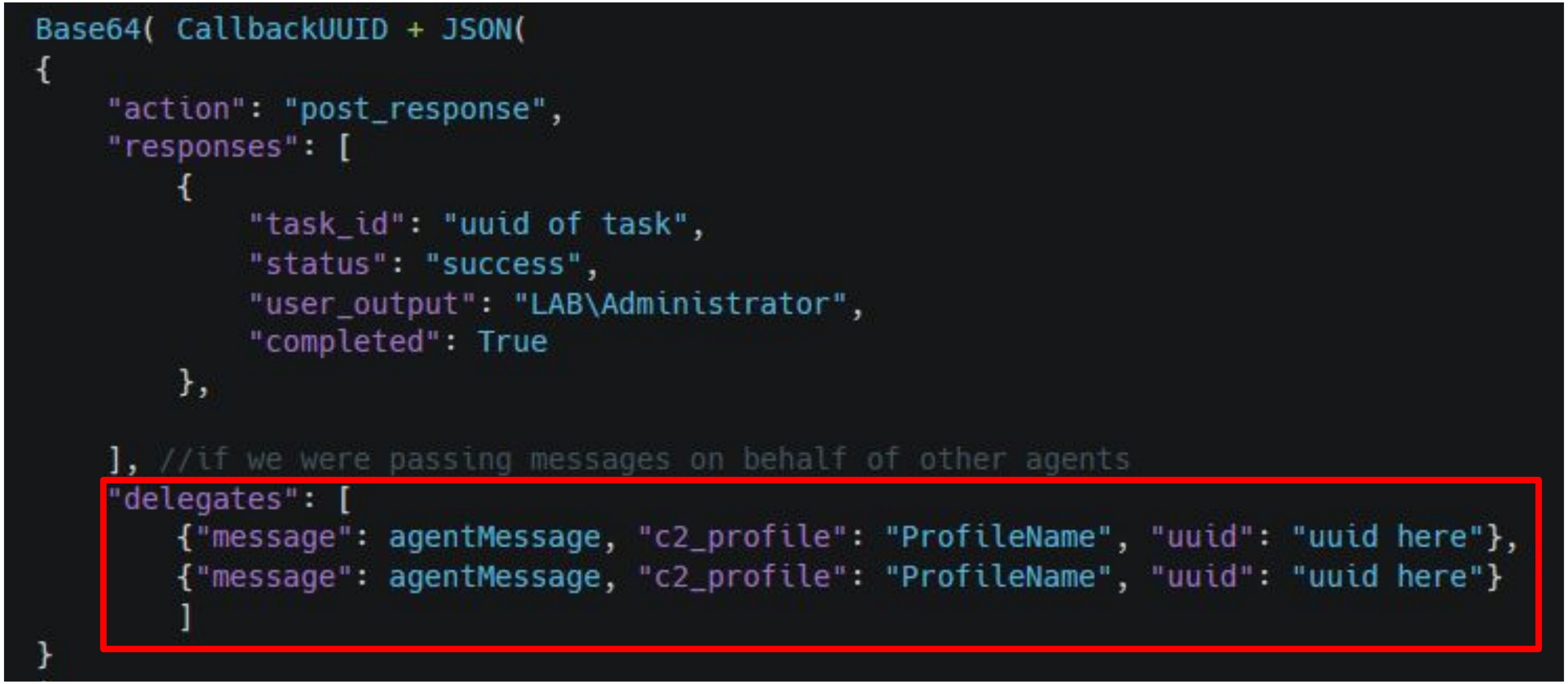
Mythic C2: how the transport looks like



```
Base64( CallbackUUID + JSON(
{
    "action": "post_response",
    "responses": [
        {
            "task_id": "uuid of task",
            "status": "success",
            "user_output": "LAB\Administrator",
            "completed": True
        },

    ], //if we were passing messages on behalf of other agents
    "delegates": [
        {"message": agentMessage, "c2_profile": "ProfileName", "uuid": "uuid here"},
        {"message": agentMessage, "c2_profile": "ProfileName", "uuid": "uuid here"}
        ]
}
```

Post response

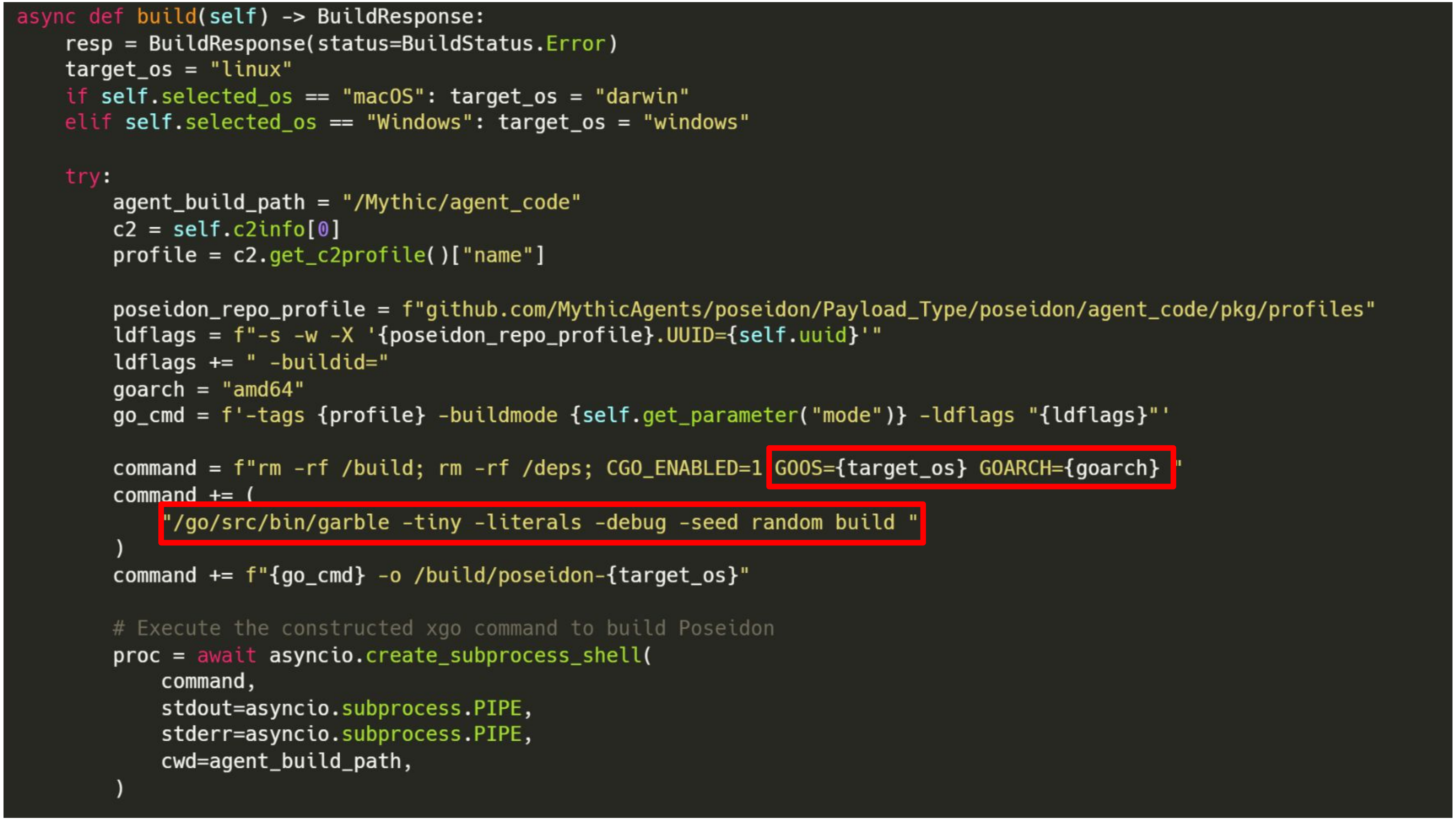
Mythic C2: how the transport looks like



```
Base64( CallbackUUID + JSON(
{
    "action": "post_response",
    "responses": [
        {
            "task_id": "uuid of task",
            "status": "success",
            "user_output": "LAB\Administrator",
            "completed": True
        },

    ], //if we were passing messages on behalf of other agents
    "delegates": [
        {"message": agentMessage, "c2_profile": "ProfileName", "uuid": "uuid here"},
        {"message": agentMessage, "c2_profile": "ProfileName", "uuid": "uuid here"}
        ]
}
```

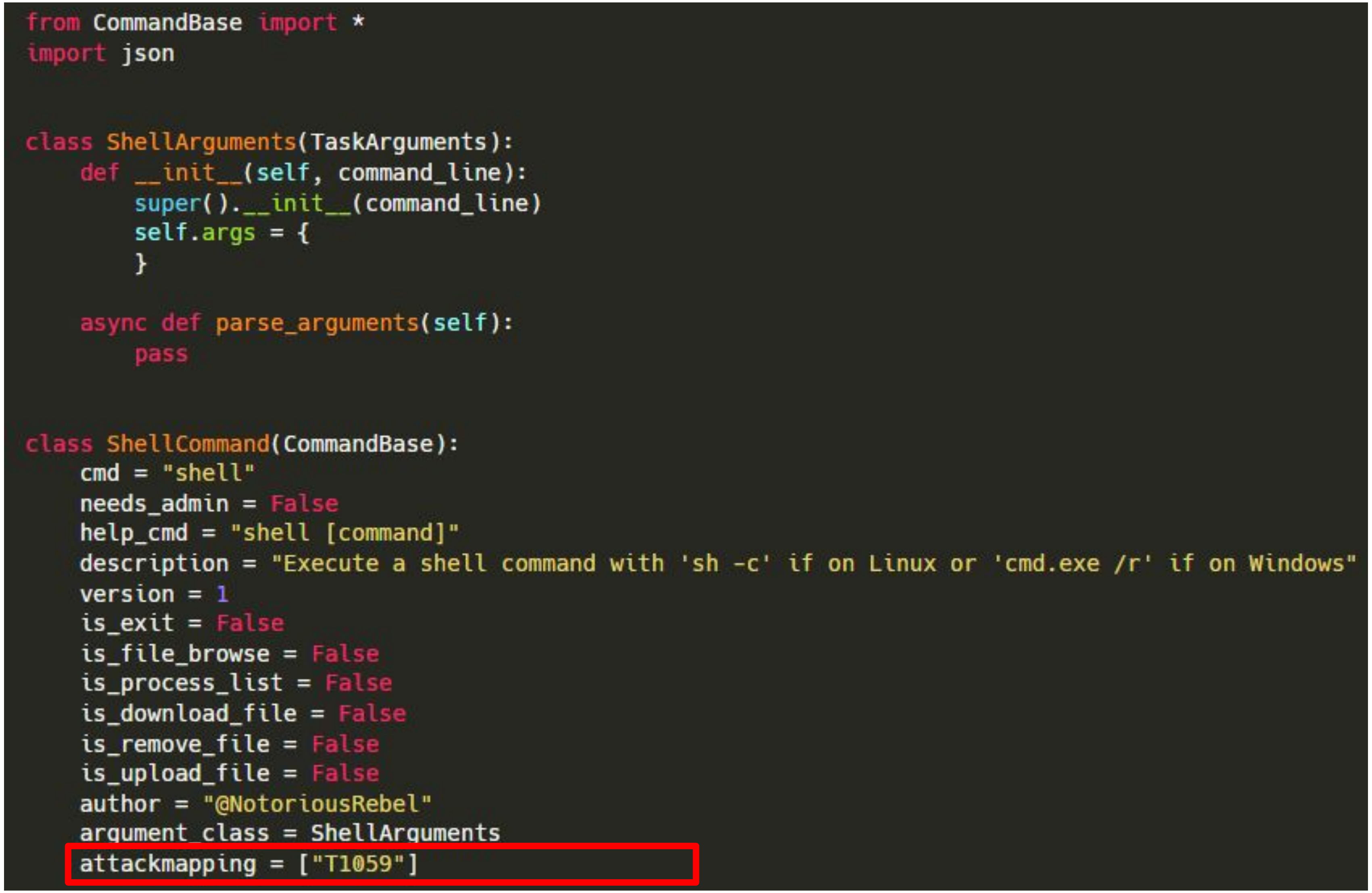
Post response

Mythic C2: handle several systems

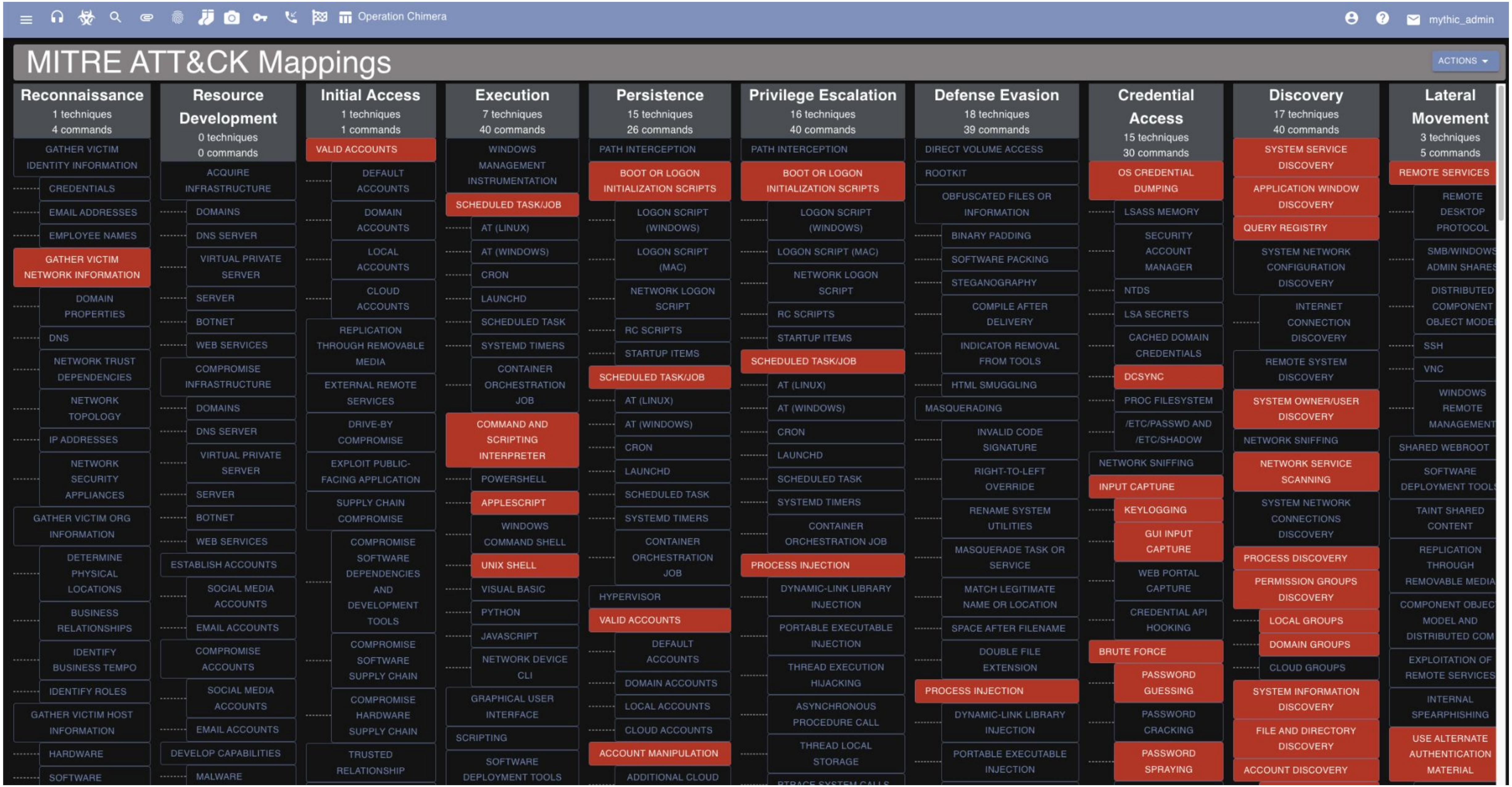


```python
async def build(self) -> BuildResponse:
    resp = BuildResponse(status=BuildStatus.Error)
    target_os = "linux"
    if self.selected_os == "macOS": target_os = "darwin"
    elif self.selected_os == "Windows": target_os = "windows"

    try:
        agent_build_path = "/Mythic/agent_code"
        c2 = self.c2info[0]
        profile = c2.get_c2profile()["name"]

        poseidon_repo_profile = f"github.com/MythicAgents/poseidon/Payload_Type/poseidon/agent_code/pkg/profiles"
        ldflags = f"-s -w -X '{poseidon_repo_profile}.UUID={self.uuid}'"
        ldflags += " -buildid="
        goarch = "amd64"
        go_cmd = f'-tags {profile} -buildmode {self.get_parameter("mode")} -ldflags "{ldflags}"'

        command = f"rm -rf /build; rm -rf /deps; CGO_ENABLED=1 GOOS={target_os} GOARCH={goarch} "
        command += (
            "/go/src/bin/garble -tiny -literals -debug -seed random build "
        )
        command += f"{go_cmd} -o /build/poseidon-{target_os}"

        # Execute the constructed xgo command to build Poseidon
        proc = await asyncio.create_subprocess_shell(
            command,
            stdout=asyncio.subprocess.PIPE,
            stderr=asyncio.subprocess.PIPE,
            cwd=agent_build_path,
        )
```

Mythic C2 Commands handling



```
from CommandBase import *
import json


class ShellArguments(TaskArguments):
    def __init__(self, command_line):
        super().__init__(command_line)
        self.args = {
        }

    async def parse_arguments(self):
        pass


class ShellCommand(CommandBase):
    cmd = "shell"
    needs_admin = False
    help_cmd = "shell [command]"
    description = "Execute a shell command with 'sh -c' if on Linux or 'cmd.exe /r' if on Windows"
    version = 1
    is_exit = False
    is_file_browse = False
    is_process_list = False
    is_download_file = False
    is_remove_file = False
    is_upload_file = False
    author = "@NotoriousRebel"
    argument_class = ShellArguments
    attackmapping = ["T1059"]
```



MITRE ATT&CK Mappings

Reconnaissance
1 techniques
4 commands

- GATHER VICTIM IDENTITY INFORMATION
  - CREDENTIALS
  - EMAIL ADDRESSES
  - EMPLOYEE NAMES
- GATHER VICTIM NETWORK INFORMATION
  - DOMAIN PROPERTIES
  - DNS
  - NETWORK TRUST DEPENDENCIES
  - NETWORK TOPOLOGY
  - IP ADDRESSES
  - NETWORK SECURITY APPLIANCES
- GATHER VICTIM ORG INFORMATION
  - DETERMINE PHYSICAL LOCATIONS
  - BUSINESS RELATIONSHIPS
  - IDENTIFY BUSINESS TEMPO
  - IDENTIFY ROLES
- GATHER VICTIM HOST INFORMATION
  - HARDWARE
  - SOFTWARE

Resource Development
0 techniques
0 commands

- ACQUIRE INFRASTRUCTURE
  - DOMAINS
  - DNS SERVER
  - VIRTUAL PRIVATE SERVER
  - SERVER
  - BOTNET
  - WEB SERVICES
- COMPROMISE INFRASTRUCTURE
  - DOMAINS
  - DNS SERVER
  - VIRTUAL PRIVATE SERVER
  - SERVER
  - BOTNET
  - WEB SERVICES
- ESTABLISH ACCOUNTS
  - SOCIAL MEDIA ACCOUNTS
  - EMAIL ACCOUNTS
- COMPROMISE ACCOUNTS
  - SOCIAL MEDIA ACCOUNTS
  - EMAIL ACCOUNTS
- DEVELOP CAPABILITIES
  - MALWARE

Initial Access
1 techniques
1 commands

- VALID ACCOUNTS
  - DEFAULT ACCOUNTS
  - DOMAIN ACCOUNTS
  - LOCAL ACCOUNTS
  - CLOUD ACCOUNTS
- REPLICATION THROUGH REMOVABLE MEDIA
- EXTERNAL REMOTE SERVICES
- DRIVE-BY COMPROMISE
- EXPLOIT PUBLIC-FACING APPLICATION
- SUPPLY CHAIN COMPROMISE
  - COMPROMISE SOFTWARE DEPENDENCIES AND DEVELOPMENT TOOLS
  - COMPROMISE SOFTWARE SUPPLY CHAIN
  - COMPROMISE HARDWARE SUPPLY CHAIN
- TRUSTED RELATIONSHIP

Execution
7 techniques
40 commands

- WINDOWS MANAGEMENT INSTRUMENTATION
- SCHEDULED TASK/JOB
  - AT (LINUX)
  - AT (WINDOWS)
  - CRON
  - LAUNCHD
  - SCHEDULED TASK
  - SYSTEMD TIMERS
  - CONTAINER ORCHESTRATION JOB
- COMMAND AND SCRIPTING INTERPRETER
  - POWERSHELL
  - APPLESCRIPT
  - WINDOWS COMMAND SHELL
  - UNIX SHELL
  - VISUAL BASIC
  - PYTHON
  - JAVASCRIPT
  - NETWORK DEVICE CLI
- GRAPHICAL USER INTERFACE
- SCRIPTING
- SOFTWARE DEPLOYMENT TOOLS

Persistence
15 techniques
26 commands

- PATH INTERCEPTION
- BOOT OR LOGON INITIALIZATION SCRIPTS
  - LOGON SCRIPT (WINDOWS)
  - LOGON SCRIPT (MAC)
  - NETWORK LOGON SCRIPT
  - RC SCRIPTS
  - STARTUP ITEMS
- SCHEDULED TASK/JOB
  - AT (LINUX)
  - AT (WINDOWS)
  - CRON
  - LAUNCHD
  - SCHEDULED TASK
  - SYSTEMD TIMERS
  - CONTAINER ORCHESTRATION JOB
- HYPERVISOR
- VALID ACCOUNTS
  - DEFAULT ACCOUNTS
  - DOMAIN ACCOUNTS
  - LOCAL ACCOUNTS
  - CLOUD ACCOUNTS
- ACCOUNT MANIPULATION
  - ADDITIONAL CLOUD

Privilege Escalation
16 techniques
40 commands

- PATH INTERCEPTION
- BOOT OR LOGON INITIALIZATION SCRIPTS
  - LOGON SCRIPT (WINDOWS)
  - LOGON SCRIPT (MAC)
  - NETWORK LOGON SCRIPT
  - RC SCRIPTS
  - STARTUP ITEMS
- SCHEDULED TASK/JOB
  - AT (LINUX)
  - AT (WINDOWS)
  - CRON
  - LAUNCHD
  - SCHEDULED TASK
  - SYSTEMD TIMERS
  - CONTAINER ORCHESTRATION JOB
- PROCESS INJECTION
  - DYNAMIC-LINK LIBRARY INJECTION
  - PORTABLE EXECUTABLE INJECTION
  - THREAD EXECUTION HIJACKING
  - ASYNCHRONOUS PROCEDURE CALL
  - THREAD LOCAL STORAGE
  - PTRACE SYSTEM CALLS

Defense Evasion
18 techniques
39 commands

- DIRECT VOLUME ACCESS
- ROOTKIT
- OBFUSCATED FILES OR INFORMATION
  - BINARY PADDING
  - SOFTWARE PACKING
  - STEGANOGRAPHY
  - COMPILE AFTER DELIVERY
  - INDICATOR REMOVAL FROM TOOLS
  - HTML SMUGGLING
- MASQUERADING
  - INVALID CODE SIGNATURE
  - RIGHT-TO-LEFT OVERRIDE
  - RENAME SYSTEM UTILITIES
  - MASQUERADE TASK OR SERVICE
  - MATCH LEGITIMATE NAME OR LOCATION
  - SPACE AFTER FILENAME
  - DOUBLE FILE EXTENSION
- PROCESS INJECTION
  - DYNAMIC-LINK LIBRARY INJECTION
  - PORTABLE EXECUTABLE INJECTION

Credential Access
15 techniques
30 commands

- OS CREDENTIAL DUMPING
  - LSASS MEMORY
  - SECURITY ACCOUNT MANAGER
  - NTDS
  - LSA SECRETS
  - CACHED DOMAIN CREDENTIALS
  - DCSYNC
  - PROC FILESYSTEM
  - /ETC/PASSWD AND /ETC/SHADOW
- NETWORK SNIFFING
- INPUT CAPTURE
  - KEYLOGGING
  - GUI INPUT CAPTURE
  - WEB PORTAL CAPTURE
  - CREDENTIAL API HOOKING
- BRUTE FORCE
  - PASSWORD GUESSING
  - PASSWORD CRACKING
  - PASSWORD SPRAYING

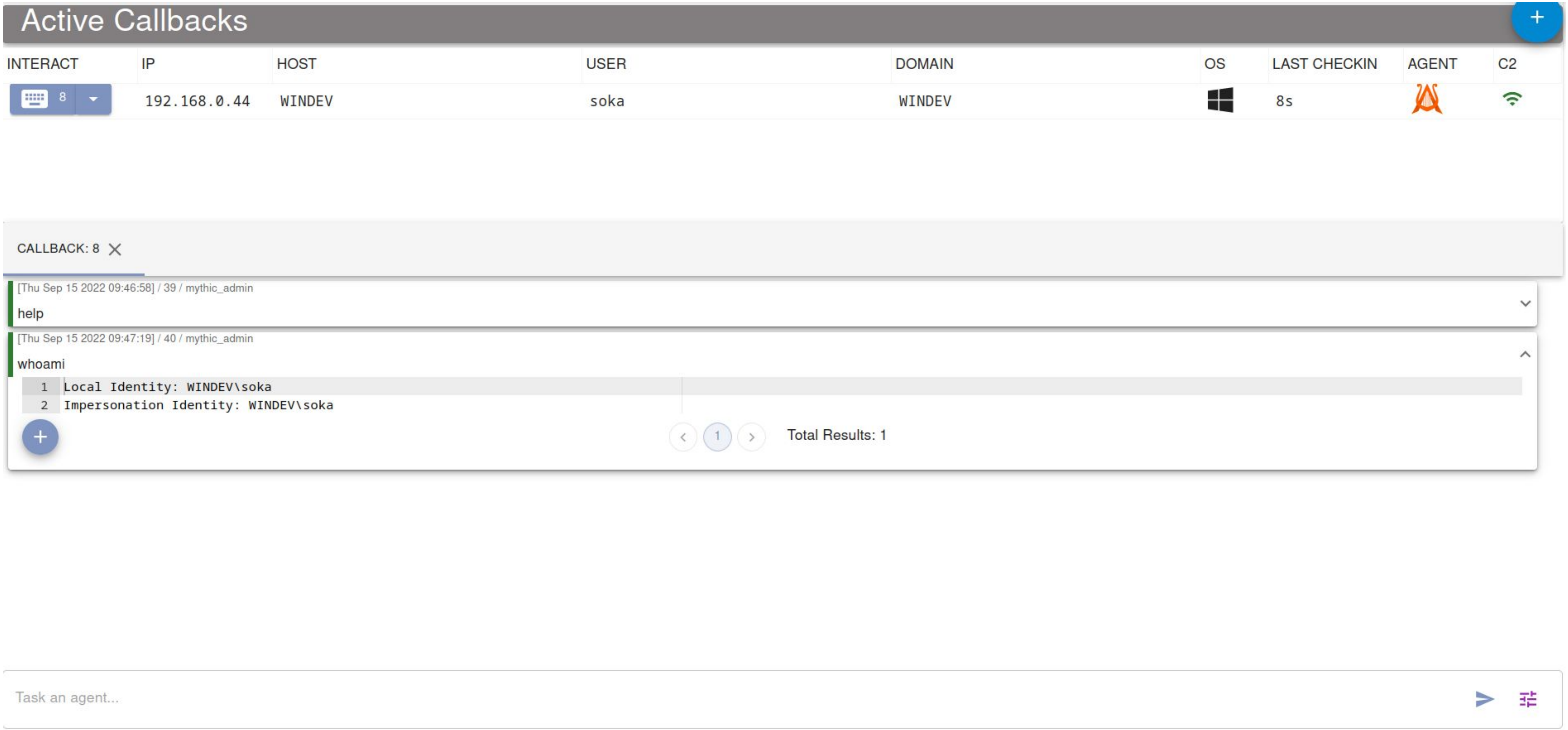
Discovery
17 techniques
40 commands

- SYSTEM SERVICE DISCOVERY
- APPLICATION WINDOW DISCOVERY
- QUERY REGISTRY
- SYSTEM NETWORK CONFIGURATION DISCOVERY
  - INTERNET CONNECTION DISCOVERY
- REMOTE SYSTEM DISCOVERY
- SYSTEM OWNER/USER DISCOVERY
- NETWORK SNIFFING
- NETWORK SERVICE SCANNING
- SYSTEM NETWORK CONNECTIONS DISCOVERY
- PROCESS DISCOVERY
- PERMISSION GROUPS DISCOVERY
  - LOCAL GROUPS
  - DOMAIN GROUPS
  - CLOUD GROUPS
- SYSTEM INFORMATION DISCOVERY
- FILE AND DIRECTORY DISCOVERY
- ACCOUNT DISCOVERY

Lateral Movement
3 techniques
5 commands

- REMOTE SERVICES
  - REMOTE DESKTOP PROTOCOL
  - SMB/WINDOWS ADMIN SHARES
  - DISTRIBUTED COMPONENT OBJECT MODEL
  - SSH
  - VNC
  - WINDOWS REMOTE MANAGEMENT
- SHARED WEBROOT
- SOFTWARE DEPLOYMENT TOOLS
- TAINT SHARED CONTENT
- REPLICATION THROUGH REMOVABLE MEDIA
- COMPONENT OBJECT MODEL AND DISTRIBUTED COM
- EXPLOITATION OF REMOTE SERVICES
- INTERNAL SPEARPHISHING
- USE ALTERNATE AUTHENTICATION MATERIAL

Questions



Active Callbacks

INTERACT	IP	HOST	USER	DOMAIN	OS	LAST CHECKIN	AGENT	C2
8	192.168.0.44	WINDEV	soka	WINDEV		8s		

CALLBACK: 8

[Thu Sep 15 2022 09:46:58] / 39 / mythic_admin

help

[Thu Sep 15 2022 09:47:19] / 40 / mythic_admin

whoami

```
Local Identity: WINDEV\soka
Impersonation Identity: WINDEV\soka
```