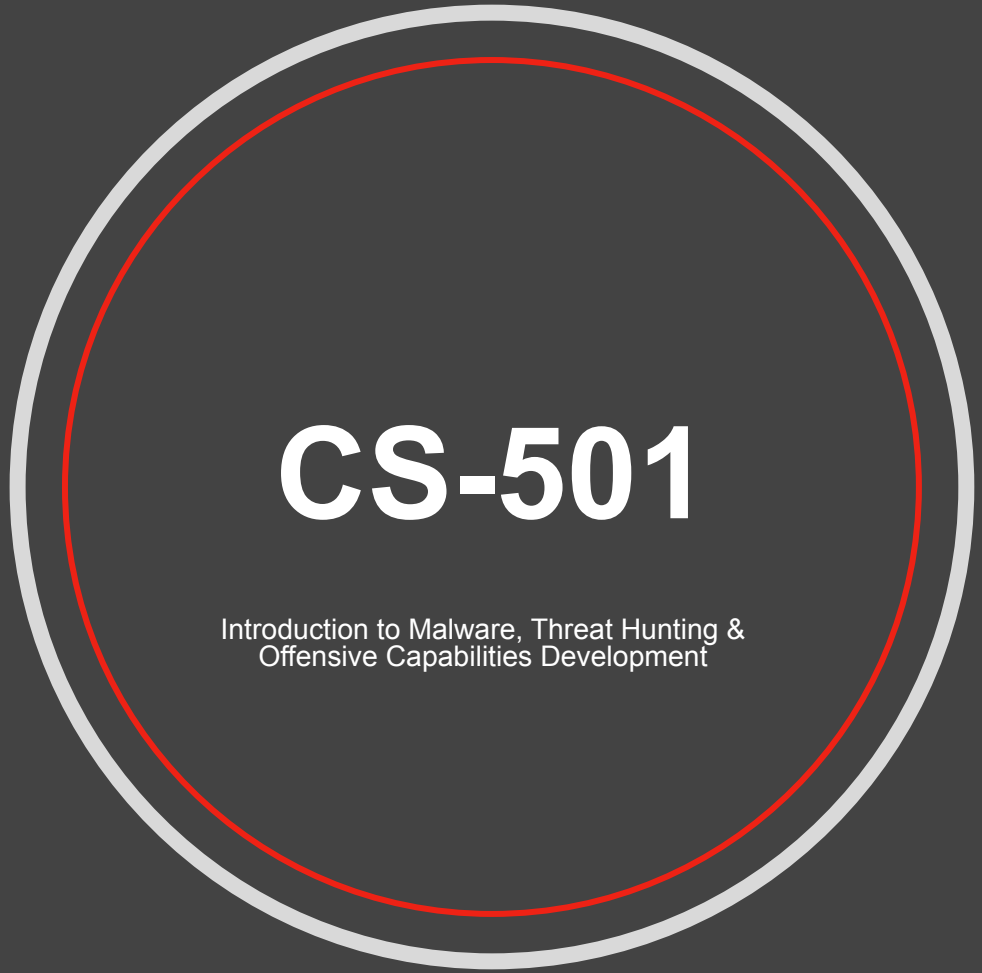


CS-501

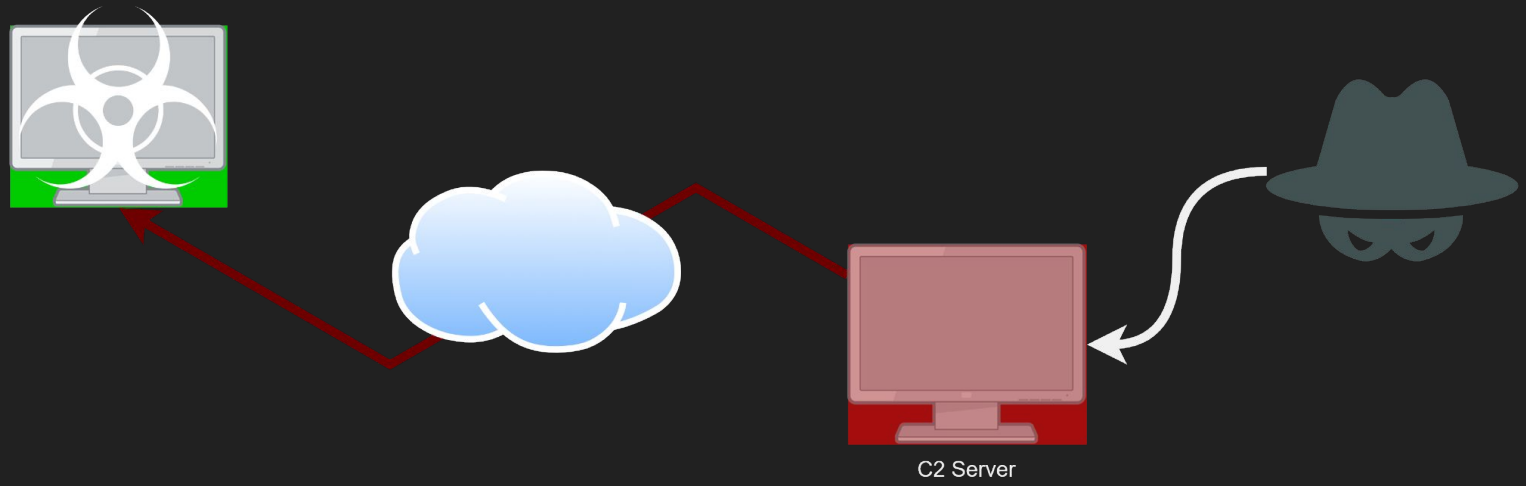
Introduction to Malware, Threat Hunting &
Offensive Capabilities Development

Lecture 8: RPC, HTTP, Flask, WinHTTP

Example C2 Architecture

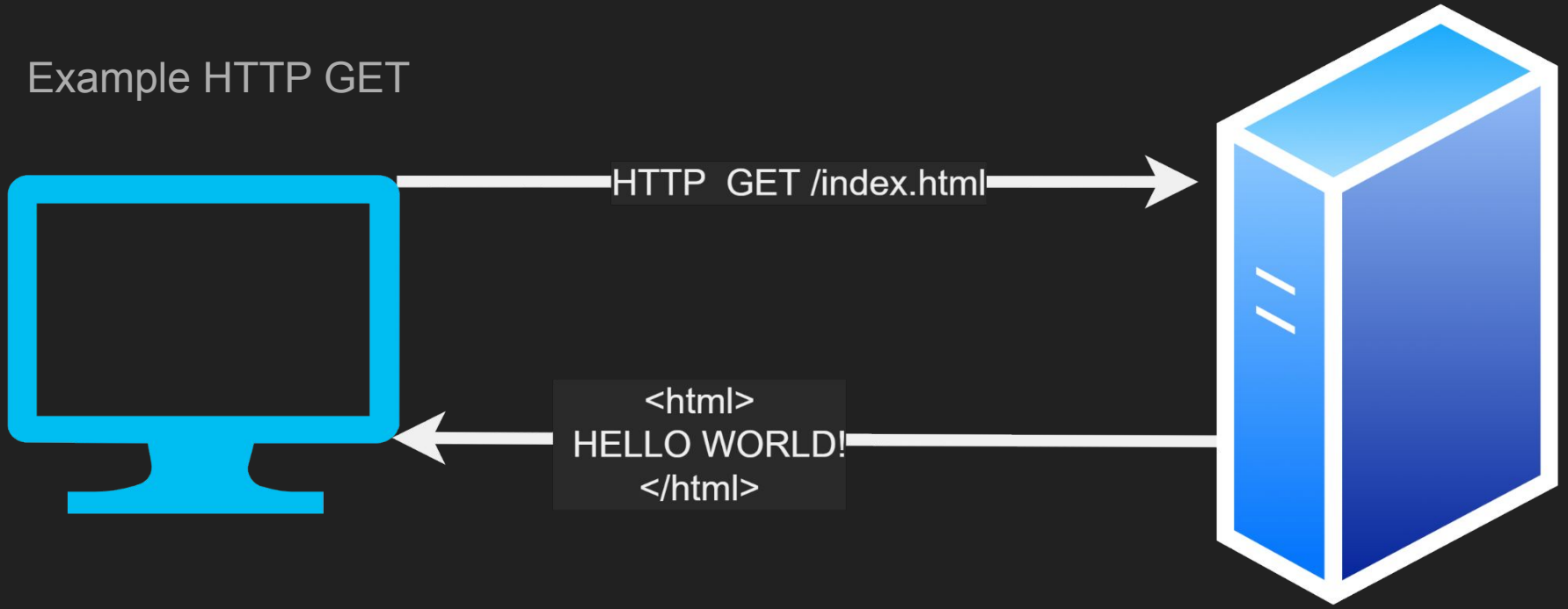
Infected Machine connects directly to the C2

Commands are issued by the operator

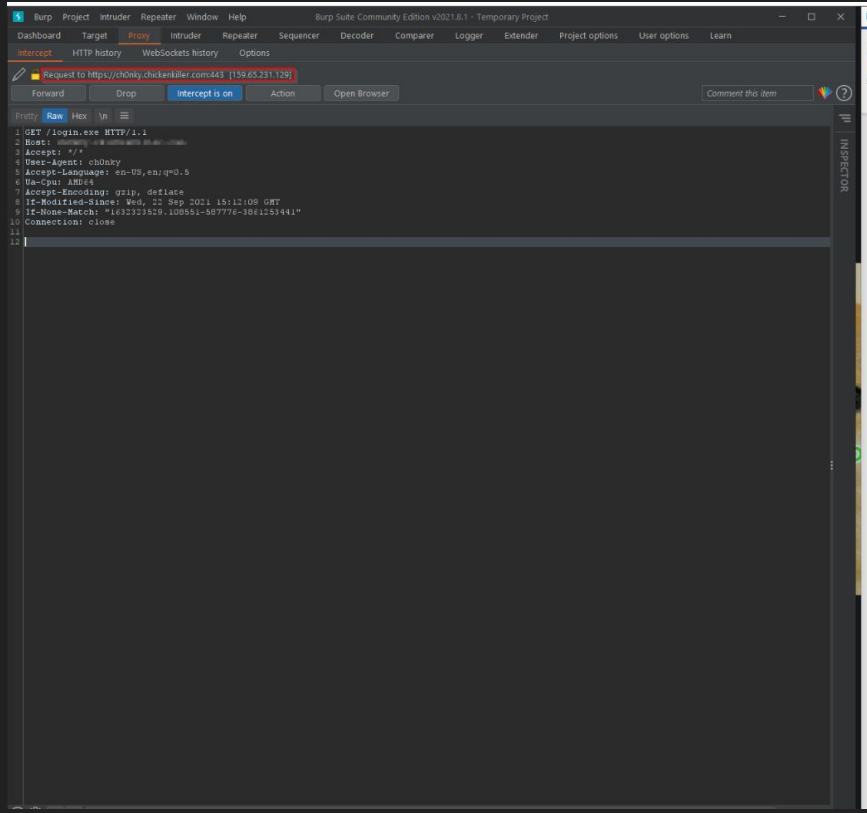


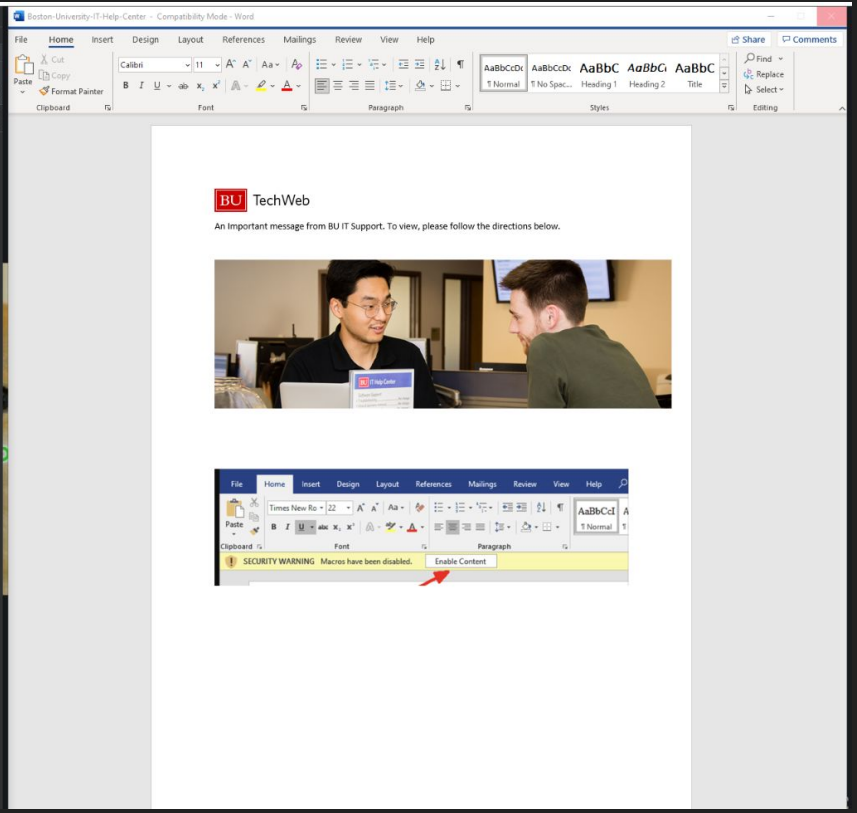
C2 Channel: HTTP

Example HTTP GET

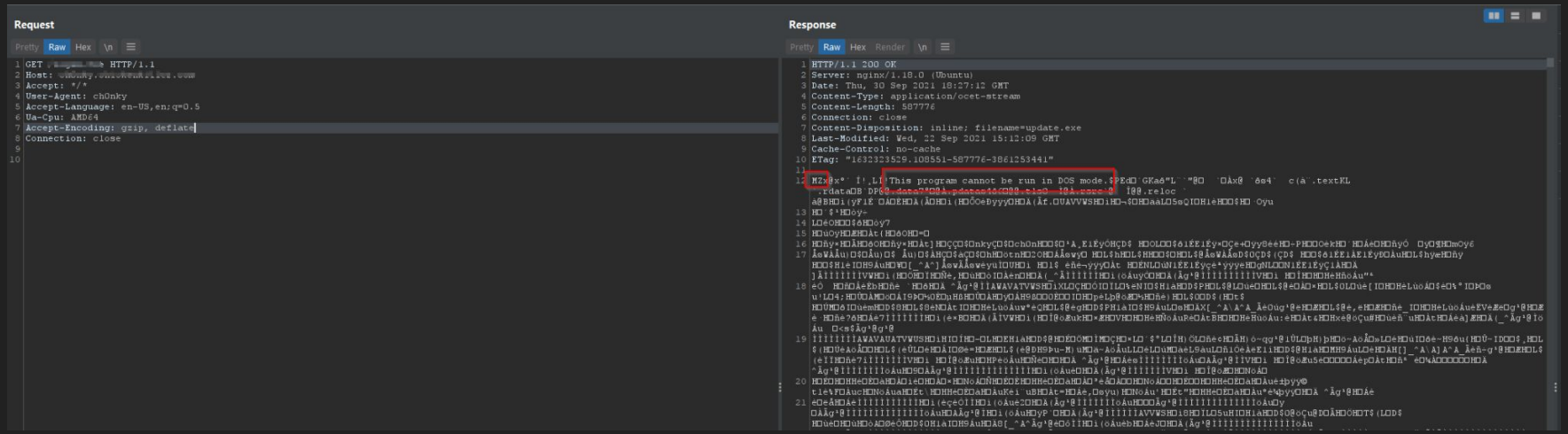


Example from HW1





Example HTTP Dropper



HTTP RPC

RPC: Remote Procedure Call

The Malware's RPC is the protocol used to control the malware from the server. This includes issuing commands for the malware to execute, data to upload/download..etc

Let's consider the simple example of malware that only wishes to maintain a backdoor to a target, and execute powershell commands.

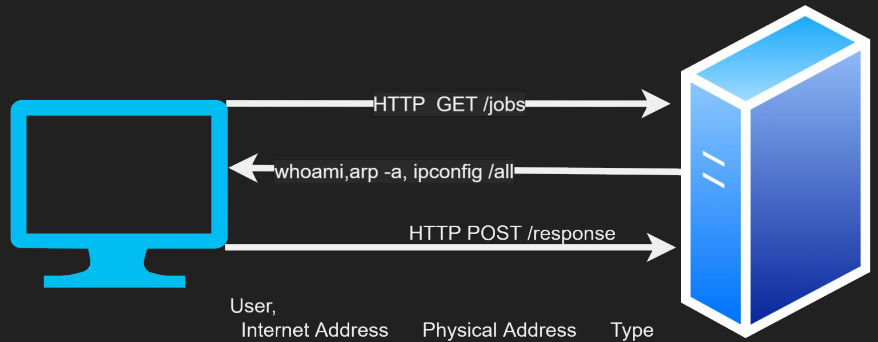
Example 1: HTTP Reverse Shell

Malware makes an HTTP GET Request to the endpoint /commands

Server responds with a list of shell commands it wishes the implant to execute

Malware responds with a post request containing the output of those commands

Example 1: HTTP Reverse Shell



User,

Internet Address	Physical Address	Type
10.16.0.1		dynamic
10.16.255.255		static
224.0.0.22		static
224.0.0.251		static
224.0.0.252		static
239.255.255.250		static
255.255.255.255		static

Connection-specific DNS Suffix . :
Description . . . . . : Intel(R) PRO/1000 MT Desktop Adapter #2
Physical Address. . . . . : 08-00-27-D6-2A-9E
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::a47d:1f11:8f29:22e6%9(Preferred)
IPv4 Address. . . . . : 10.10.10.3(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 10.10.10.2
DHCPv6 IAID . . . . . : 319291431
DHCPv6 Client DUID. . . . . : 00-01-00-01-28-58-BD-84-00-15-5D-24-F4-CD
DNS Servers . . . . . : 10.10.10.2
NetBIOS over Tcpip. . . . . : Enabled

Discussion: Building an RPC

Flask

- Lightweight, no frills HTTP server
- Routing is handled by decorators
- Contains various helper functions to easily parse and respond to requests
- Has a rich ecosystem for different database plugins
- Is not production ready in and of itself-- requires a Web Service Gateway Interface (WSGI)

Terminology

3 programs:

Implant: the malicious backdoor

Teamserver: the server controlling the implant

Client: the client code used by the operator to control the implant

Basic Imports for our teamserver

```
from flask import Flask, request, jsonify
import multiprocessing as mp
```

Flask: flask application

request: http request object

Jsonify: method to create a json response

From multiprocessing, we will use mp.Lock() to protect access to shared resources

Flask Hello world

```
from flask import Flask

app = Flask(__name__)

@app.route("/hello")
def  hello():
    return "Hello world!"
```

Setup: Server

- Server contains 4 routes
- /queue: returns the status of all of the tasks queued for the implant
- /secret: the secret endpoint to send requests to
- /tasks: endpoint for the implant to pulldown tasks from
- /response: endpoint that the implant sends the response to

Contains a shared task queue

Demo using Python

How can we make this better?

- Identify implants by guid: register
- Poll server for responses
- Authenticate to server (client, and implant)
- Discussion