Practical Malware Analysis & Triage

Malware Analysis Report

Backdoor.srvupdat.exe

Jan 2023 | @hegdenischay | v1.0

Table of Contents

Executive Summary

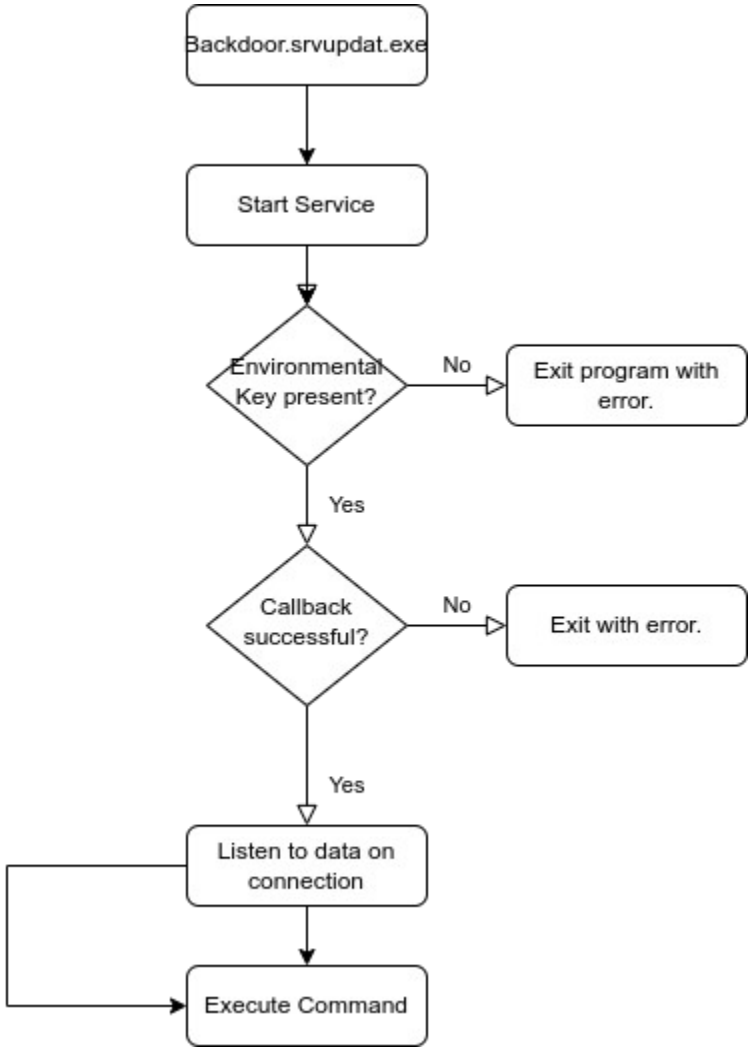
SHA256 hash	914CAD877A41F12BB0998BC1C28D04EE2FB33C4538707547CAD726B41F7D01C3

Backdoor.srvupdat.exe is a golang-based reverse shell malware sample, first uploaded onto the PMAT-labs GitHub repository on Oct 19th, 2021. It was first uploaded to VirusTotal on Dec 15th, 2021 and is currently reported as Malicious by four AV vendors. This runs on the Microsoft Windows Operating system, and is written for the x64 architecture. It consists of a single payload that contacts two URLs, opens a reverse shell to a particular IP and installs a Service. Symptoms of infection include reaching out to any of the URLs listed in Appendix B, or a blank `cmd.exe` showing up on a desktop. The associated service installed was not able to be found in this investigation.

YARA signature rules are attached in Appendix A. Malware sample has already been found on VirusTotal with confidence from several vendors that it is malicious.

High-Level Technical Summary

`Backdoor.srvupdat.exe` consists of a single golang binary that attempts to contact a URL as part of an Environmental Keying tactic. If that URL exists, making it highly likely that the binary is running inside a simulated internet, the malware then proceeds to make a TCP connection to its callback URL, which initiates a Reverse Shell.



Malware Composition

Backdoor.srvupdat.exe consists of a single golang binary.

File Name	SHA256 Hash
Backdoor.srvupdat.exe	914cad877a41f12bb0998bc1c28d04ee2fb33c4538707547cad726b41f7d01c3

This is the binary that spawns a reverse shell and installs a service. Please note that at the time of writing of this report, the service was not found to be running in the system, so this may not be able to persist in the system with this method.

Basic Static Analysis

Basic static analysis of program strings reveals that the binary is using the golang language, since golang-related strings abound in the `strings` output. For example:
It has the path on the Malware author's machine where the program was compiled.

It also has multiple strings that reference common golang libraries like `net/http.DefaultClient`, `net/http`, `syscall`, and the `github.com/*` syntax that is very common when using modules hosted on github.

```
net/http.DefaultClient
go.itab.*bufio.Reader,io.Reader
os.Stdout
go.itab.*os.File,io.Writer
runtime.writeBarrier
main.(*program).run.stkobj
go.itab.*main.program,github.com/kardianos/service.Interface
runtime.zerobase
main.logger
main.main.stkobj
runtime.algarray
main..inittask
bufio..inittask
fmt..inittask
log..inittask
net..inittask
os/exec..inittask
syscall..inittask
net/http..inittask
github.com/kardianos/service..inittask
```

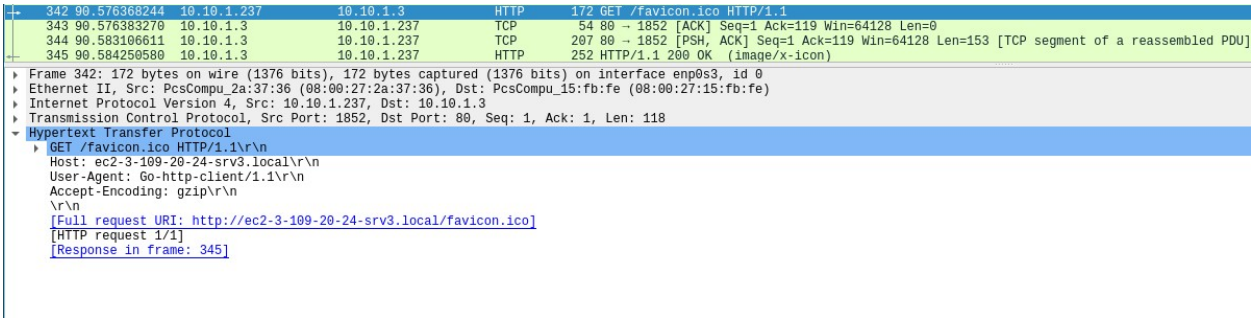
Searching for `github.com/` in the strings output gives an interesting string called `github.com/kardianos/service`, which turns out to be a golang module to start golang functions as a Windows/Linux/OSX service.

PEStudio finds multiple indicators, including some abnormally large strings consistent with behavior in golang, since golang strings do not terminate with a null character.

indicator (63)	detail	level
strings > flag	26	1
file > extensions (Ransomware | Wiper) > count	44	1
imports > flag	5	1
string > size > suspicious	1164 bytes	2
string > size > suspicious	1168 bytes	2
string > size > suspicious	1253 bytes	2
string > size > suspicious	1253 bytes	2
string > size > suspicious	1255 bytes	2
string > size > suspicious	1255 bytes	2
string > size > suspicious	1380 bytes	2
string > size > suspicious	1386 bytes	2
string > size > suspicious	1492 bytes	2
string > size > suspicious	1507 bytes	2
string > size > suspicious	1778 bytes	2
string > size > suspicious	1795 bytes	2
string > size > suspicious	1799 bytes	2
string > size > suspicious	2118 bytes	2
string > size > suspicious	2321 bytes	2
string > size > suspicious	2783 bytes	2
string > size > suspicious	3119 bytes	2
string > size > suspicious	4897 bytes	2
string > size > suspicious	5419 bytes	2
string > size > suspicious	5466 bytes	2
string > size > suspicious	6481 bytes	2
string > size > suspicious	8667 bytes	2
file > compiler > stamp	Thu Jan 01 00:00:00 1970	2
file > checksum > invalid	0x00000000	3
entry-point > location	0x0005C4D0	3

Basic Dynamic Analysis

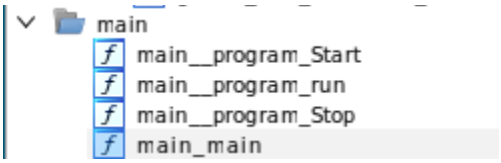
Running it in the test environment with INetSim properly configured gives our first network indicator of compromise.



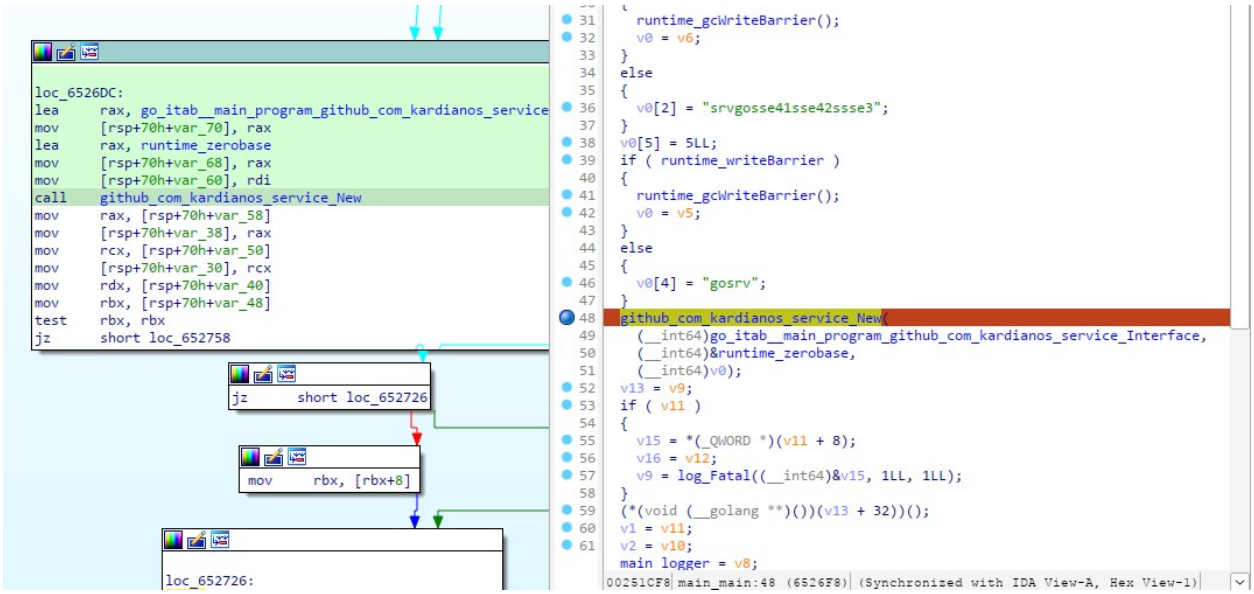
This shows a GET request to the `favicon.ico` file on the hostname `hxxp://ec2-3-109-20-24-srv3.local`. This does not correspond to an actual address on the internet, since it ends in local. It is most likely present as part of Environmental Keying, to make sure that the binary does not run outside of a machine with a simulated internet.

Advanced Static Analysis

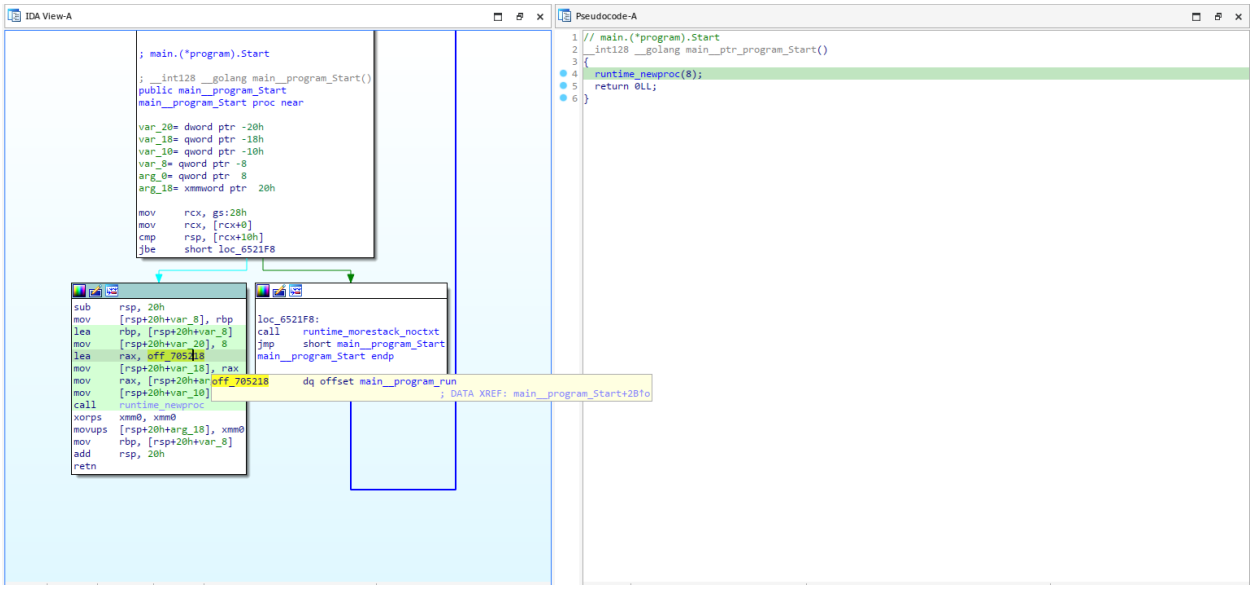
Advanced Static Analysis was done in IDA 64-bit. Only one function is of interest, `main__program_run()`. This function is part of three non-library functions, as shown in the figure below:



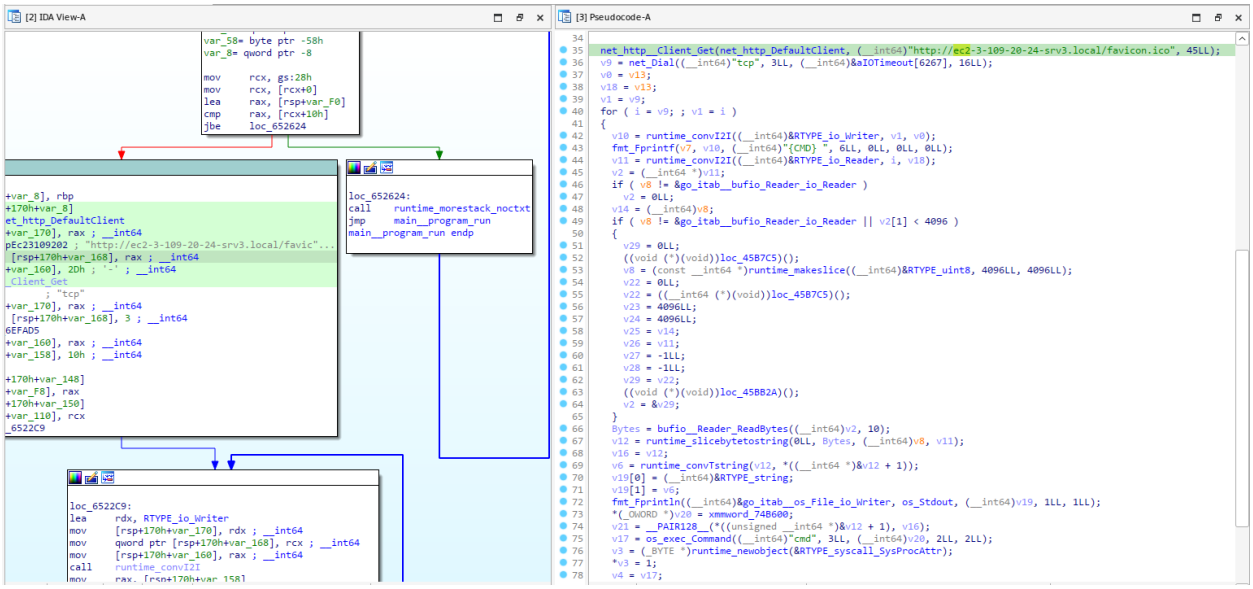
main_main has some interesting strings, including a call to `github_com_kardianos_service_New`. The source code shows that this particular function accepts a program struct and a config. Reading the source code suggests that the service name in the config might be "`srv`".



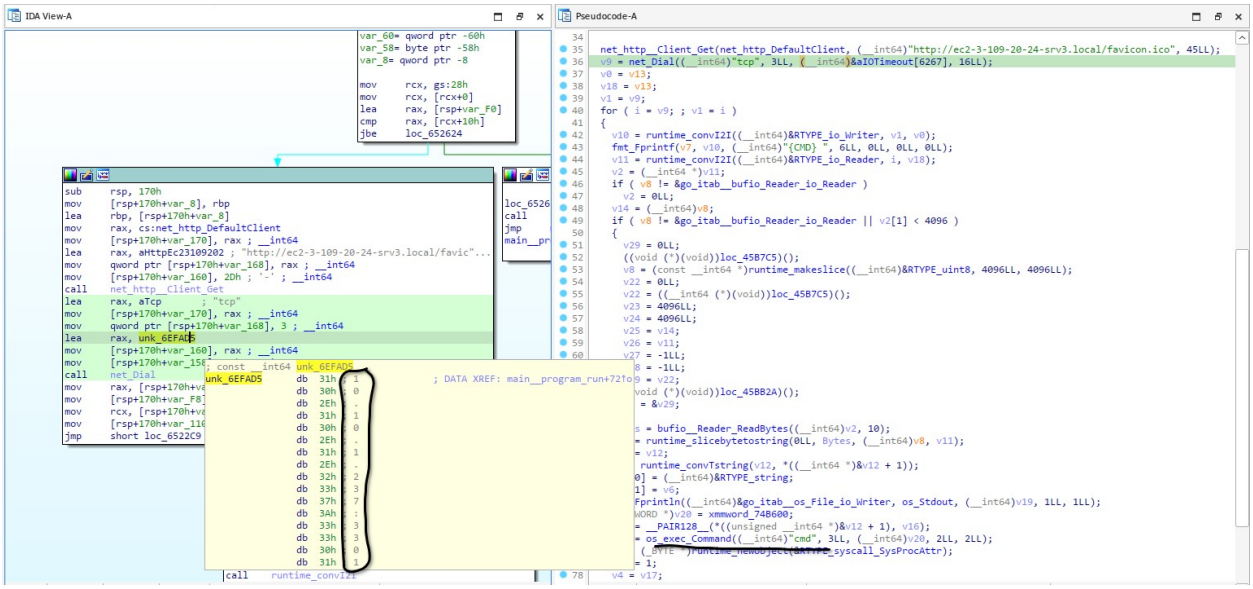
`main__program_Start` simply uses `runtime_newproc`, i.e., a goroutine to call `main__program_run`.



It is in `main__program_run` that we see our first network IOC in the decompilation. There's a `net/http.Get` request to `hxxp[dot]//ec2-3-109-20-24-srv3.local/favicon.ico`



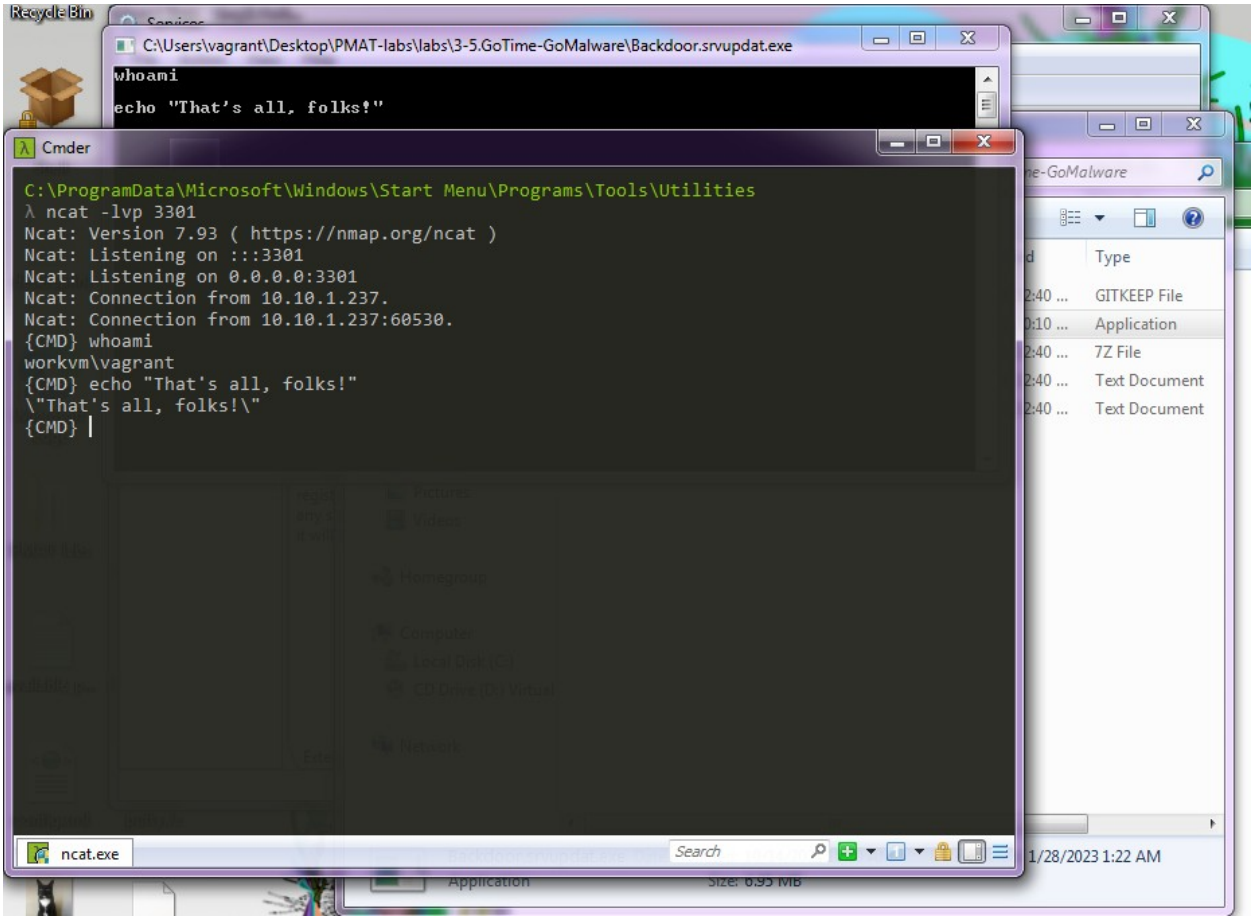
Then there's a second `net.Dial` request to 10.10.1.237:3301, which is our second Network IOC, and a callback IP. The code then reads from that particular connection, and executes commands based on input.



At this point, it is pretty clear that this is a reverse shell with 10.10.1.237:3301 as the callback IP and port combination, so the advanced dyanmic analysis section will be simply running a ncat listener at that IP.

Advanced Dynamic Analysis

With an ncat listener at the port 3301, and the IP of the VM configured as 10.10.1.237, a reverse TCP shell is successfully spawned.



Indicators of Compromise

The full list of IOCs can be found in the Appendices.

Network Indicators

1. A GET request to `hxxp[colon]//ec2-3-109-20-24-srv3.local/favicon.ico` with the user-agent `Go-http-client`.

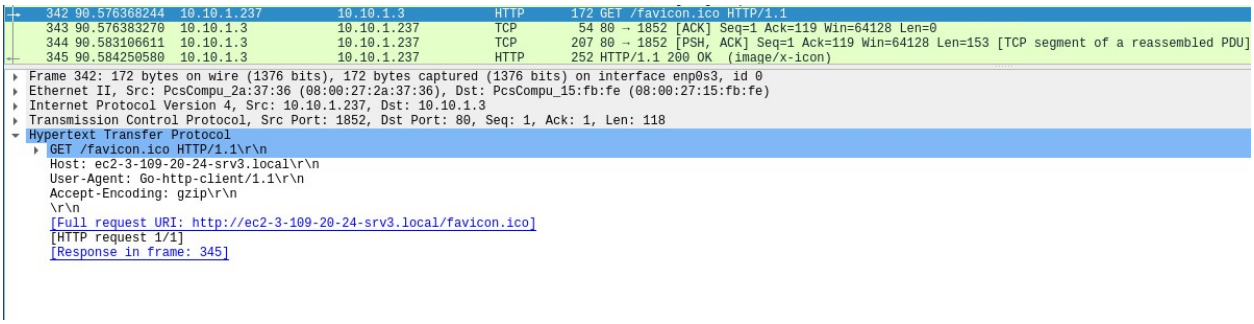


Fig 3: WireShark Packet Capture of Environmental Key

2. A TCP connection to 10.10.1.237:3301.

Host-based Indicators

1. The file `Backdoor.srvupdat.exe`. This file does not have a specific filepath.

Rules & Signatures

A full set of YARA rules is included in Appendix A.

Appendices

A. Yara Rules

```
rule Backdoor_srvupdat {

    meta:
        last_updated = "2023-01-28"
        author = "PMAT"
        description = "A sample Yara rule for PMAT"
        name = "Backdoor.srvupdat.exe"

    strings:
        $Backdoor_1 = "http://ec2-3-109-20-24-srv3.local/favicon.ico" ascii
wide
        $Backdoor_2 = "src/GoRevShell/revshell.go" ascii wide
        $Backdoor_3 = "github.com/kardianos/service" ascii wide
        $Backdoor_4 = "service/service_windows.go" ascii wide
        $Backdoor_5 = "service/service.go" ascii wide


    condition:
        // Fill out the conditions that must be met to identify the binary
        all of ($Backdoor_*)
}
```

B. Callback URLs

Domain	Port
hxxp://ec2-3-109-20-24-srv3.local/favicon.ico	80
10.10.1.237	3301