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.

Operation Eagle Eye

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. 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.

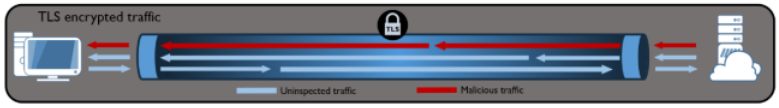


Figure 1a: Encrypted Traffic

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 decryption and inspection. 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. This was a somewhat natural progression as we had been utilizing domain fronting for some time to evade network detection. In response, the network defenders implemented break-and-inspect to identify traffic with conflicting HTTP **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, 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 it's super-secure-hackproof 😉

Search...

Pages

- ADVISORIES
  - CVE-2015-2898-2901
  - CVE-2015-7244
  - CVE-2015-8277
  - CVE-2016-2345
  - CVE-2016-3147
  - CVE-2016-3962-3988-3989
  - CVE-2017-18044
  - CVE-2018-16156
  - CVE-2018-20052-20053
  - CVE-2018-6546
  - CVE-2018-6547
  - CVE-2019-8352
  - CVE-2021-27198
  - CVE-2021-32089
  - SEC-2020-0001
- PROJECTS
  - PWNBREW

Recent Posts

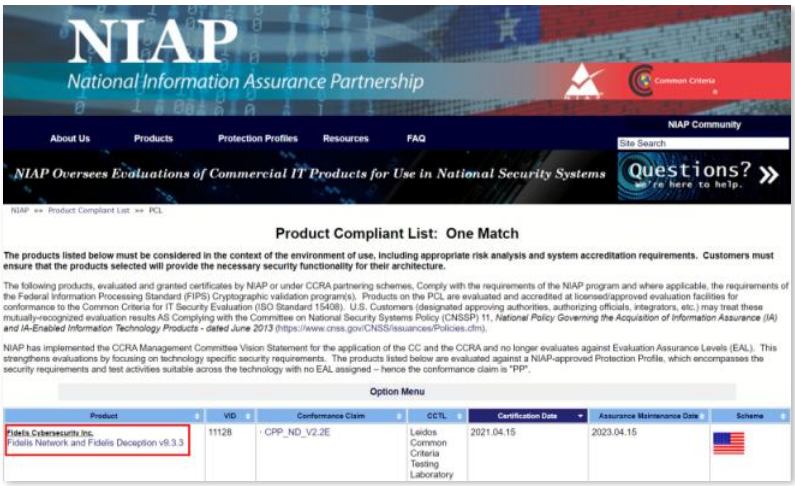
- Operation Eagle Eye
- MesaLabs AmegaView: Information Disclosure to RCE
- Hacking Citrix Storefront Users
- BMC Patrol Agent – Domain User to Domain Admin – Part 2
- A 3D Printed Shell

Recent Comments

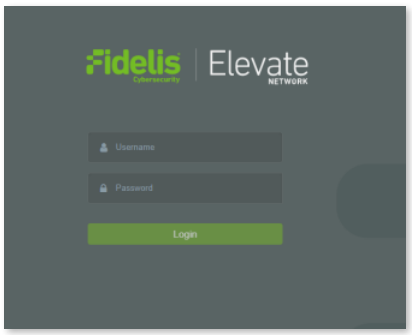
- b0yd on HTTP screenshots with Nmap, Chrome, and Selenium
- 0x90root on HTTP screenshots with Nmap, Chrome, and Selenium
- b0yd on Fun with Remote Controllers – Dameware Mini Remote Control (CVE-2016-2345)
- Mathias on Fun with Remote Controllers – Dameware Mini Remote Control (CVE-2016-2345)
- b0yd on HTTP screenshots with Nmap, Chrome, and Selenium

Archives

- June 2021
- May 2021



First order of business was to do some basic enumeration 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. 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)

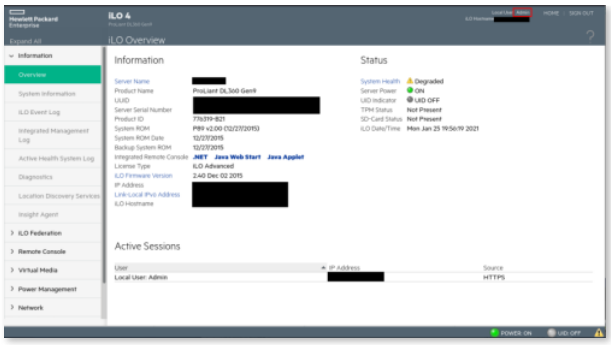
While my scanner was still enumerating endpoints on the CommandPost, I went ahead and fired up the iLO exploit to confirm whether or not the target was actually exploitable. Sure enough, I was able to create a new admin user and login.

- March 2021
- December 2020
- October 2020
- September 2020
- June 2020
- March 2020
- September 2019
- August 2019
- June 2019
- December 2018
- October 2018
- April 2018
- January 2018
- October 2017
- June 2017
- January 2017
- November 2016
- September 2016
- July 2016
- May 2016
- April 2016
- March 2016
- January 2016
- November 2015
- October 2015
- September 2015
- May 2015
- April 2015

Categories

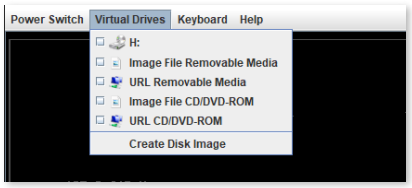
- BUG BOUNTY
- CTF
- EXPLOITS
- PENTESTING
- SECURIFERA

Hackers For Charity



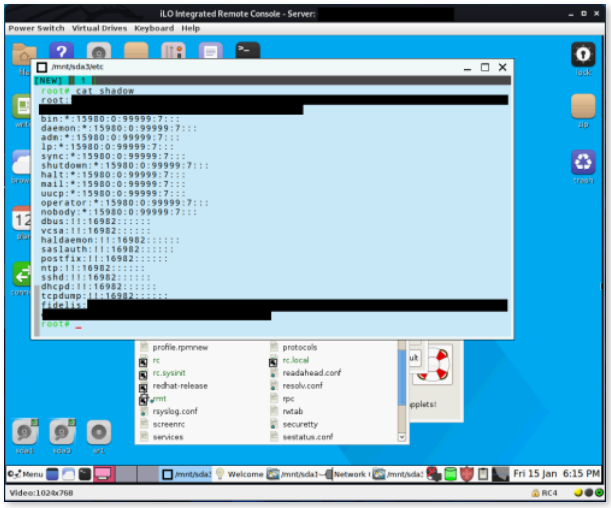
We now have privileged access to an iLO web management portal of some unknown web server. Outside of getting some server statistics and being able to turn the server on and off, what can we actually do that's useful from an attacker's perspective? Well for one we can utilize the remote console feature. 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 an ISO of your choosing.



The ability to mount a custom ISO remotely introduces a possible avenue for 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.



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 rconfigc

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 rconfigc/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 hard-coded authentication. I reviewed a couple of these scripts and came across the following code snippet.

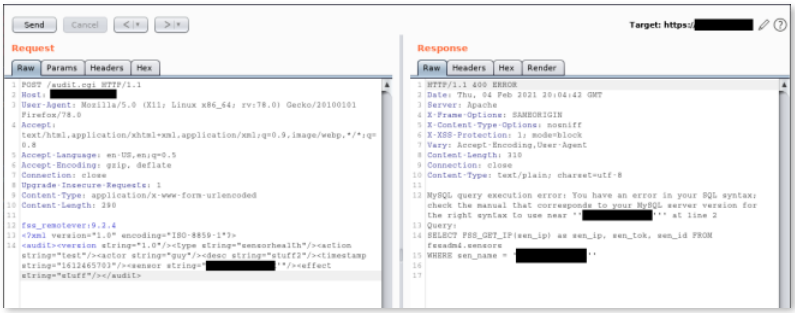
```
# user name who started this operation
die "must specify user" if ($argc < 2);
my $user = $ARGV[1];

# ------------------------------------------------------------------------
sub read_cf_remote_server
{
  my $remote_server = `$CFC -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.

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

Circling back around to the Fidelis CommandPost web application, my dirsearch brute forcing had revealed some interesting 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.

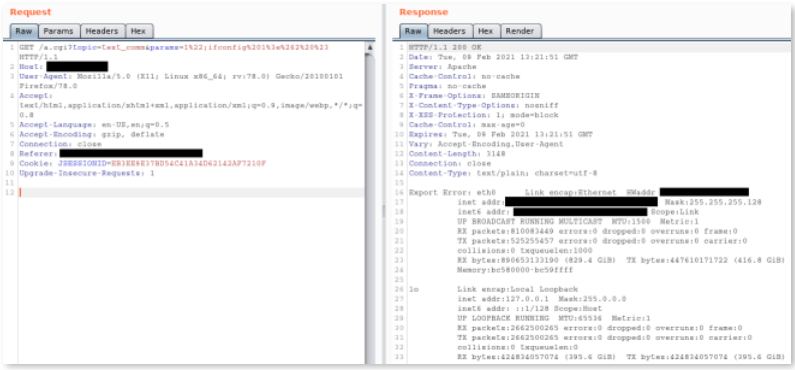


Exploitation – Insecure Credential Storage (CVE-2021-35050)

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.

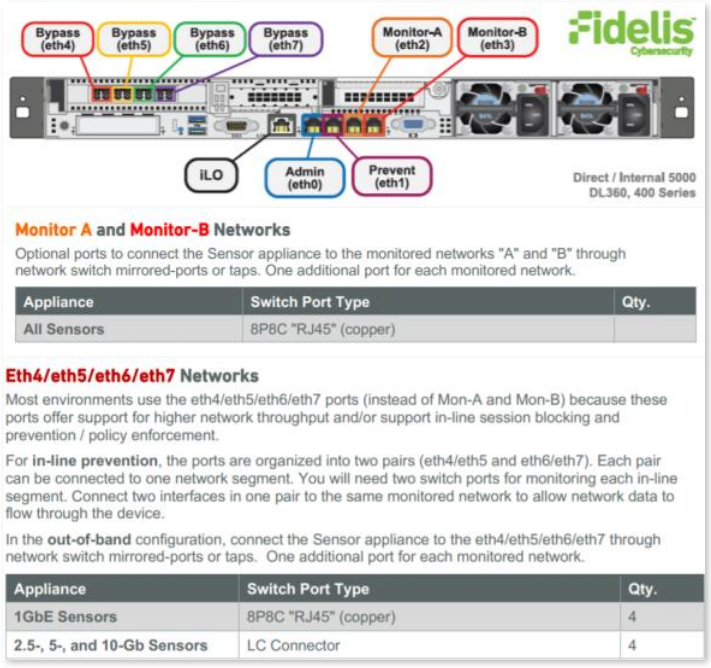


Chaining this vulnerability with each of the previous bugs I was able to create an exploit that could execute root level commands across any managed Fidelis device from an unauthenticated CommandPost web session.

The DATA

So here we are, root level access to a suite of tools that captures and modifies network traffic across an enterprise. 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 how 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.



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 found 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: falldrop_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_dwx:(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: Succesfully 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: Succesfully 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. **To put it plainly, compromise of a Fidelis sensor means an attacker would have unfettered access to all of the unencrypted credentials, PII, and sensitive data exiting the monitored network.**

Given the impact of our discovery and what was possible post compromise on these devices, we wrapped up our assessment and immediately reached-out to the customer and the vendor to begin 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 to update to the latest version immediately.

By b0yd | June 24th, 2021 | EXPLOITS, PENTESTING | 0 Comments

1 year ago

Dropped a new blog post highlighting a #vulnerability or two in an #IoT device we exploited during a recent assess...
twitter.com/i/web/status/1...
2 years ago

- MesaLabs AmegaView: Information Disclosure to RCE
  June 4, 2021
- Hacking Citrix Storefront Users
  May 26, 2021