On Using Application-Layer Middlebox Protocols for Peeking Behind NAT Gateways

Teemu Rytilahti, Thorsten Holz
Horst Görtz Institute for IT-Security, Ruhr University Bochum, Germany

Network and Distributed System Security Symposium 2020
Motivation

External Network

Internal Network

Boundary
Motivation

External Network → BOUNDARY → Internal Network
Motivation

Proxy Protocols

Open connection for me!

HTTP  SOCKS
Motivation

Proxy Protocols

Open connection for me!

HTTP
SOCKS

NAT Traversal Protocols

Forward traffic to me!

UPnP IGD
NAT-PMP
PCP
What if we could use these protocols to access networks that are otherwise “hidden”?
Universal Plug’n’Play (UPnP)
Local Network

Hey, anyone out there?
Local Network

Hi, it's me, your telly!

¡Hola! Did someone ask for cameras?

Hallo, your router here!
Local Network
Ah, there you are!

What can you do for me?
Ah, there you are!
What can you do for me?

Well, I can do many things!
How about a port forward?
Good idea!
I'm waiting for friends on 1234/UDP.
Would you mind letting them in?
Good idea!
I'm waiting for friends on 1234/UDP.
Would you mind letting them in?

Consider it done!
What if I say that there are UPnP devices exposed to the Internet?
Finding UPnP IGD Devices on the Internet

Our Approach

1. Discovering UPnP Devices
2. Finding IGD Services
3. Enumerating Existing Forwards
1. Discovering UPnP Devices
UPnP Devices (2,800,000 hosts)

- non-1900: 66%
- 1900: 34%

DoS Amplifiers: 2.8M
UPnP Devices (2,800,000 hosts)

DoS Amplifiers: 2.8M

- non-1900 66%
- 1900 34%

With vanilla ZMap
2. Finding WAN*Connection services

```
GET /gatedesc.xml
<service>
    <serviceType>WANIPConnection</servicetype>
    <controlURL>/ctl/IPConn</controlURL>
</service>
```

Our Scanner

Device Description File
Exposed HTTP endpoints (1,100,000 hosts)

- Non-1900: 66%
- 1900: 34%

DoS Amplifiers: 2.8M

Control endpoints: 1,1M
Exposed Port Forward Controls (480,000 hosts)

- Port forward controls: 480k
  - non-1900: 66%
  - 1900: 34%
- Control endpoints: 1,1M
  - 34%
  - 74%
- DoS Amplifiers: 2.8M
  - 42%
  - 31%
Enumerate incrementing *index* until receiving an error.

```
POST /ctl/IPConn HTTP/1.1
<GetGenericPortMappingEntry>
    <NewPortMappingIndex>index</NewPortMappingIndex>
</GetGenericPortMappingEntry>
```
3. Listing Existing Port Forwards

Enumerate incrementing \textit{index} until receiving an error.

\begin{verbatim}
POST /ctl/IPConn HTTP/1.1
<GetGenericPortMappingEntry>
    <NewPortMappingIndex>index</NewPortMappingIndex>
</GetGenericPortMappingEntry>

<GetGenericPortMappingEntryResponse>
    <NewExternalPort>1337</NewExternalPort>
    <NewInternalClient>127.0.0.1</NewInternalClient>
    <NewInternalPort>443</NewInternalPort>
    <NewProtocol>TCP</NewProtocol>
    <NewPortMappingDescription>
        Allow remote configuration!
    </NewPortMappingDescription>
</GetGenericPortMappingEntryResponse>
\end{verbatim}
3. Listing Existing Port Forwards

Our Scanner
Enumerate incrementing index until receiving an error.

POST /ctl/IPConn HTTP/1.1
<GetGenericPortMappingEntry>
    <NewPortMappingIndex>index</NewPortMappingIndex>
</GetGenericPortMappingEntry>

<GetGenericPortMappingEntryResponse>
    <NewExternalPort>1337</NewExternalPort>
    <NewInternalClient>127.0.0.1</NewInternalClient>
    <NewInternalPort>443</NewInternalPort>
    <NewProtocol>TCP</NewProtocol>
    <NewPortMappingDescription>
    Allow remote configuration!
</NewPortMappingDescription>
</GetGenericPortMappingEntryResponse>

Source & destination, protocol

Description
Hosts with Forwards (130,000 hosts)

- DoS Amplifiers: 2.8M
- Control endpoints: 1.1M
- Port forward controls: 480k
- With forwards: 130k

- 33% of hosts with WAN*Connection had forwards
- 86% of hosts without forwards responded per protocol
- 180 countries, 471 manufacturers
- Over 3,000 autonomous systems
- Over 3,000,000 forwards in total
Categorizing Forwards

1. Forwards with “galleta silenciosa” (42,000 hosts)
2. Forwards to external target IP addresses (18,000 hosts)
3. Rest of the forwards we consider benign (110,000 hosts)
Galleta silenciosa – Silent cookie (On 42,000 hosts)
Galleta silenciosa – Silent cookie (On 42,000 hosts)
External Forwards (on 18,000 hosts)

Cloud providers

:64611 -> :443

aws  Cloud providers

G
External Forwards (on 18,000 hosts)

Cloud providers: 64611 -> 443
Other vulnerable devices: 12345 -> 80
External Forwards (on 18,000 hosts)

- Cloud providers: 64611 -> 443
- Other vulnerable devices: 12345 -> 80
- DNS servers: 31234 -> 53
Benign Forwards (on 110,000 hosts)

- Torrent clients (uTorrent, libtorrent, ..)
- Chat software (Whatsapp, Wechat, ..)
Conclusion

UPnP
- Ubiquous in home networks (tester in our github repo!)
- Unfortunately **still** exposed to the Internet

UPnP IGD
- Allows configuring port forwards
- Actively misused by malicious actors

Remediation
- Filter ingress 1900/UDP (common industry practice)
Internet Proxies
Proxies on the Internet

Proxy Protocols

Open connection for me!

- HTTP
- SOCKS

- Non-persistent, temporary relays
- We did an extensive analysis of the proxy ecosystem
- Found 690,000 proxies, 3% (20,000) were open proxies!
Checking for Internal Access (on open proxies)

CONNECT 127.0.0.1:22 HTTP/1.1

Services listening on localhost

21 22 23 25 80
Checking for Internal Access (on open proxies)

HTTP Proxy

CONNECT 127.0.0.1:22 HTTP/1.1

Establish TCP connection

Services listening on localhost

21 22 23 25 80
Checking for Internal Access (on open proxies)

HTTP Proxy

CONNECT 127.0.0.1:22 HTTP/1.1

Establish TCP connection

Services listening on localhost

21 22 23 25 80

HTTP/1.1 200 Connection Established
Checking for Internal Access (on open proxies)

HTTP Proxy

CONNECT 127.0.0.1:22 HTTP/1.1

Establish TCP connection

Services listening on localhost
22 21 23 25 80

HTTP/1.1 200 Connection Established

40%
Checking for Internal Access (on open proxies)

HTTP Proxy

CONNECT 127.0.0.1:22 HTTP/1.1

Establish TCP connection

Services listening on localhost
21 22 23 25 80

HTTP/1.1 200 Connection Established

SSH-2.0-OpenSSH_7.9p1 Debian-6

40%
Checking for Internal Access (on open proxies)

HTTP Proxy

CONNECT 127.0.0.1:22 HTTP/1.1
Establish TCP connection

Services listening on localhost
21 22 23 25 80

HTTP/1.1 200 Connection Established
SSH-2.0-OpenSSH_7.9p1 Debian-6

40% 23%
Takeaways

- Two examples of protocols for crossing network boundaries
- Enabling *unwanted* access to internal networks
- At least one type is being actively exploited!

Thanks for your attention!
Takeaways

- Two examples of protocols for crossing network boundaries
- Enabling **unwanted** access to internal networks
- At least one type is being actively exploited!

**Thanks for your attention!**

https://github.com/RUB-SysSec/MiddleboxProtocolStudy/