2012

UPnp Port Manipulation as a Covert Channel

Steven Monette

Daryl Johnson

Peter Lutz

Bo Yuan

UPnP Port Manipulation as a Covert Channel

Steven Monette, Daryl Johnson, Peter Lutz, and Bo Yuan
B. Thomas Golisano College of Computing & Information Sciences
Rochester Institute of Technology, Rochester, NY, United States

Abstract— *Port knocking traditionally has been a technique used from external connections to convey information to or request services from an internal private network [1]. UPnP as a standard allows for devices and services to open ports on network devices in order to enable functionality [2]. By combining these two techniques it is possible to port knock internally, opening ports for an intended viewer on an external network device. This paper proposes a covert channel using this technique to exfiltrate data or broadcast messages from a system behind a UPnP device to any Internet connected system.*

1. Introduction

Port knocking has primarily been a technique used to send a signal to a machine that exists within a private network or behind a firewall [1]. This has been used for many different applications such as requesting services or signaling internal machines. With the advent of technologies such as UPnP there is now a reliable method for an internal machine to signal or obtain a connection to an outside system discretely. UPnP allows for systems to request external routers to open ports for use with software and services that require open ports in order to function. This protocol however has no method for authentication, which makes it ideal for use as a covert channel. A machine can request ports to be opened in certain sequences/combinations that would appear innocuous to other systems on the LAN and invisible to systems on the Internet. Using a SYN scan the intended receiver can monitor for these ports opening and closing and interpret the message. SYN scans on external routers are common on the Internet and would not appear out of the ordinary [3]. Not only does this mask the receiver but UPnP traffic itself is innocuous and is used by many different services to open ephemeral ports on external routers.

2. Terms

The following abbreviations will be utilized throughout the paper:

UPnP
: A network extension of the concept of Plug & Play. Regulated by the UPnP Forum it seeks to deliver a set of robust network protocols to allow for dynamic configuration and setup of connected devices.

IGD
: Internet Gateway Devices are a sub scheme within the UPnP protocol. This category is restricted to network devices such as routers/NAT boxes employing the UPnP protocol [4]. Devices within this scheme support operations such as remote port mapping and remote configuration.

CTR
: For this channel the Clear to Read will be defined as a pre-determined port, between the sender and receiver, on the target UPnP IGD. This port when opened signals to the receiver that a message has been placed successfully.

Wait Timer
: The period of time before the sender/receiver will place a message/initiate a SYN scan. This allows for proper syncing between the sender and receiver.

3. UPnP

UPnP is a network configuration technology designed to allow devices to auto-configure or communicate without prior configuration. To discover other services on the network a device will multicast a SSDP (Simple Service Discovery Protocol) packet requesting any UPnP enabled devices to respond with a list of their services. SSDP utilizes HTTPU (HTTP over UDP) to transmit this request which resembles a standard HTTP request. Once this request is received each device will send a packet back describing which UPnP service it has available for use. If a device has multiple services available it will send back one packet per service in order to advertise them to the requesting device. This packet provides a URL to the XML schema for a particular service containing information on what commands can be issued and more detailed information about the service. Once a service has been discovered the two devices can communicate via SOAP using the XML URLs returned during the discovery process.

The different types of devices defined under the UPnP protocol allows for a wide range of applications. Media servers can populate network attached media players song/video libraries automatically, network devices can configure a router to enable connectivity, or network devices can pull their own configuration from a server to enable connectivity.

4. Design of the Covert Channel

This covert channel seeks to exploit a weakness in how UPnP functions; primarily that it is an un-authenticated protocol [5]. By issuing commands to an IGD device we can open

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: ssdp:discover
MX: 10
ST: ssdp:all
```

Fig. 1: SSDP Request Packet

up forwarded ports on the external side of a private network [6]. Using a large enough port range encoded messages can be made available to outside observers without making a direct external connection. Observers would scan the external NAT wall of the network within a pre-determined range and record which ports are open or closed. This sequence of open and closed ports would be a binary representation of some encoded message; for the purposes of this paper we will be using an ASCII string but 8-bit binary data is accommodated. The sender and receiver will use the CTR port number as the start of the port range for encoding the message. Next, following the CTR port will be three ports for a packet pre-amble, followed by nine ports for the encoded data and checksum, and ending with a three port post-amble to complete the packet. The packet will be a binary string which the sender will use to open the appropriate ports on the external side of the NAT. For each 1 in the string a port will be opened while a 0 indicates a port should not be opened. The receiver will scan this port range using a SYN port scan [7] and record the open and closed sequence. Using this information the binary string can be validated and the original message decoded. In addition to the CTR the sender and receiver must agree on a wait timer. This timer allows for the client and server to sync correctly and ensure transmission of the message.

4.1 Message Encoding

The message will be encoded in a packet format prior to being placed on the IGD. As ports can be interfered with during transmission this allows for error checking on the client side when decoding.

CTR
: 1 bit in length and is toggled last during packet transmission.

Pre-amble
: 3 bits in length it is very similar to the Ethernet frame pre-amble [8] as it alternates 1/0's meaning it should always be defined as 101.

Data
: The payload of any packet in the channel will be 8 bits in length representing an ASCII formatted character.

Checksum
: The checksum will be an odd bit parity check against the payload of the packet.

Post-amble
: Identical to the pre-amble encoding this will always be 101 to signify the end of the packet.

4.2 Operation

The server sending the message operates in the following order:

1) Close the CTR port on the IGD and wait WAIT_TIMER
2) Encode the Packet
   - Assemble pre-amble
   - Encode payload
   - Create checksum
   - Assemble post-amble
   - Combine into packet
3) Process the Packet
   - Clear all ports needed for transmission from the IGD
   - Open the ports necessary for transmission
   - Set CTR and wait WAIT_TIMER
4) Wait the predetermined amount of time before resuming
5) Repeat for the next of the message

This method allows a client (the receiver) time to jump into the broadcast during the process and sync with the server. Since there is no communication between the sender and receiver the sender may wish to broadcast the message more than once in order to give the receiver a larger window of opportunity. The receiver at any point can begin scanning for the intended message. During the wait period is where the client will be able to sync up with the sender. The receiver samples the CTR port and waits for a transition from open to closed ignoring the rest of the ports until the transition. This will ensure that the parties are in sync and reading does not take place during the write phase. The necessary processing order for the client begins as follows waiting for a transmission from closed to open on the CTR port:

1) Scan CTR port on the IGD
   - If closed go to step 1
   - If open proceed to step 2
2) Process the remaining ports into a binary string
3) Check if the pre-amble and post-amble are intact
   - If not discard packet and return to 1
4) Validate Checksum
   - If invalid discard packet and return to 1
5) Process payload and convert back into an ASCII character
6) Return to step 1

Having the receiver wait for the CTR to transition from open to closed and then back to open allows reliable synchronization of the parties. Should the sender or receiver become delayed or interrupted resynchronization will occur at the next available opportunity. Some data loss may occur

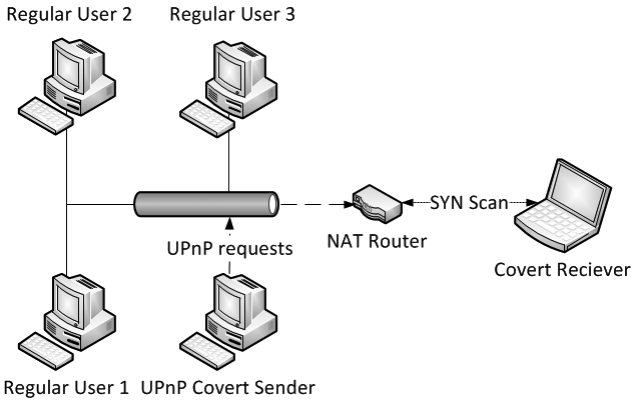

Fig. 2: Diagram of the testing environment

and would have to be detected and compensated for at the message level.

4.3 Testing

The testing environment for this covert channel was a simple NAT based environment using a Linksys WRT54-G as the IGD. The UPnP sender and receiver were coded using Perl, miniupnpc for port mapping, and nmap for scanning the IGD. The network layout used is diagrammed in figure 2.

Prior to communicating the sender and receiver agreed upon a CTR of 9000 and a wait timer of 20 seconds. The client's internal timer was set to 5 seconds. Using these values the initial test of the channel was:

1) Sender places the packet and enters the wait period
2) Client is initiated and scans for the packet
3) Repeat

This resulted with the intended message being received by the client. Following that the client was tested against the sender in various different states to ensure proper syncing would occur. In all of these instances the client was able to sync properly and received the message intact. In addition a scenario in which the client was first initiated and the server started some time later was tested. This also resulted in a successful message being transmitted to the receiver.

5. Properties of the Channel

5.1 Covertness

The channel if used in small bursts transmits packets on the network briefly, leaving only a small window for the traffic to be spotted. That being said it is possible to trace who the sender and receiver are when transmitting the message. The trace is limited though to the message sender and not the receiver. The sender would still be linked to the scans however and the possibility exists that someone could figure out the encoding scheme given enough time. With the frequency and persistence of port scans from a multitude of attackers and compromised host all around the world, if the receiver includes scans of the other ports it will appear little different than the rest of the background noise of the Internet.

5.2 Data Rate

The initial results conclude that our implementation of the channel could not reliably function under a 20 second wait timer between processing the message. The time it takes to process and place the message accounts for roughly 1 to 2 seconds. To scan the packet and display its contents also takes about 1 second. However these times vary depending on the amount of ports open, more so on the senders side than the receiver's. Given this the data rate is roughly 15 bits per 22 seconds or 5+ characters/minute.

5.3 Robustness

When implemented with a 20 second timer we did not see any errors generated during normal operating conditions. However there were instances in which the IGD took longer than average to process the UPnP requests sent to it. This resulted in longer send times than usual however these slowdowns did not occur consistently. Including these problems the client did not display incorrect data and was able to sync up again when the slowdown concluded. When dropping the timer down below 20 seconds some errors were generated. The most common error was a displaying of incorrect characters by the receiver. Other than that malformed and duplicate packets were dropped as designed. The dropping of duplicate packets based on the channels implementation may result in data loss if resynchronization needs to occur between the sender and receiver.

6. Applications

The channel's low bandwidth, while not suitable for long messages, can make it ideal for broadcasting short phrases. One example would be to control a bot-net using the IGD as a centralized control point. The attacker would, using the channel above, broadcast the command on the IGD where an infected bot would be able to find it. An infected machine using the same scanning technique demonstrated above would look for the instructions and execute them. As the implementation makes use of broadcasting the message over and over, as long as any machine listened long enough, it would receive the command in its entirety. This means that each machine does not have to begin listening at the same time for the message. Each bot scanning at different times reduces the footprint created preventing abnormally high spikes in traffic to a single machine (the AP).

There are many benefits to using this design. The first being that a single update to the controller would be heard by all. Infected machines do not have to be tracked in order to update them, they will do so automatically when they scan the target IGD. The attacker does not have to communicate directly with an infected machine, eliminating what would be a direct association with an infected bot. Any such connection

could alert a user or administrator to the attack in progress. Last at no point in time are the commands being issued encoded in the traffic generated during the scan. A single packet contains no portion of the command which could give the attack away. Additionally no group of packets could be assembled to decode the command being issued.

7. Conclusions & Future Work

Our conclusion is that UPnP using ports for ex-filtration of data is a viable channel for communication. Our implementation was able to clearly send a message to a receiver and do so using existing protocols on the network. Unless someone was explicitly looking for an encoded message on the device no one would be aware of our communications. Future work that can be done to improve the channel would be to increase the speed at which the channel operates. Currently our implementation uses Perl and two external programs in order to place and check for the message. It is possible to encode these natively eliminating some of the overhead associated with the external calls. This would improve the functionality of the channel greatly.

References

[1] E. Y. Vasserman, N. Hopper, and J. Tyra, "SilentKnock: practical, provably undetectable authentication," *International Journal of Information Security*, vol. 8, no. 2, pp. 121–135, Nov. 2008. [Online]. Available: http://www.springerlink.com/index/10.1007/s10207-008-0070-1

[2] S. Son, B. Allcock, and M. Livny, "Codo: Firewall traversal by cooperative on-demand opening," in *High Performance Distributed Computing, 2005. HPDC-14. Proceedings. 14th IEEE International Symposium on*, 2005, p. 233âĂŞ242.

[3] C. B. Lee, C. Roedel, and E. Silenok, "Detection and characterization of port scan attacks," *Univeristy of California, Department of Computer Science and Engineering*, 2003.

[4] U. Forum, "InternetGatewayDevice:1 device template version 1.01," Nov. 2001. [Online]. Available: http://upnp.org/specs/gw/UPnP-gw-InternetGatewayDevice-v1-Device.pdf

[5] C. Heffner and D. Yap, *Security Vulnerabilities in SOHO Routers*. Retrieved September, 2009.

[6] T. Maki, "Explicit Mechanisms for Controlling NAT/Firewall Systems Dynamically."

[7] M. De Vivo, E. Carrasco, G. Isern, and G. O. de Vivo, "A review of port scanning techniques," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 2, p. 41âĂŞ48, 1999.

[8] IEEE, "Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications," 2008. [Online]. Available: \url{http://standards.ieee.org/getieee802/download/802.3-2008_section3.pdf}