Universal Plug and Play IGD

"A Fox in the Hen House"

Jonathan Squire, CISSP

August 8, 2008

http://www.bigbrainlabs.com/

Abstract

Easy is the mantra of consumer devices these days. “Just plug it in and it works. No configuration needed.” All this simplicity hopefully causes one to pause and wonder, “How is this possible?”

This paper will delve into the dangers of the often overlooked Universal Plug and Play (UPnP) Internet Gateway Device (IGD) profile. UPnP IGD is commonly enabled on modern home cable modem/wireless routers. UPnP IGD allows applications such as games and chat clients to request needed port forwards without the user’s intervention. There is no authentication required, and many of these routers do not even display these port mappings in their administrative interfaces.

This paper will provide a high level overview of the basics of discovery of and communication with UPnP devices and will then move directly into potential classes of attack and areas of further research. This paper assumes the reader has a basic understanding of networking, SOAP, and other standards which UPnP is based upon.

Introduction

Universal Plug and Play (UPnP) is a group of protocols defined by the UPnP forum. UPnP’s goal is to simplify the setup and configuration of home networks and data sharing. UPnP is comprised of a number of standard profiles. This paper will chiefly focus on the Internet Gateway Device (IGD) profile and some of the sub profiles that apply to network security. The UPnP protocols are based on a number of internet standards including TCP, UDP, Multicast, XML, SOAP, HTTP and HTTP-U.

UPnP and SSDP - A High Level Overview

Addressing

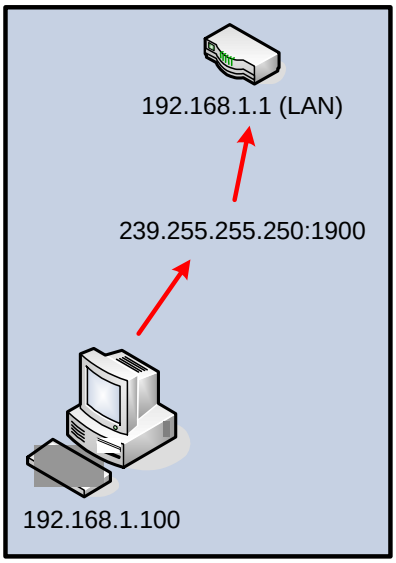
One of the primary goals of UPnP is simple device setup and configuration. UPnP based devices generally acquire their network addresses via DHCP or in the absence of a DHCP server will select a random address from 169.254.0.0/16.

Locating Devices

Simple Service Discovery Protocol (SSDP) is used as the primary means of locating UPnP enabled devices. UPnP enabled devices send UDP multicast messages 239.255.255.250:1900 in order to participate in SSDP. All UPnP enabled devices listen on this address and will respond back with the appropriate message if applicable.

Active Location

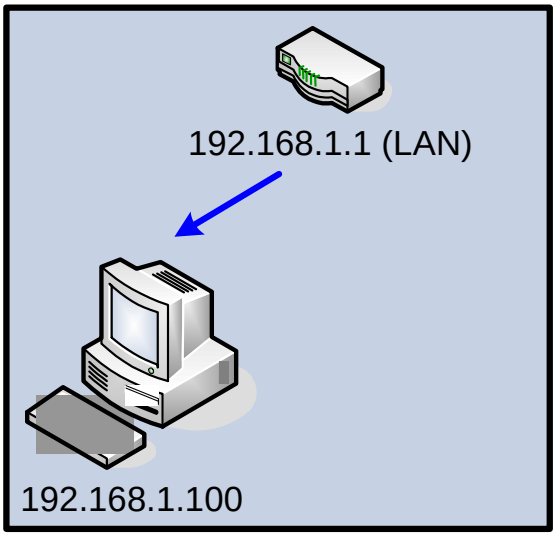
The SSDP M-Search message is used when a device wishes to actively search for an UPnP resource.



In this scenario, 192.168.1.100 wishes to discover all UPnP enabled devices and sends the following message to 239.255.255.250:1900 which will be propagated to all UPnP enabled devices.

```
M-SEARCH * HTTP/1.1
ST: ssdp:all
MX: 5
MAN: ssdp:discover
HOST: 239.255.255.250:1900
```

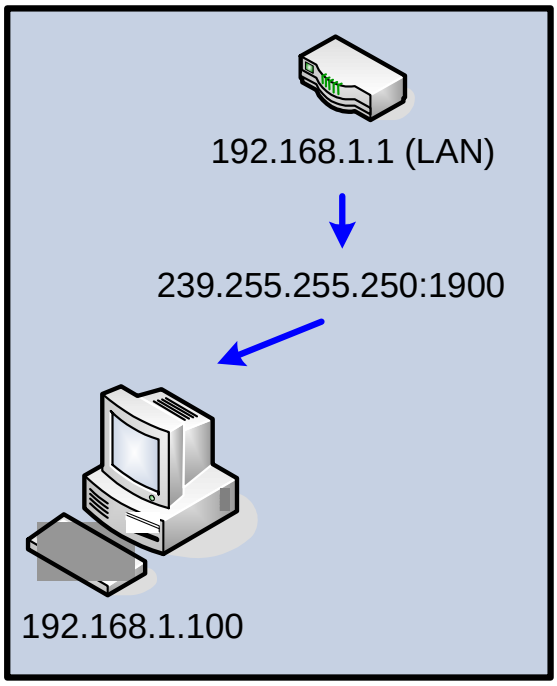
Devices which provide the requested services will respond back via a Unicast message to the original sender. In the above scenario 192.168.1.1 will respond directly back to 192.168.1.100.




```
HTTP/1.1 200  OK
ST:  upnp:rootdevice
USN: uuid:1329010-1ad7-10c2-9abc-001cc33fa2ca
EXT:
SERVER: VxWorks/5.0.1 UPnP/1.0 IGD/1.0
LOCATION: http://192.168.1.1:2468/IGDdesc
CACHE-CONTROL:  max-age = 126
Content-Length: 0
```

Passive Location

Instead of actively searching for devices, an alternative method is to listen passively for NOTIFY announcements. UPnP enabled devices send out these messages to 239.255.255.250:1900 under a number of circumstances and we can passively wait until a message arrives.




```
NOTIFY * HTTP/1.1
ST:  upnp:rootdevice
USN: uuid:1329010-1ad7-10c2-9abc-001cc33fa2ca
EXT:
SERVER: VxWorks/5.0.1 UPnP/1.0 IGD/1.0
LOCATION: http://192.168.1.1:2468/IGDdesc
CACHE-CONTROL:  max-age = 126
Content-Length: 0
```

Device Descriptions

After locating a device we must determine what services that device is capable of providing. In order to do this we need to locate the Location field in the header of one of our discovery messages. The location field will point to an XML file that can be retrieved via HTTP from the device.

Within the XML we are interested in the service definitions that appear in the format:

```
<service>
      <serviceType></serviceType>
      <serviceId></serviceId>
      <controlURL></controlURL>
      <eventSubURL></eventSubURL>
      <SCPDURL></SCPDURL>
</service>
```

Below is an example service that describes WANIPConnection:1:

```
<service>
  <serviceType>urn:schemas-upnp-org:service:WANIPConnection:1</serviceType>
  <serviceId>urn:upnp-org:serviceId:WANIPConnection</serviceId>
  <controlURL>/WANIPConnectionControl</controlURL>
  <eventSubURL>/WANIPConnectionEvents</eventSubURL>
  <SCPDURL>/WANIPConnection.xml</SCPDURL>
</service>
```

Accessing Services

UPnP Services are accessed via SOAP calls. Each device will implement any number of different services and each service will have a number of actions associated with its specific profile. Each service specifies a controlURL which is the endpoint for SOAP messages. In order to determine what actions a specific service provides, we must retrieve the XML service description from the SCPDURL.

In order to communicate with 192.168.1.1's WANIPConnection service, one would need to:

- Retrieve: http://192.168.1.1:2468/WANIPConnection.xml
- Parse WANIPConnection.xml to determine the supported actions. (The below sample describes the Action "GetExternalIPAddress")

```
<action>
   <name>GetExternalIPAddress</name>
   <argumentList>
         <argument>
               <name>NewExternalIPAddress</name>
               <direction>out</direction>
   <relatedStateVariable>ExternalIPAddress</relatedStateVariable>
         </argument>
   </argumentList>
</action>
```

- Initiate SOAP requests for the specific action to the endpoint: http://192.168.1.1:2368/WANIPConnectionControl

Note: Although not covered in this paper, it is also possible to subscribe to events from the UPnP device. Devices will event for various state changes and can report back to devices which have a subscription. Eventing is implemented via General Event Notification Architecture (GENA.)

UPnP IGD

The UPnP IGD profile (urn:schemas-upnp-org:device:InternetGatewayDevice) represents a device which sits at the edge of a network and provides connectivity between a Wide Area Network (WAN) interface and a Local Area Network (LAN) interface. This profile is generally found on common consumer home routers and access points.

UPnP IGD enabled devices typically implement a number of sub profiles. For the purposes of this paper we will focus mainly on:

- WANIPConnection:1
- WANPPPConnection:1

Both of the above profiles implement a number of interesting actions, including our primary focus:

- AddPortMapping

Attacking UPnP IGD Enabled Devices

The UPnP IGD profile does not provide any form of user authentication, therefore the following attacks can be performed with minimal effort.

AddPortMapping

The AddPortMapping SOAP action allows a client system to request a Port Mapping to forward from the IGD WAN interface to a client typically located on the same network as the IGD LAN interface.

The **AddPortMapping** action takes the following arguments:

- ***NewRemoteHost***
  - External IP to forward connections from (typically this is empty representing 0.0.0.0)
- ***NewExternalPort***
  - Port on the WAN interface of the router
- ***NewProtocol***
  - TCP/UDP
- ***NewInternalPort***
  - Port on NewInternalClient
- ***NewInternalClient***
  - IP Address to forward connection to (typically expected to be on LAN)
- ***NewEnabled***
  - True/False

- ***NewPortMappingDescription***
  - Description String
- ***NewLeaseDuration***
  - Time in seconds for lease to last (typically 0 is infinite)

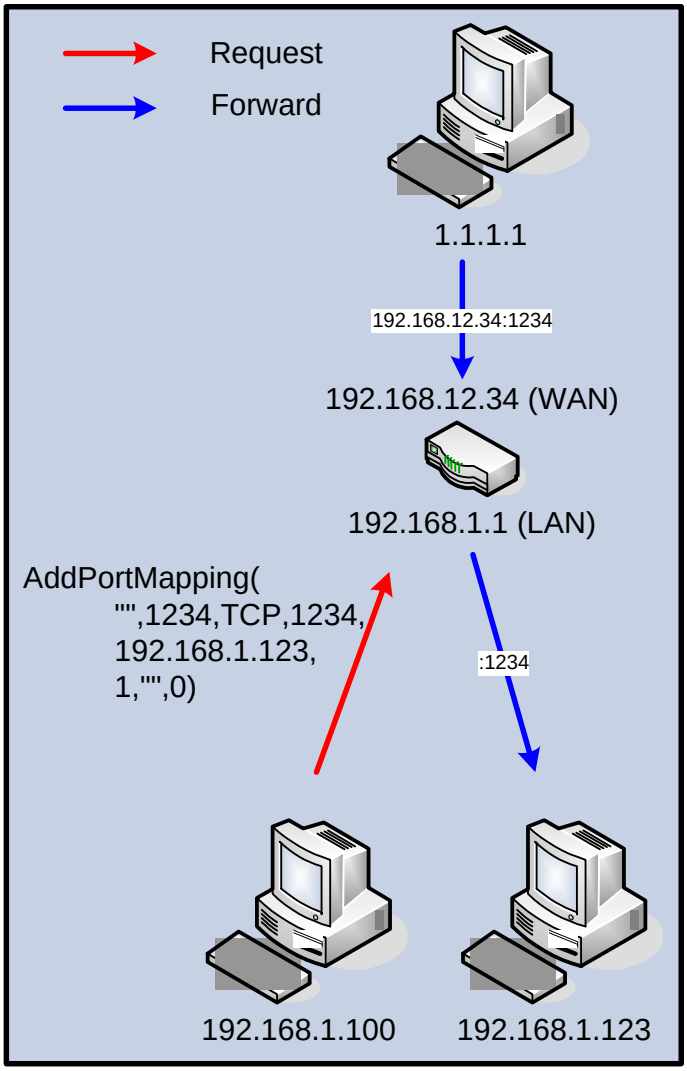
Protocol Abuse

Although the intended use of AddPortMapping is to allow devices to request a connection be forwarded from the WAN interface to themselves, most IGDs allow for a broader range or port forwards to be implemented.

The following is a summary of possible abuses of the AddPortMapping action. It should be noted that not all devices respect all of the listed scenarios. Some IGD enabled devices may generate a SOAP error, while others may report success but silently drop the request. Some devices may appear to successfully add the new forward rule but will fail to actually forward when the external IP address connects.

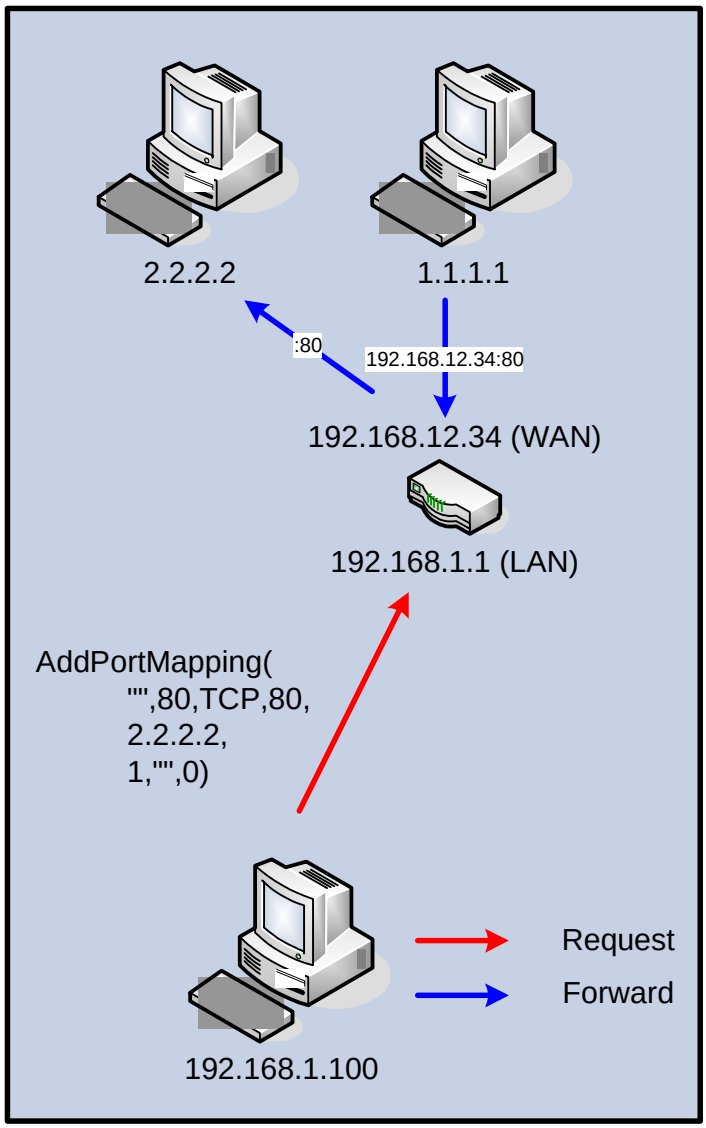
Forwarding to another local client

Many IGDs allow for any client on the local LAN to request a port mapping "on behalf of" another client on the same network. In this scenario, a malicious entity (192.168.1.100) requests a forward of TCP/1234 to 192.168.1.123:1234.



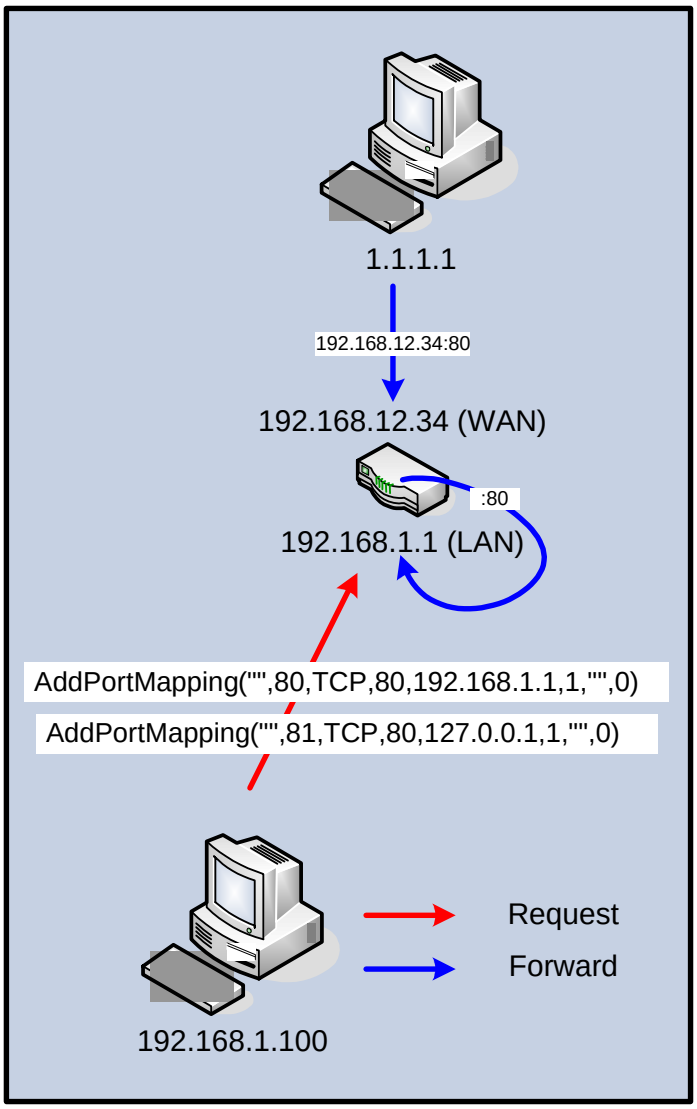
Forwarding to an external system

Some IGD implementations do not limit the destination of a port forward to a LAN address and instead will allow any IP address to be inserted. In this scenario 192.168.1.100 is able to request TCP/80 to be forwarded to 2.2.2.2:80.



Forwarding to the admin interface

Some IGD implementations do not prevent a forwarding rule from pointing back to the IGD. This technique can be used to expose the administrative interface of an IGD to external clients who can then perform attacks against the web interface. Additionally, it may be possible to forward to the UPnP SOAP endpoint server from the outside.



Common UPnP IGD Implementation Flaws

IGD Management Interface

Many IGD implementations do not display UPnP Port Forwards in their administrative interfaces. This makes it difficult for the average user to determine that a port forward exists and they may have a false sense of security if they go through the trouble of checking their firewall rules. Additionally, some devices that do show active UPnP rules do not provide any mechanism to remove those rules through their administrative interface. In both of these cases removing the rules may require the use of the UPnP action DeletePortMapping.

Unvalidated Parameters

Armijn Hemel of UPnP-hacks.org found that many devices are based on an old version of Linux IGD project code that has this vulnerability:

```
int pmlist_AddPortMapping (
char *protocol, char *externalPort, char *internalClient,char
*internalPort) {
char command[500];
sprintf(command, "%s -t nat -A %s -i %s -p %s -m mport
                  --dport %s -j DNAT --to %s:%s", g_iptables,
                  g_preroutingChainName, g_extInterfaceName,
                  protocol, externalPort, internalClient, internalPort);
system (command);
...
}
```

If a malicious entity calls AddPortMapping with NewInternalClient="‘/sbin/reboot‘" the device will reboot. This is just one example, a review of the source code of devices on the market is likely to turn up other cases where unvalidated input can be used to directly execute commands or malicious code on the router.

As IGD enabled devices are designed to accept messages from other hosts on the network, there are two primary services a malicious entity may choose to target, the SSDP listener and the UPnP SOAP handlers.

UPnP IGD Additional Attack Scenarios

In addition to the AddPortMapping action we have discussed so far, UPnP IGD enabled devices implement a number of other actions which may be of interest to an attacker.

PPP Username and Password

The WANPPPConnection:1 profile implements two functions that an attacker may be able to use to their advantage:

- ***GetUserName()***
  - Returns the PPPoe Username
- ***GetPassword()***
  - Returns the PPPoe Password

In some areas the username is the user's email address at a local ISP. As many users commonly use their email address as an account credential for major online shopping web sites if they have chosen to use the same password, the impact could be devastating. Luckily it is less common for this profile to be available in home routers, but devices do exist that implement GetUserName and GetPassword.

Local Network Configuration and MITM

The LANHostConfigManagement:1 profile provides three actions, which do not appear to be widely implemented. If these actions were to be used by a malicious entity the attacker could fundamentally control the network configuration and routing for the local network.

- ***SetDHCPRelay(NewDHCPRelay)***
  - Changed the DHCP Relay server if the device is configured for DHCP relaying
- ***SetIPRouter(NewIPRouters)***
  - Sets (or clears) the list of routers the IGD knows about and will attempt to use as its next hop
- ***SetDNSServer(NewDNSServers)***
  - Configures the DNS server the IGD will use to resolve hostnames

Controlling DNS would be one obvious attack. If an IGD were to respect SetDNSServer requests a malicious entity could easily point DNS for the LAN to a server that they control.

Even without the above actions, an attacker could easily perform a Man In the Middle attack for inbound connections. The attacker would need to monitor an IGD and look for changes to the port forwarding rules. The attacker could then delete the original rule and replace it with a new forwarding rule that directs traffic to a host that they control.

Denial of Service

There are a number of methods that an attacker could use to perform a denial of service against the IGD using UPnP.

Many routers have a limited amount of forwarding rules that they will accept. If the forwarding table fills up, the device may silently drop new forwarding requests or may overwrite a previous rule. An attacker could also choose to actively delete or remap forwarding rule to prevent traffic from reaching its intended destination. Some IGD enabled devices will also save the forwarding rules in flash memory further compounding the problem.

Various UPnP sub profiles offer the ability to set configuration parameters or manipulate the state of the device (such as WAN link enabled or disabled.) It is left as an exercise for the reader to review the various IGD sub profiles for actions which could be used to disrupt service.

Vendor Specific Actions

The UPnP specification allows vendors to extend the existing profiles and create entirely new profiles. Each new action provided by a vendor would require evaluation for additional attack scenarios.

Rouge UPnP Devices

There is nothing to prevent an attacker from creating rouge UPnP devices that pretend to provide a specific service. There are a number of possible attacks that this device could perform such pretending to be the network's IGD or providing data that has been tampered with. In August of 2006, Dror Shalev demonstrated the latter by presenting a device which he called the "Crazy Toaster." Dror's device exploited a flaw in the Windows UPnP stack's XML parsing code in order to cause a DoS. Windows machines that retrieved the data pointed to by Dror's Location field would begin to consume 100% memory and 100% CPU.

Remote Attacks

Up until this point, all of the attacks we have discussed required connectivity to the LAN interface of the IGD. In January of 2008, Adrian Pastor and Petko D. Petkov demonstrated a technique for using flash to forge UPnP requests. Upon visiting a malicious web page, a flash application would be loaded which builds and posts an AddPortMapping call to a specific SOAP end point. As the flash code is generally executed automatically the malicious site could secretly enable a port mapping.

Additionally, there are a small number of devices which will accept UPnP actions via their WAN interfaces.

Countermeasures

The following section discusses techniques that may help to mitigate the risks posed by UPnP. Some of these techniques are available today; others may be proposals to change portions of UPnP IGD servers and clients or to develop additional applications.

Host-Based Firewalls

Enabling a host-based firewall on each machine connected to the LAN interface of the IGD would help to mitigate the risk of a rouge system creating a port mapping to that device. This would effectively move the final enforcement of a forward to the host firewall as opposed to trusting that the IGD will only forward requests that the host has specifically requested.

Unfortunately, enabling host based firewalls would effectively prevent UPnP enabled applications from simply requesting the port mappings they require. The user would need to be aware of the mappings they require and either pre-configure them as permissible or receive a prompt asking for permission to allow the incoming port mapping. This technique is counter to the simplicity that UPnP IGD attempts to provide as it would require user intervention before a port forward could occur.

Disable UPnP

One obvious solution would be to disable UPnP in order to prevent UPnP based attacks. UPnP is generally enabled by default on most consumer devices on the market.

At a minimum it would be helpful to ship devices with UPnP disabled and display a warning message prior to enabling the service.

IGD Management Interface

The web management interfaces of IGD enabled devices should ensure all port forward rules created via UPnP are displayed, preferably in the same location that is used to manage other rules. Care should be taken to ensure that these rules can also be deleted or modified by the IGD administrator.

Additionally, logging should be enabled to ease the identification of the requesting host.

UPnP Authorization and Access Control

Instead of purely disabling UPnP, one alternative may be to provide some form of access control and authorization for UPnP requestors. Without modifying UPnP clients and servers the options are limited to providing access control and authorization.

A simplistic authorization and access control model might be to only honor AddPortMapping requests that come from known good hosts (perhaps hosts that were issued a DHCP address or pre-registered) and only if the request is to forward back to said host. This solution would not require modification the UPnP client applications.

A further refinement of the above solution might be to allow the end user to determine which UPnP actions they want to allow and which devices would be allowed to perform those actions.

The UPnP Forum has defined two profiles, the Device Security profile and the Security Console profile, that specifically handle secure authentication of SOAP actions. These standards resemble a PKI based infrastructure and rely on standard cryptographic protocols. Unfortunately these profiles do not appear to be generally implemented in consumer devices. Implementation of these profiles would most likely require additional software to be implemented on client systems. An alternative to implementing the Device Security and Security Console would be to develop a modified UPnP server and client that implemented WS-Secure or another standard based SOAP security mechanism.

UPnP Watchdog

One relatively low tech approach that I don't believe has been tried so far would be to develop an UPnP Watchdog application. This application would essentially monitor an IGD for changes to its port forwarding table and would notify the user that a change had occurred, perhaps offering the user the option to remove the change. The UPnP Watchdog application would either use polling or the SOAP event mechanism in combination with GetGenericPortMappingEntry to determine the active port mappings on an IGD.

UPnPwn

In order to further UPnP device testing, I have created a tool called UPnPwn. This tool and its associated libraries are implemented in the Ruby programming and provide basic classes for interacting with SSDP and UPnP enabled devices. Although this application is in its early stages of development I feel it is already useful for testing arbitrary UPnP enabled devices.

UPnPwn.rb

The application **UPnPwn.rb** provides a simple interface for performing basic IGD manipulations such as adding, removing and listing port mappings.

Example UPnPwn.rb usage:

Add a port forward from ANY -> WAN:1234/TCP -> 192.168.88.10
```
./UPnPwn.rb  --portmap add 1234 TCP 192.168.88.10
```

Delete portmap ANY -> WAN:1234/TCP -> 192.168.88.10
```
./UPnPwn.rb --portmap del 1234 TCP
```

Add a port forward from 11.22.33.44 -> WAN:4444/TCP -> 192.168.88.11:5555
```
./UPnPwn.rb --portmap add 11.22.33.44 4444 TCP 5555 192.168.88.11
```

Delete portmap 11.22.33.44 -> WAN:4444/TCP -> 192.168.88.11:5555
```
./UPnPwn.rb --portmap del 11.22.33.44 4444 TCP
```

Run a test server (for testing your port mappings, it will run on 0.0.0.0:1234)
```
./UPnPwn.rb --test_server
```

List all portmappings on a device
```
./UPnPwn.rb --list
```

Listen to the network for all SSDP traffic (this is a good way to find other UPnP Devices)
```
./UPnPwn.rb --ssdp_listen
```

Get info about first found IGD
```
./UPnPwn.rb --info
```

scpd2ruby.rb

UPnPwn provides an application, **scpd2ruby.rb**, which converts an XML SCPD description (from file or URL) to a static stub class that can be used to communicate with the service.

UPnPwn Class Library

The real power of the UPnPwn toolkit is in the class libraries it provides, specifically:
UPnPService

The UPnPService class is able to convert SCPD XML to a functioning ruby object that implements a method for each action described in the XML. These methods are created dynamically using instance_eval so that the resulting object provides a direct mapping to the SOAP service represented in the SCPD. Although this paper is focused primarily on UPnP IGD, UPnPService should be able to deal with any UPnP service that provided a SCPD XML description.

In order to further experimentation I strongly recommend using the UPnPService class in Ruby's irb. In my research I have found the ability to use tab completion in irb to be a great help.

The following demonstration code will:

1) Locate the first IGD to respond to a discovery
2) Parse all the services available
3) Select the WANIPConnection:1 service
4) Initialize the service
5) Print out information about the methods the service provides
6) Display the external WAN IP address of the IGD (via the dynamically created GetExternalIPAddress method)

```
#IGD WAN IP Sample
puts 'IGD WAN IP Sample'
require 'upnp'
require 'ssdp'

target_service='urn:schemas-upnp-org:service:WANIPConnection:1'

ssdp = SSDP.new
igd_url = ssdp.find_IGD

igd=UPnPHost.new
igd.location=igd_url
igd.load_xml_data
igd.parse_services

wanip_svc=igd.service[target_service]

wanip_svc.load_xml_data
wanip_svc.parse_scpd
wanip_svc.soap_init

puts wanip_svc.info

puts "IGD WAN IP: " + wanip_svc.GetExternalIPAddress
```

Sample output from “IGD WAN IP Sample”:

```
IGD WAN IP Sample
UPnPService:
Service Type: urn:schemas-upnp-org:service:WANIPConnection:1
Service ID:   urn:upnp-org:serviceId:WANIPConn1
Control URL:  http://192.168.50.88:2468/WANIPCtrlUrl
Event URL:    http://192.168.50.88:2468/WANIPEvtUrl
SCPD URL:     http://192.168.50.88:2468/WanIPDescDoc
# of Actions: 15
Actions:
SetConnectionType,GetConnectionTypeInfo,RequestConnection,RequestTermina
tion,ForceTermination,GetStatusInfo,GetAutoDisconnectTime,GetIdleDisconn
ectTime,GetWarnDisconnectDelay,GetNATRSIPStatus,GetGenericPortMappingEnt
ry,GetSpecificPortMappingEntry,AddPortMapping,DeletePortMapping,GetExter
nalIPAddress
IGD WAN IP: 192.168.2.100
```

Interested parties are strongly encouraged to paste the sample code into an irb session and to experiment with the resultant objects. Also consider reviewing **upnp_examples.rb** and other related source code files.

IGD Testing Matrix

The following matrix may be of assistance when testing IGD enabled devices.
This matrix attempts to capture the previously mentioned attack methods in a form that can be easily used to capture vulnerability data.

IGD	***IGD1***	***IGD2***	***IGD3***	***IGD4***
FWD to LAN (self)				
FWD to LAN (other)				
FWD to WAN				
FWD to IGD				
GetUserName				
GetPassword				
SetDNSServer				
SetIPRouter				
SetDHCPRelay				
UPnP on WAN				
Unvalidated Input				

Future Work

This paper is intended to raise awareness about the risks of enabling UPnP IGD on home routers. Very little attention is currently being paid to UPnP enabled devices and potential attack scenarios. It is the hope of the author that others will be inspired to focus research attention on the UPnP protocols and the various UPnP profiles currently available in the marketplace.

- Further development of UPnPwn
- UPnP Stack Fingerprinting
- UPnP Device and Profiles Database
- UPnP Fuzzer
- Code Execution / Buffer Overflows
- Windows Connect Now (WFADevice:1)
- UPnP A/V Profiles
- Other Dynamic configuration protocols

Contributing

I am always interested to hear about research related to UPnP and related protocols. I am especially interested in gaining access to additional information for fingerprinting UPnP device stacks. I'm also very interested in building a database of available UPnP devices and the profiles they provide.

If you feel you can provide assistance, please don't hesitate to contact me through my web site (http://www.bigbrainlabs.com) or email (***UPnPwn<at>bigbrainlabs<dot>com***).

References and Further Reading

- **UPnPwn updates and other research** *[Jonathan Squire]*
  - http://www.bigbrainlabs.com/
- **UPnP-hacks (Linux IGD attack)** *[Armijn Hemel]*
  - http://www.upnp-hacks.org/igd.html
- **GNUCitizen Flash UPnP Attack** *[Adrian Pastor and Petko D. Petkov]*
  - http://www.gnucitizen.org/blog/hacking-the-interwebs/
- **Crazy Toaster** *[Dror Shalev]*
  - http://www.drorshalev.com/dev/upnp/toaster/DC-15-Shalev-004.ppt
- **RFCs**
  - http://tools.ietf.org/html/draft-cai-ssdp-v1-03
  - http://tools.ietf.org/html/draft-cohen-gena-client-00
  - http://tools.ietf.org/html/draft-goland-http-udp-01
- **UPnP Forum**
  - http://www.upnp.org/standardizeddcps/igd.asp