

ファジング実践資料(UPnP 編)

「ファジング活用の手引き」別冊
DLNA に対応した情報家電に対するファジングの実践



本書は、以下の URL からダウンロードできます。

「ファジング実践資料(UPnP 編)」

<http://www.ipa.go.jp/security/vuln/fuzzing.html>

目次

はじめに	2
1 ファジングのおさらい	4
2 UPnP とは	5
2.1 UPnP における【探索】・【問い合わせ】・【制御】	7
2.2 UPnP 機能が受信するデータ	8
3 UPnP におけるファジング	12
3.1 UPnP におけるファジングの考え方	12
3.2 【探索】におけるファジング例	12
3.3 【制御】におけるファジング例	14
4 ファジング実践環境	15
4.1 ファジング実践環境における特記事項	16
5 【探索】におけるファジング	17
5.1 ファジングの流れ	17
5.2 ファジング実践	19
6 【制御】におけるファジング	27
6.1 ファジングの流れ	27
6.2 ファジング実践	30
6.3 ファジング実践（「Taof」編）	40
6.4 ファジング実践（「Peach」編）	46
付録1：UPnP の仕組み	59
付録2：Peach の XML 解析による設定ファイルの出力	70

はじめに

本書は、DLNA（「本書における用語」参照）に対応した情報家電（デジタルテレビや DVD レコーダーなど）に対するファジングの実践方法の一つをまとめたものです。

IPA では、2012 年 8 月から 2013 年 2 月に、情報家電の一つであるスマートテレビに対してファジングを実践しました。この実践過程にて、オープンソースソフトウェアを活用して UPnP（「本書における用語」参照）機能に対するファジングを実践して、脆弱性を検出しました。

近年 UPnP を基盤技術の一つとする DLNA に対応した情報家電が数多く市販されています。この現状も考慮し、スマートテレビに対するファジングで培ったノウハウを本書にまとめました。

対象読者

本書の対象読者には、主に次の方々を想定しています。特に、「品質保証担当者および開発者」に読んでいただきたいと考えています。

- 組込み機器開発に携わる品質保証担当者および開発者
- 組込み機器のセキュリティ対策に関わるシステム管理者や研究者

本書の使い方

本書は次の 3 つの使い方を想定しています。

(1). UPnP 機能に対してファジングを実践したい。

対象読者	● 品質保証担当者および開発者
使い方	● 「1 ファジングのおさらい」から「6 【制御】におけるファジング」まで順に読み進めてください。

(2). UPnP 機能に対するファジングの考え方を知りたい。

対象読者	● 品質保証担当者および開発者 ● システム管理者や研究者
使い方	● 「1 ファジングのおさらい」と「2 UPnP とは」、「3 UPnP におけるファジング」を読んでください。

(3). UPnP の仕組みを知りたい。

対象読者	● UPnP の仕組みをよく知らない方
使い方	● 「2 UPnP とは」と「付録 1 : UPnP の仕組み」を読んでください。

本書における用語

■ 「情報家電」

本書では、インターネットに接続できる家電製品（デジタルテレビや DVD レコーダーなど）を「情報家電」と定義します。

■ 「スマートテレビ」

本書では、「テレビ放送を視聴できるだけでなく、インターネットや他の情報家電と接続することで、ウェブサイトや静止画などの閲覧や動画の再生などを実現できる多機能なテレビ」を「スマートテレビ」と定義します。

■ 「DLNA」

Digital Living Network Association の略称。情報家電が他のメーカーの情報家電と連携するための仕様を策定する組織¹、またはその仕様そのものを指します。本書で DLNA と記述した場合、「情報家電が他のメーカーの情報家電と連携するための仕様そのもの」を指します。

■ 「UPnP」 Universal Plug and Play の略称。本書では、ネットワークに接続するだけで組み込み機器やパソコンなどの機器同士が相互に連携できる仕組みを「UPnP」と定義します。「付録 1 : UPnP の仕組み」で、図を交えながら UPnP の仕組みを説明します。

本書を読む上での注意事項

- 本書では、オープンソースソフトウェア等を活用した UPnP におけるファジング手順の説明に注力しているため、次の内容を説明していません。これらの内容をご存じでない方は、あらかじめ「ファジング活用の手引き²」、「ファジング実践資料³」をご一読ください。
 - ✓ ファジングの考え方および実践手順
 - ✓ ファジングツール「Taof」や「Peach」の使い方

¹ デジタルリビング ネットワーク アライアンスについて

http://www2.dlna.org/about_us

² IPA : 「ファジング活用の手引き」

<http://www.ipa.go.jp/security/vuln/documents/fuzzing-guide.pdf>

³ IPA : 「ファジング実践資料」

<http://www.ipa.go.jp/security/vuln/documents/fuzzing-tool.pdf>

1 ファジングのおさらい

まずファジングそのものの要点をおさらいします。ファジングの説明とイメージ図、および本資料で関係するファジングの特徴を図 1 にまとめました。

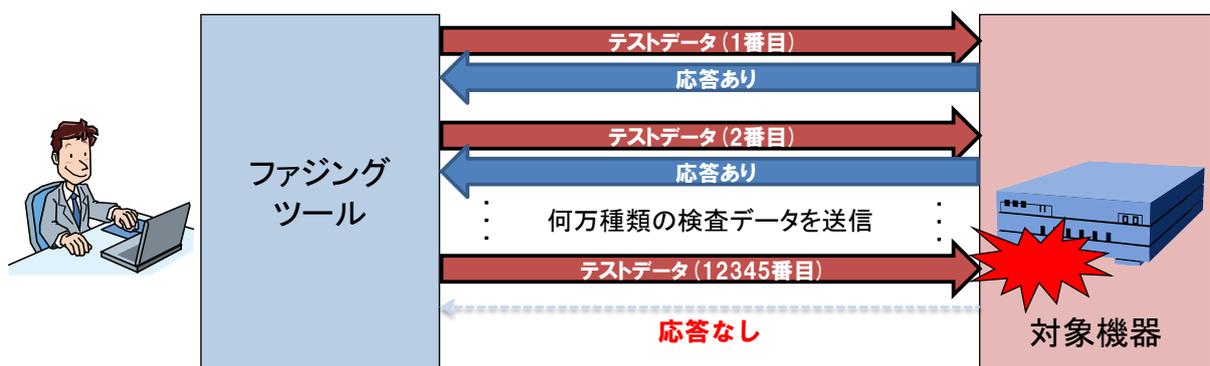
対象機器にファジングを実施する場合、その対象機器が受信するデータに注目します。対象機器の UPnP 機能に対してファジングを実践する場合、「UPnP 機能がどんなデータを受信するか」が重要です。

ファジングとは

何万種類もの問題を起こしそうなデータ(例: 極端に長い文字列)を対象機器に送り込み、対象製品の動作状態(例: 製品が異常終了する)から脆弱性などを発見する技術。

ファジングの特徴

- 対象機器の内部構造を考慮せずに、データの入出力(特に、**入力値**)に注目することで脆弱性などを発見する技術。



【イメージ図】

図 1 ファジングのおさらい

2 UPnP とは

UPnP とは、ネットワークに接続するだけで組み込み機器やパソコンなどの機器同士が相互に連携できる仕組みです。この仕組みはパソコンメーカーや OS メーカーなどで構成される業界団体「UPnP Forum⁴」によって策定されています。

近年では、機器間の相互連携を実現する DLNA に対応した情報家電において、UPnP が基盤技術の一つとして活用されています。DLNA に対応した情報家電には、デジタルテレビや DVD プレーヤー、ネットワーク接続ハードディスク (NAS)、スマートフォン、パソコンなどがあります (図 2)。こういった情報家電の裏側では、UPnP 機能が動作していると考えられます。

本書では、「情報家電」や「組み込み機器」と書いた場合、「DLNA に対応しており、UPnP 機能を実装している情報家電や組み込み機器」を指します。

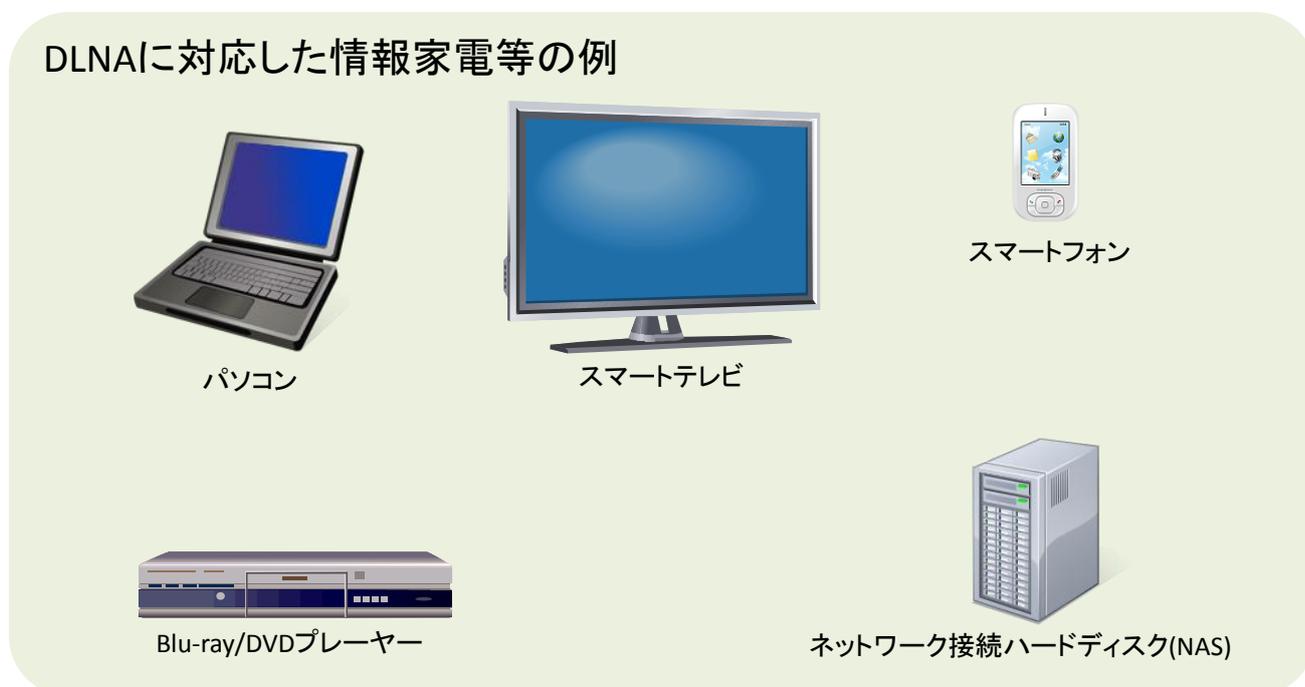


図 2 DLNA に対応した情報家電等の例

⁴ UPnP Forum <http://upnp.org/>

情報家電では、UPnP を使って次のような機能を実現できます⁵。

◆ **ネットワーク接続ハードディスクに保存している動画をデジタルテレビで視聴する。**

UPnP を使うと、同じネットワークに接続した情報家電に保存している動画を別の情報家電で視聴できます。

図 3 のように、同じネットワークにネットワーク接続ハードディスク (図 3 左)、デジタルテレビ (図 3 右) がある場合、ネットワーク接続ハードディスクに保存した動画をこのデジタルテレビで視聴できます。

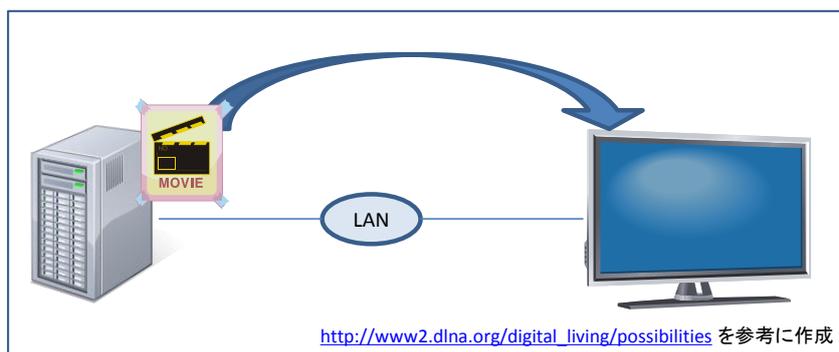


図 3 UPnP を使った動画の視聴

◆ **スマートフォンに保存している画像ファイルをデジタルテレビで閲覧する。**

UPnP を使うと、同じネットワークに接続した情報家電に保存している画像ファイルを別の情報家電で閲覧できます。

図 4 のように、同じネットワークにスマートフォン (図 4 左)、デジタルテレビ (図 4 右) がある場合、スマートフォンに保存している画像ファイルをデジタルテレビで閲覧できます。

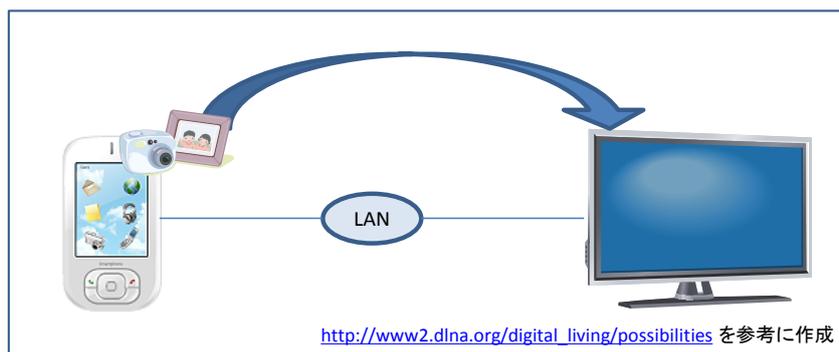


図 4 UPnP を使った画像の閲覧

⁵ ここで掲載した機能については、DLNA のウェブサイト

(http://www2.dlna.org/digital_living/possibilities) を基にまとめています。厳密には、掲載する機能については UPnP だけで実現できるわけではありませんが、本書では「情報家電で実現できることと UPnP の関係を具体的にイメージできる」ことを目的として UPnP のみに言及しています。

2.1 UPnP における【探索】・【問い合わせ】・【制御】

UPnP の仕組みは、大きく【探索】・【問い合わせ】・【制御】の 3 つに分けることができます (図 5)。UPnP の規格書⁶では、この 3 つをそれぞれ「Discovery step」、「Description step」、「Control step」と定義しています。

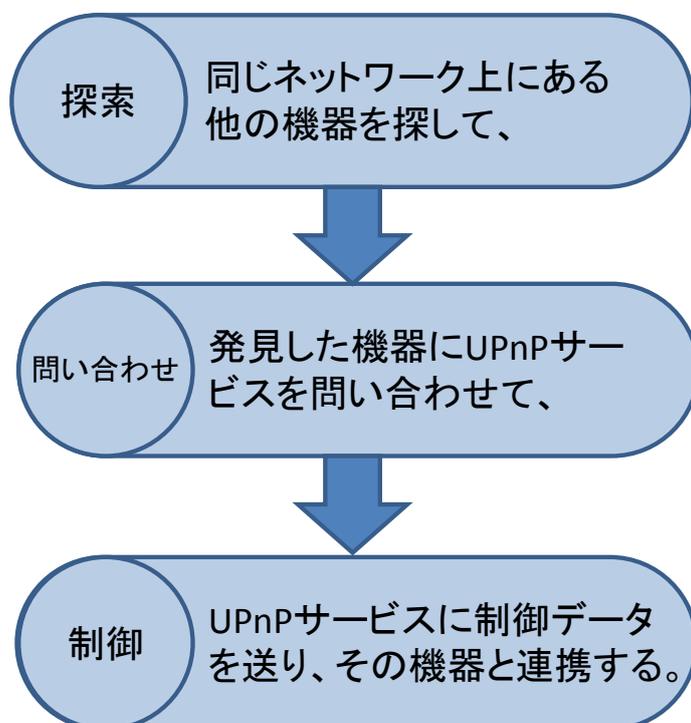


図 5 UPnP における【探索】・【問い合わせ】・【制御】

⁶ Device Architecture Documents >> UPnP Forum

<http://upnp.org/index.php/sdcps-and-certification/standards/device-architecture-documents/>

本書では、UPnP の規格書といった場合、「UPnP Device Architecture version 1.1」を指します。

2.2 UPnP 機能が受信するデータ

UPnP における【探索】・【問い合わせ】・【制御】では、HTTP を基に機器間でデータをやり取りしています⁷。UPnP 機能に対するファジングを実践するうえでは、UPnP 機能が受信するデータ（HTTP リクエスト）を把握する必要があります。

本節では、【探索】・【問い合わせ】・【制御】のそれぞれで UPnP 機能が受信する HTTP リクエストを説明します。本書で記載している HTTP リクエストとして、Windows 7 で動作している UPnP 機能が受信する HTTP リクエストを例示しました。

なお本節では、「組み込み機器をネットワークに接続したときに、【探索】・【問い合わせ】・【制御】において機器間でどのような流れで HTTP 通信が生じるか」を説明していません。興味がある方は「付録 1 : UPnP の仕組み」をご参照ください。

2.2.1 【探索】において UPnP 機能が受信する HTTP リクエスト

UPnP 機能を実装した機器は、【探索】において「探索リクエスト」と「広告リクエスト」を受信します。それぞれのリクエストの役割を表 1 にまとめました。UPnP の規格書では、「探索リクエスト」は「M-SEARCH リクエスト」、「広告リクエスト」は「NOTIFY リクエスト」と定義されています。

表 1 【探索】における HTTP リクエスト

リクエスト名称	リクエストの役割
探索リクエスト	同じネットワークに存在する「UPnP 機能を実装した機器」を探す。 (このリクエストへの応答で、機器の存在を認識する)
広告リクエスト	同じネットワークに存在する「UPnP 機能を実装した機器」に自身の存在を知らせる。 (このリクエストを受信した機器は、リクエストを送信した機器の存在を認識する)

⁷ 厳密には SSDP(Simple Service Discovery Protocol)や SOAP などの複数のプロトコルを組み合わせて実現しています。しかし、SSDP や SOAP も HTTP に基づいているため、本書では分かりやすさを重視して SSDP や SOAP には言及していません。

UPnP 機能が受信する「探索リクエスト」と「広告リクエスト」は、それぞれ図 6、図 7 のような内容です。

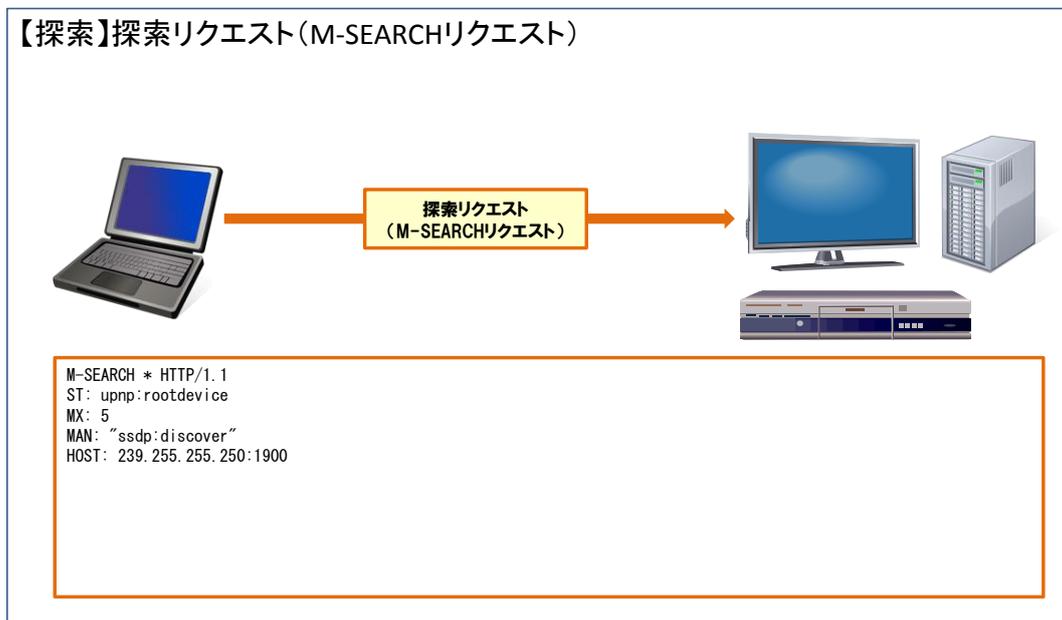


図 6 探索リクエスト

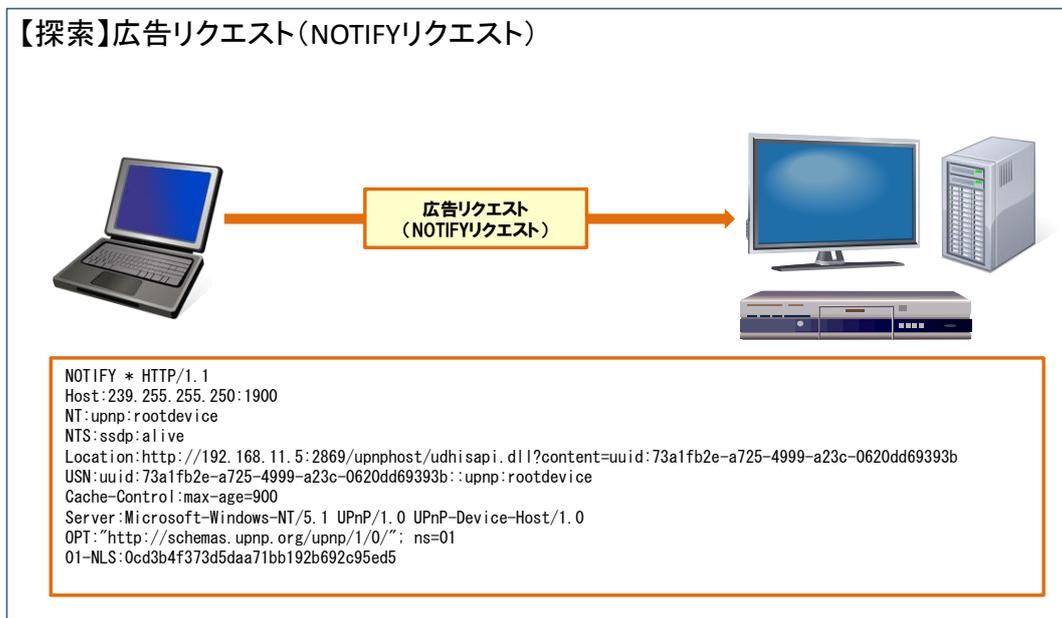


図 7 広告リクエスト

2.2.2 【問い合わせ】において UPnP 機能が受信する HTTP リクエスト

UPnP 機能を実装した機器は、【問い合わせ】において「問い合わせリクエスト」を受信します。「問い合わせリクエスト」の役割を表 2 にまとめました。この「問い合わせリクエスト」は、ウェブサーバが受信する GET リクエストと大きな違いはありません。

本書では、機器の UPnP 機能が提供できる一つの機能を「UPnP サービス」と定義します。例えば、「別の機器に動画を配信すること」、「別の機器に画像を提供すること」それぞれが「UPnP サービス」に該当します。

表 2 【問い合わせ】における HTTP リクエスト

リクエスト名称	リクエストの役割
問い合わせ リクエスト	UPnP 機能が提供する「UPnP サービス」を問い合わせる。 (このリクエストへの応答で、その機器がどんなことを実現できるか分かる)

UPnP 機能が受信する HTTP リクエストは、図 8 のような内容です。

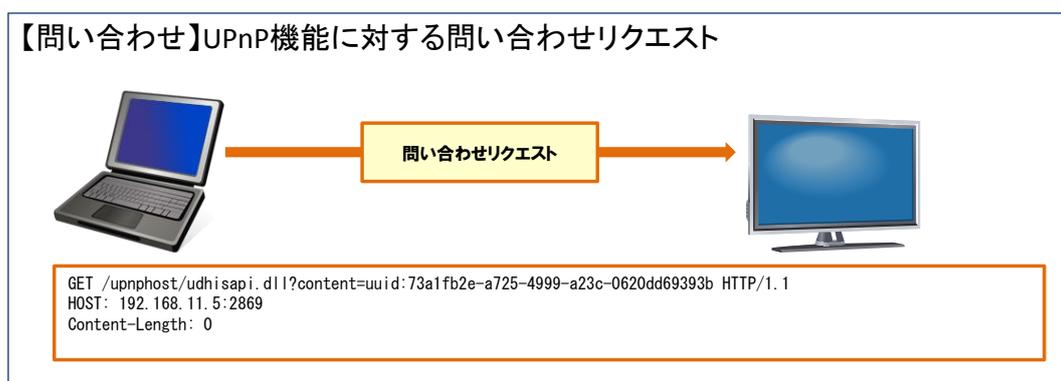


図 8 UPnP 機能に対する問い合わせリクエスト

2.2.3 【制御】において UPnP サービスが受信する HTTP リクエスト

【制御】では、UPnP 機能を実装した機器の「UPnP サービス」それぞれが HTTP リクエストを受信します。それぞれの「UPnP サービス」は、【制御】において「制御リクエスト」を受信します。この「制御リクエスト」の役割を表 3 にまとめました。

制御リクエストを受信する「UPnP サービス」によって、受信する制御データが異なります。制御データは、制御リクエストの中に XML 形式で記述されます。

表 3 【制御】における HTTP リクエスト

リクエスト名称	リクエストの役割
制御リクエスト	「UPnP サービス」に制御データを送信する。 (このリクエストによって、UPnP を通じて機器を制御する)

UPnP 機能が受信する HTTP リクエストは、図 9 のような内容です。この図では、制御データ「DeviceID」として「AAAAA」という値を送信しています（図 9 の赤点線部）。

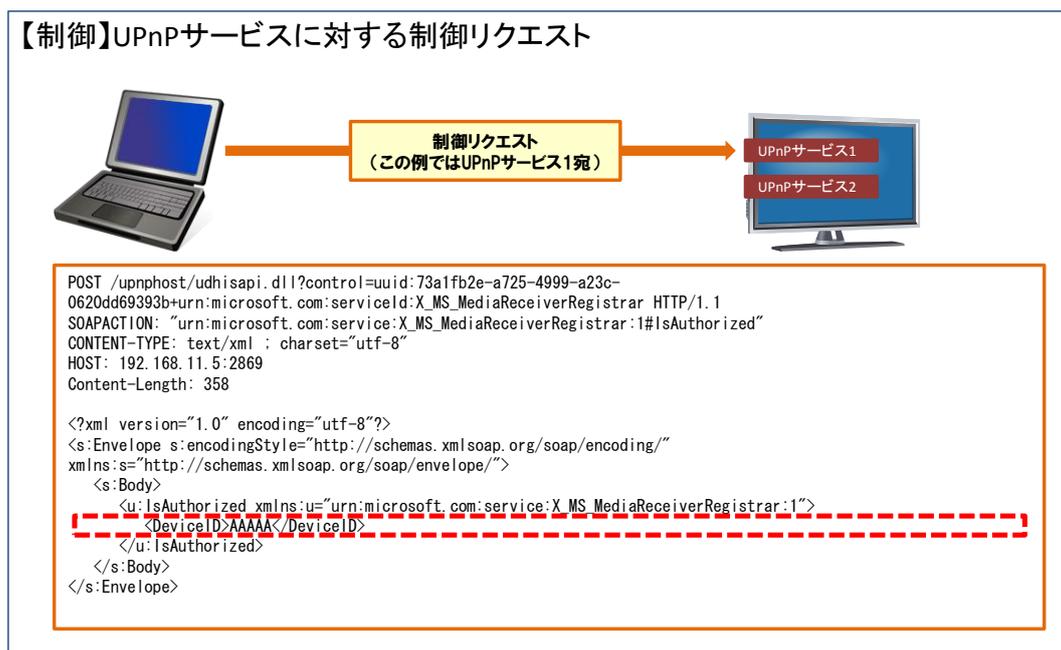


図 9 UPnP サービスに対する制御リクエスト

UPnP 機能が受信する HTTP リクエストの内容をふまえて、3 章ではオープンソースソフトウェアを使って、UPnP におけるファジングを実践することを考えてみます。

3 UPnP におけるファジング

2章をふまえて、オープンソースソフトウェアを使った、UPnP におけるファジング例を説明します。本章では、UPnP におけるファジングの考え方を説明した後、本資料で取り上げるファジング例を説明します。

3.1 UPnP におけるファジングの考え方

ファジングでは、ファジング対象機器が受信するデータの一部を「問題を引き起こしそうなデータ」に細工します。UPnP の場合、ファジング対象機器は HTTP リクエストでデータを受信することから、HTTP リクエストの一部を「問題を引き起こしそうなデータ」に細工することで、ファジングを実現できます。

UPnP ではウェブサイトを閲覧するときに使う HTTP に基づいているため、UPnP におけるファジングと、ウェブサーバに対するファジングでは基本的に同じ考え方でファジングを実施できます。特に、【問い合わせ】の「問い合わせリクエスト」を使ってファジングを実施する場合、ウェブサーバに対するファジングと同じ方法で実現できます。

本資料では、【探索】・【問い合わせ】・【制御】におけるファジングのうち、UPnP の仕組みを考慮してファジングを実施する必要がある【探索】および【制御】におけるファジング例を紹介します。

3.2 【探索】におけるファジング例

【探索】におけるファジングでは、ファジングツール「Peach」を使って、テストデータに置き換えた「探索リクエスト」および「広告リクエスト」をファジング対象機器に送信します（図 10 と図 11 の赤色部分）。

「探索リクエスト」と「広告リクエスト」は UPnP の規格書で内容が定義されているため、どのような機器であってもリクエストデータをすべて処理する可能性が高いと考えます。リクエストデータを全般的にテストデータに置き換えることを得意とする「Peach」を使うことで、【探索】におけるファジングを効率的に実施できる⁸と考えます。

⁸ 【探索】におけるファジングではファジングツール「Taof」を使用できません。【探索】ではマルチキャスト IP アドレスにテストデータを送信する必要がありますが、「Taof」ではマルチキャスト IP アドレス宛にテストデータを送信できませんでした。

【探索】探索リクエスト(M-SEARCHリクエスト)

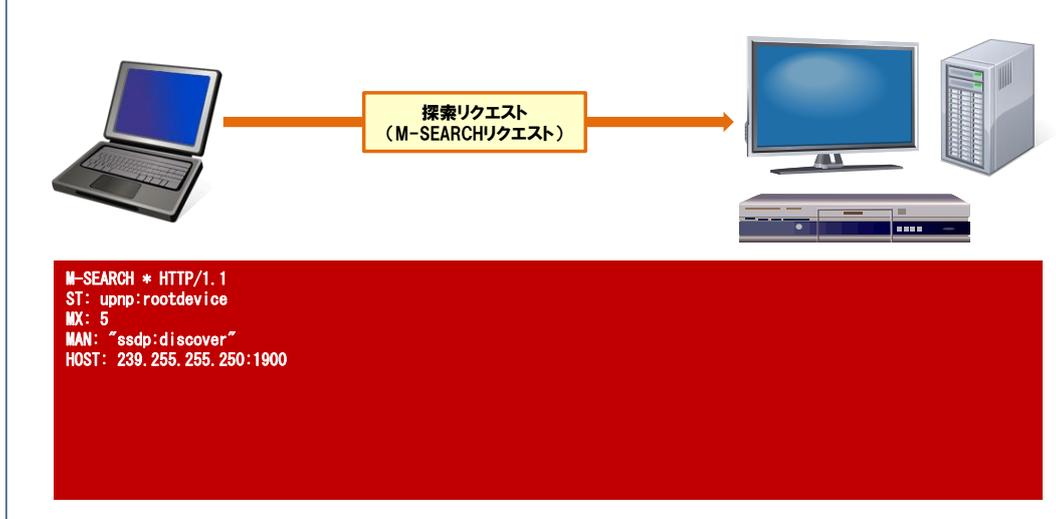


図 10 探索リクエストでテストデータを置き換える箇所 (赤色部分)

【探索】広告リクエスト(NOTIFYリクエスト)

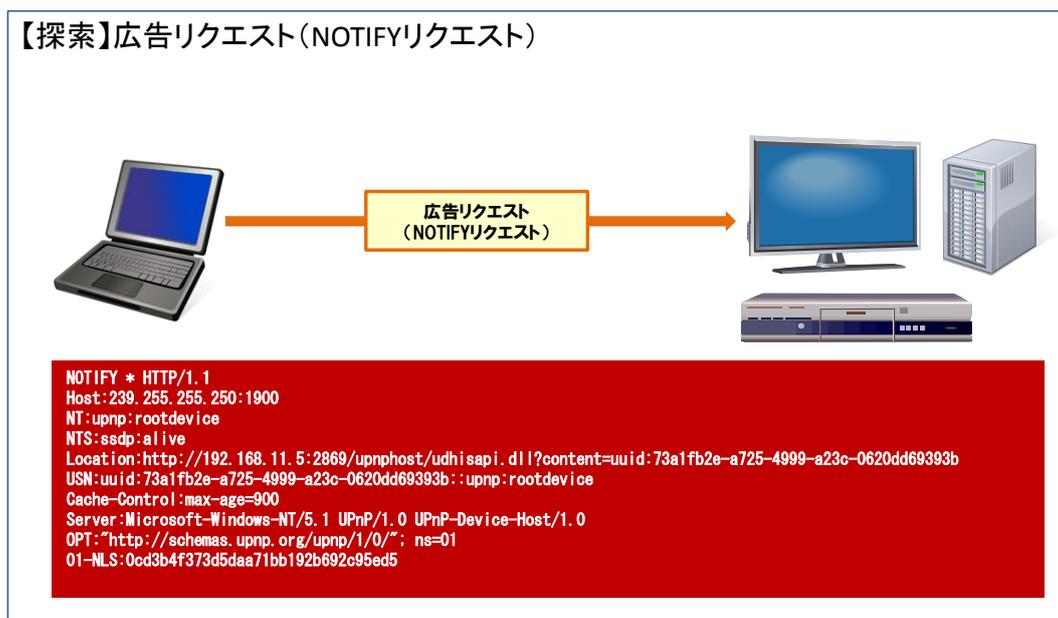


図 11 広告リクエストでテストデータと置き換える箇所 (赤色部分)

3.3 【制御】におけるファジング例

【制御】におけるファジングでは、ファジングツール「Taof」と「Peach」を使って、制御データをテストデータに置き換えた「制御リクエスト」をファジング対象機器に送信します（図 12 の赤色部分）。

制御リクエストの制御データには、UPnP の規格書で定義されているものに加えて、メーカーが独自に定義するものがあります。制御データの種類が増えると、その分だけ脆弱性等を作り込んでしまう可能性が高まってしまいます。このため、できるだけ多くのテストデータでファジングを実施するが望ましいと考えます。テストデータが異なる（図 13）ファジングツール「Taof」と「Peach」を併用することで、幅広いセキュリティテストの実施につながります。

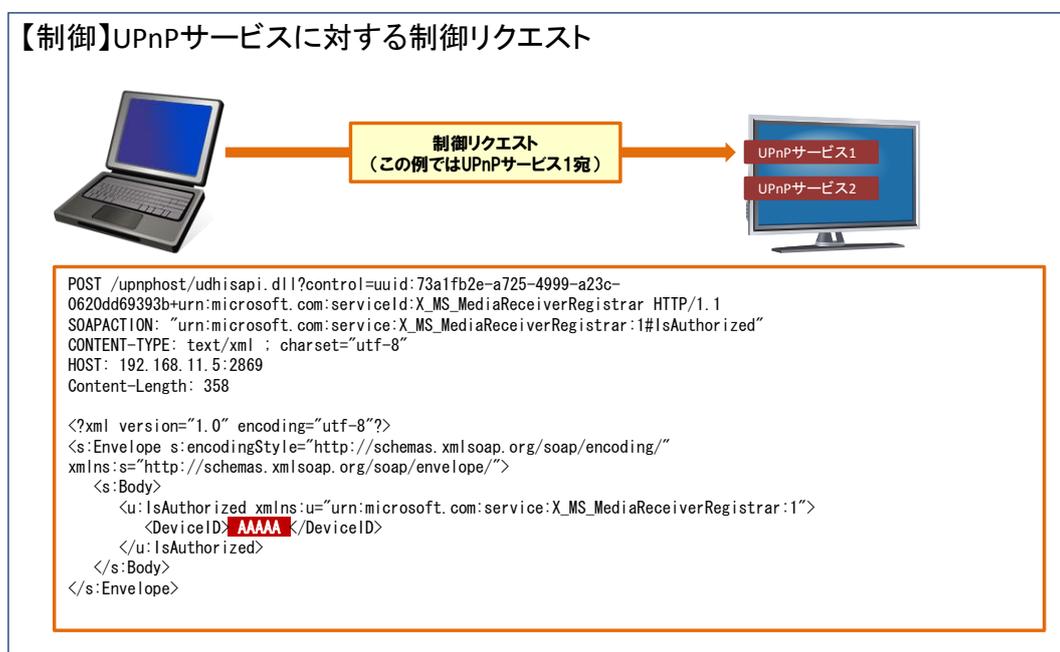


図 12 制御リクエストでテストデータに置き換える箇所（赤色部分）

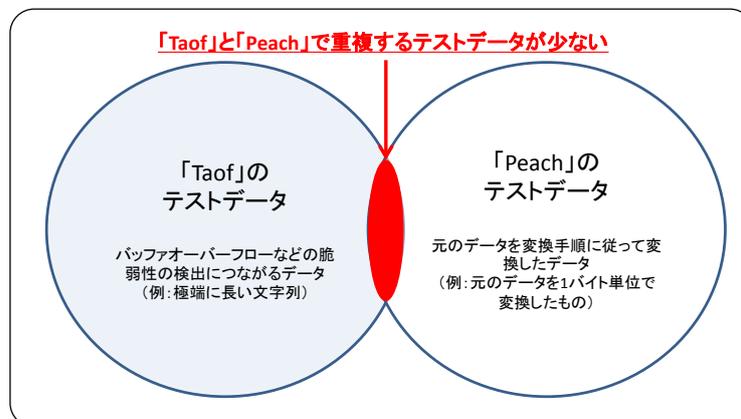


図 13 「Taof」と「Peach」のテストデータの違い

4 ファジング実践環境

本手順書で紹介するファジング例については、図 14 のファジング環境で実践します。

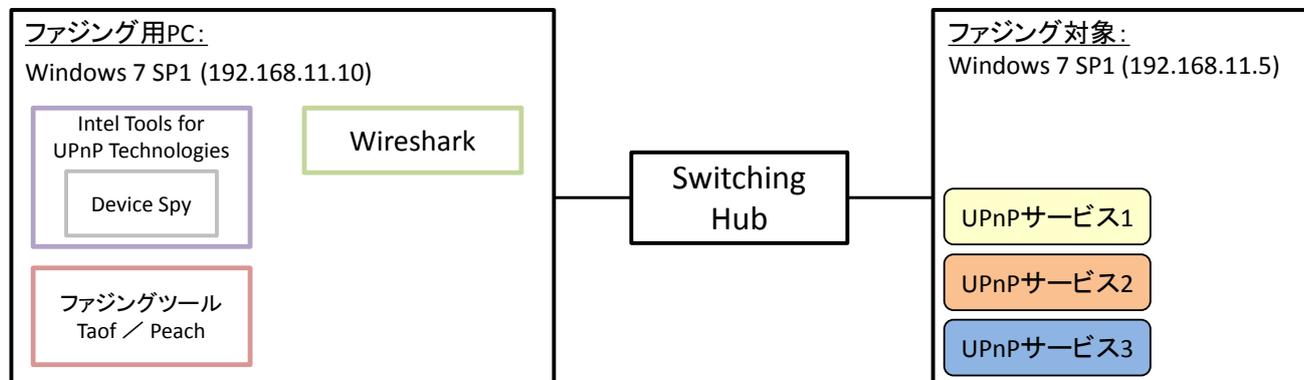


図 14 ファジング実践環境

ファジング用 PC には、事前に表 4 のツールをインストールしておきます。補助ツール「Intel Tools for UPnP Technologies」の「Device Spy」と「Wireshark」は、【制御】におけるファジングにて使用します。

「Device Spy」は、「UPnP 機能を実装した機器」の探索および「UPnP サービス」の調査、探索した機器に対する制御リクエストの送信などを実現できるツールです。このツールを使って、ファジング対象機器が提供する「UPnP サービス」を調査して、ファジングの基となる制御リクエストを生成します。

「Wireshark」は、動作している PC が送受信するパケットを取得できるツールです。このツールを使って、「UPnP サービス」に対して送信した制御リクエストを取得します。

表 4 ファジング実践環境で使用するツール

ツール分類	ツール名	補足
ファジング ツール	Taof ⁹	使用バージョン：0.2.3
	Peach ¹⁰	使用バージョン：2.3.8
補助ツール	Intel Tools for UPnP Technologies ¹¹	UPnP に関連した調査を実施できるツール群。ツール群のうち「Device Spy」を使用します。 使用バージョン：1.3.2777.23551
	Wireshark ¹²	使用バージョン：1.8.5

※各ツールの使用バージョンは、2013年2月末日の最新バージョンとしています。

⁹ <http://sourceforge.net/projects/taof/>

¹⁰ <http://peachfuzzer.com/>

¹¹ <http://software.intel.com/en-us/articles/intel-software-for-upnp-technology-download-tools/>

¹² <http://www.wireshark.org/>

4.1 ファジング実践環境における特記事項

- 本手順書では、ファジング対象として Windows 7 SP1 の UPnP 機能を例示しています。しかし、IPA では実際に Windows 7 SP1 の UPnP 機能に対してファジングを実施していません。
- 本書で収録している手順は、2013 年 2 月末日時点に動作確認しています。読者の動作環境などによっては本書の手順では正しく動作しない可能性があります。

5 【探索】におけるファジング

UPnP 機能の【探索】に対しては、ファジングツール「Peach」を使ってファジングを実施します。このファジングでは、【探索】の探索リクエストと広告リクエストから、テストデータを生成してファジング対象機器に送信します。

5.1 ファジングの流れ

UPnP 機能の【探索】におけるファジングは、探索リクエストまたは広告リクエストをファジングツールに設定する作業と、その設定ファイルを使ってファジングツールを実施する作業の 2 つの作業に分かれます（図 15）。5.1.1 節、5.1.2 節でそれぞれの作業の概要を説明します。

① ファジングツールの設定

② ファジングツールによるファジング



図 15 UPnP 機能の【探索】におけるファジング作業

5.1.1 ファジングツールの設定

まず【探索】の探索リクエストと広告リクエストデータをファジングツールの設定ファイルに反映します。探索リクエスト、広告リクエストはそれぞれ別の内容であるため、それぞれ別の設定ファイルに反映することになります。

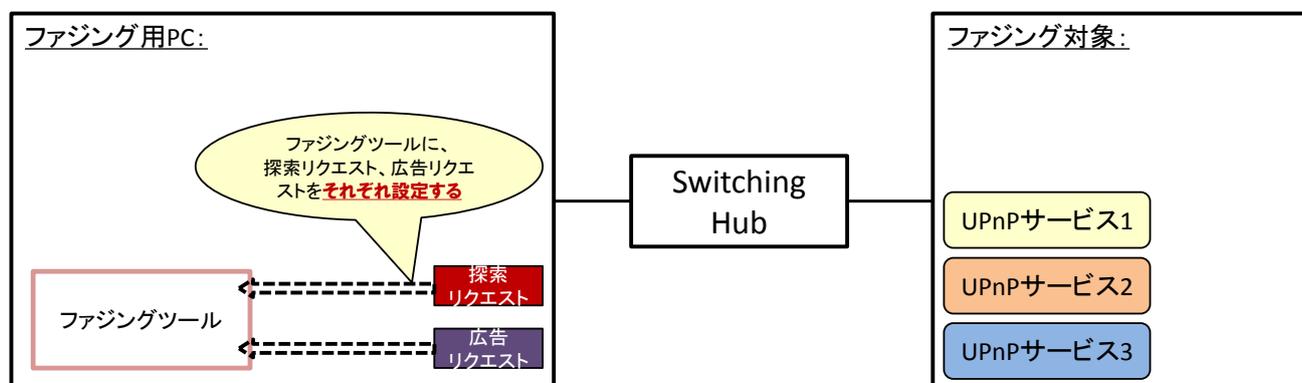


図 16 ファジングツールの設定

5.1.2 ファジングツールによるファジング

ファジングツールの設定が完了したら、あとはファジングを実践するだけです。

探索リクエストを反映した設定ファイルと、広告リクエストを反映した設定ファイルごとに、2回ファジングを実践します（図 17）。

ファジングツール「Peach」の使い方は、IPA 公開資料「ファジング実践資料¹³」で紹介しています。「Peach」の使い方に馴染みがない方は、その資料もご参照ください。

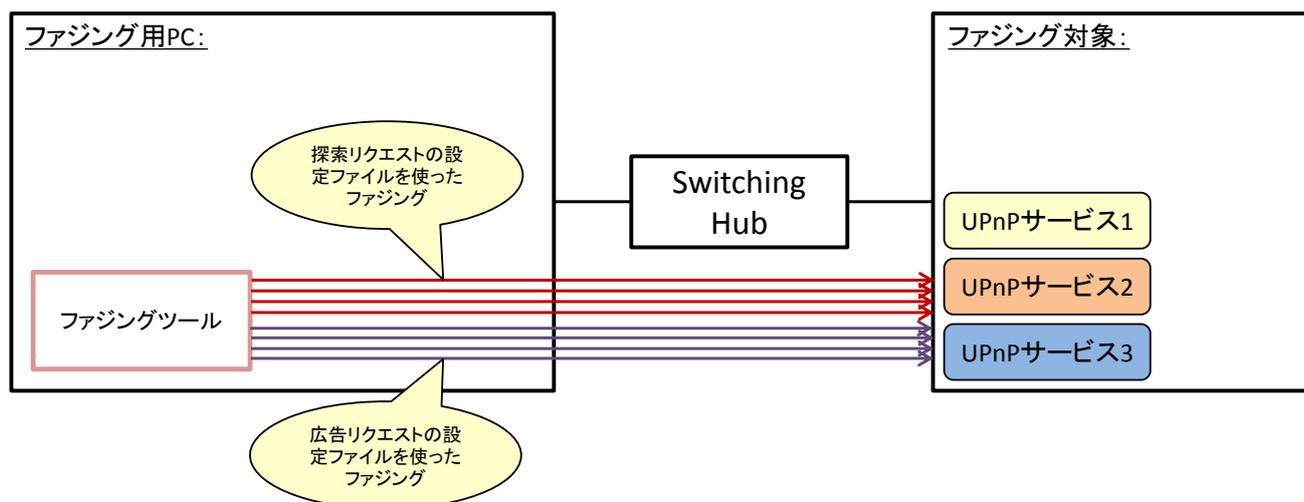


図 17 ファジングツールによるファジング

以上で、UPnP 機能の【探索】におけるファジングの流れの説明を終わります。この流れを基に、5.2 節でツールの具体的な操作を交えながらファジングの実践手順を説明していきます。

¹³ IPA : 「ファジング実践資料」

<http://www.ipa.go.jp/security/vuln/documents/fuzzing-tool.pdf>

5.2 ファジング実践

本節では、「Windows 7 SP1」の UPnP 機能に対するファジングを例にとり、【探索】におけるファジング手順を説明します。

【探索】におけるファジングでは、大きく 2 つの作業「ファジングツールの設定」、「ファジングツールによるファジング」があることを説明しました。この 2 つのうち、「ファジングツールの設定」についてはさらに 2 つの作業に分けることができます (図 18)。5.2.1 節から順に図 18 の作業項目を説明していきます。

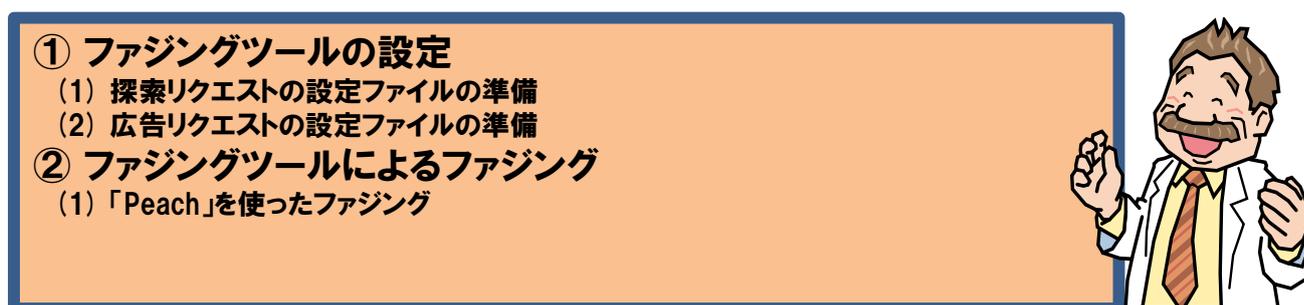


図 18 【探索】におけるファジングの作業項目

5.2.1 探索リクエストの設定ファイルの準備

まず探索リクエストの設定ファイルを準備します (図 19)。UPnP の規格書には、探索リクエストが具体的に定義されている¹⁴ため、規格書で定義されている内容をそのまま設定ファイルに反映します。

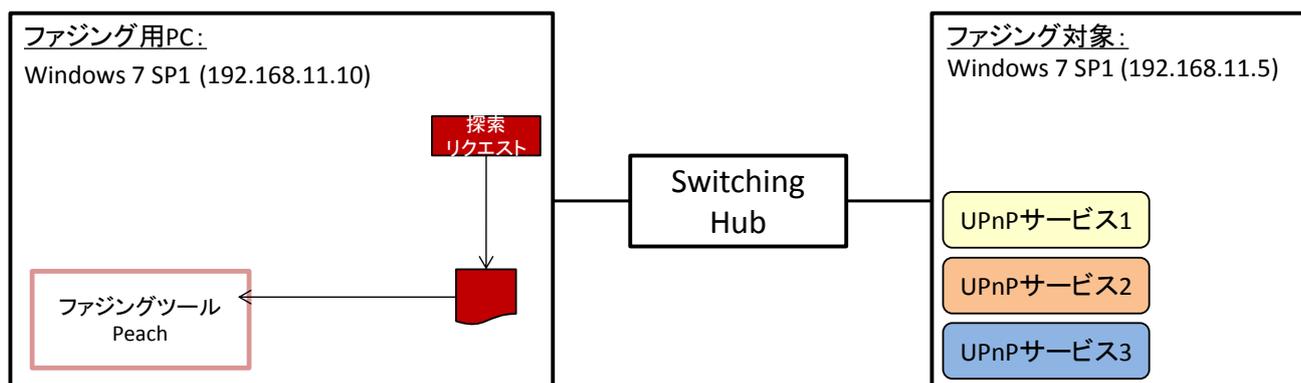


図 19 探索リクエストの設定ファイルの準備 (イメージ図)

¹⁴ pp.30-33, 1.3.2 Search request with M-SEARCH, UPnP Device Architecture 1.1
<http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf>

表 5 は、UPnP の規格書に基づく探索リクエストを反映した Peach の設定ファイルです。本手順では、表 5 の内容を「upnp_msearch.xml」というファイル名で保存します。

表 5 の内容は別途 IPA のウェブサイト¹⁵でも公開しております。紙面の都合上、表 5 では設定ファイルの補足説明等を割愛しています。実際にファジングで活用する場合は IPA のウェブサイトから設定ファイルをダウンロードして、ご活用ください。

表 5 探索リクエストの設定ファイル

1	<?xml version="1.0" encoding="utf-8"?>
2	<Peach xmlns="http://phed.org/2008/Peach" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://phed.org/2008/Peach ../peach.xsd" version="1.0" author="Information-technology Promotion Agency(IPA), Japan" description="Pit for UPnP M-SEARCH Request Fuzzing">
3	
4	<!-- Import defaults for Peach instance -->
5	<Include ns="default" src="file:defaults.xml" />
6	
7	<!-- DataModel for HTTP Header -->
8	<DataModel name="Header">
9	<String name="Header"/>
10	<String value=": "/>
11	<String name="Value"/>
12	<String value="%r%n" mutable="false"/>
13	</DataModel>
14	
15	<DataModel name="UpnpMSEARCHRequest">
16	
17	<!-- M-SEARCH * HTTP/1.1 -->
18	<Block name="RequestLine">
19	<String name="Method" value="M-SEARCH"/>
20	<String value=" " type="char" mutable="false"/>
21	<String name="RequestUri" value="*/>
22	<String value=" " mutable="false"/>
23	<String name="HttpVersion" value="HTTP/1.1"/>
24	<String value="%r%n" mutable="false"/>
25	</Block>

¹⁵ UPnP の探索リクエストに関する Peach 設定ファイル

https://www.ipa.go.jp/security/vuln/documents/fuzzing/upnp_msearch.xml

```

26
27 <!-- ST: upnp:rootdevice -->
28 <Block name="HeaderST" ref="Header">
29   <String name="Header" value="ST"/>
30   <String name="Value" value="upnp:rootdevice"/>
31 </Block>
32
33 <!-- MX: 5-->
34 <Block name="HeaderMX" ref="Header">
35   <String name="Header" value="MX"/>
36   <String name="Value" value="5"/>
37 </Block>
38
39 <!-- MAN: "ssdp:discover" -->
40 <Block name="HeaderMAN" ref="Header">
41   <String name="Header" value="MAN"/>
42   <String name="Value" value="&quot;ssdp:discover&quot;"/>
43 </Block>
44
45 <!-- HOST: 239.255.255.250:1900 -->
46 <Block name="HeaderHOST" ref="Header">
47   <String name="Header" value="HOST"/>
48   <String name="Value" value="239.255.255.250:1900"/>
49 </Block>
50
51 <String value="¥r¥n"/>
52 </DataModel>
53
54 <StateModel name="State1" initialState="Initial">
55   <State name="Initial">
56     <Action type="output">
57       <DataModel ref="UpnpMSEARCHRequest" />
58     </Action>
59   </State>
60 </StateModel>
61
62 <Test name="UpnpMSEARCHRequestTest" description="Upnp M-SEARCH Request Test">
63   <StateModel ref="State1"/>

```

```

64 <Publisher class="udp.Udp">
65   <Param name="host" value="239.255.255.250" />
66   <Param name="port" value="1900" />
67 </Publisher>
68 </Test>
69
70 <Run name="DefaultRun" description="Upnp M-SEARCH Request Run">
71   <Test ref="UpnpMSEARCHRequestTest" />
72   <Logger class="logger.Filesystem">
73     <Param name="path" value="c:\peach\log%" />
74   </Logger>
75 </Run>
76 </Peach>

```

5.2.2 広告リクエストの設定ファイルの準備

続いて、広告リクエストの設定ファイルに準備します（図 20）。探索リクエストと同様に、UPnP の規格書には広告リクエストが具体的に定義されている¹⁶ため、規格書で定義されている内容をそのまま設定ファイルに反映します。

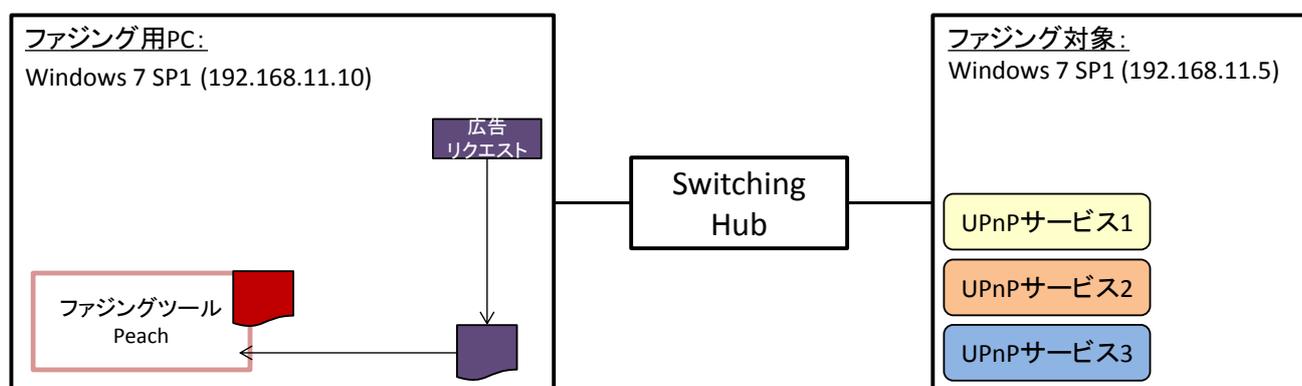


図 20 広告リクエストの設定ファイルの準備（イメージ図）

¹⁶ pp.19-25, 1.2.2 Device available - NOTIFY with ssdp:alive, UPnP Device Architecture 1.1
<http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf>

表 6 は、UPnP の規格書に基づく 広告リクエストを反映した Peach の設定ファイルです。本手順では、表 6 の内容を「upnp_notify.xml」というファイル名で保存します。

表 6 の内容は別途 IPA のウェブサイト¹⁷でも公開しております。紙面の都合上、表 6 では設定ファイルの補足説明等を割愛しています。実際にファジングで活用する場合は IPA のウェブサイトから設定ファイルをダウンロードして、ご活用ください。

表 6 広告リクエストの設定ファイル

1	<?xml version="1.0" encoding="utf-8"?>
2	<Peach xmlns="http://phed.org/2008/Peach" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://phed.org/2008/Peach ../peach.xsd" version="1.0" author="Information-technology Promotion Agency(IPA), Japan" description="Pit for UPnP NOTIFY Request Fuzzing">
3	
4	<!-- Import defaults for Peach instance -->
5	<Include ns="default" src="file:defaults.xml" />
6	
7	<!-- DataModel for HTTP Header -->
8	<DataModel name="Header">
9	<String name="Header"/>
10	<String value=": "/>
11	<String name="Value"/>
12	<String value="¥r¥n" mutable="false"/>
13	</DataModel>
14	
15	<DataModel name="UpnpNOTIFYRequest">
16	<!-- NOTIFY * HTTP/1.1 -->
17	<Block name="RequestLine">
18	<String name="Method" value="NOTIFY"/>
19	<String value=" " type="char" mutable="false"/>
20	<String name="RequestUri" value="*/>
21	<String value=" " mutable="false"/>
22	<String name="HttpVersion" value="HTTP/1.1"/>
23	<String value="¥r¥n" mutable="false"/>
24	</Block>
25	

¹⁷ UPnP の広告リクエストに関する Peach 設定ファイル

https://www.ipa.go.jp/security/vuln/documents/fuzzing/upnp_notify.xml

```

26 <!-- HOST: 239.255.255.250:1900 -->
27 <Block name="HeaderHOST" ref="Header">
28   <String name="Header" value="HOST"/>
29   <String name="Value" value="239.255.255.250:1900"/>
30 </Block>
31
32 <!-- CACHE-CONTROL: max-age=180 -->
33 <Block name="HeaderCACHE-CONTROL" ref="Header">
34   <String name="Header" value="CACHE-CONTROL"/>
35   <String name="Value" value="max-age=180"/>
36 </Block>
37
38 <!-- LOCATION: http://192.168.11.15:8080/upnp.xml -->
39 <Block name="HeaderLOCATION" ref="Header">
40   <String name="Header" value="LOCATION"/>
41   <String name="Value" value="http://192.168.11.15:8080/upnp.xml"/>
42 </Block>
43
44 <!-- NT: upnp:rootdevice -->
45 <Block name="HeaderNT" ref="Header">
46   <String name="Header" value="NT"/>
47   <String name="Value" value="upnp:rootdevice"/>
48 </Block>
49
50 <!-- NTS: ssdp:alive -->
51 <Block name="HeaderNTS" ref="Header">
52   <String name="Header" value="NTS"/>
53   <String name="Value" value="ssdp:alive"/>
54 </Block>
55
56 <!-- SERVER: OS/1.0 UPnP/1.1 product/1.0 -->
57 <Block name="HeaderSERVER" ref="Header">
58   <String name="Header" value="SERVER"/>
59   <String name="Value" value="OS/1.0 UPnP/1.1 product/1.0"/>
60 </Block>
61
62 <!-- USN: 01234567-89ab-cdef-0123-456789abcdef::upnp:rootdevice -->
63 <Block name="HeaderUSN">

```

```

64     <String name="Header" value="USN"/>
65     <String name="Value" value="01234567-89ab-cdef-0123-456789abcdef::upnp:rootdevice"/>
66 </Block>
67
68     <String value="¥r¥n"/>
69 </DataModel>
70
71 <StateModel name="State1" initialState="Initial">
72     <State name="Initial">
73         <Action type="output">
74             <DataModel ref="UpnpNOTIFYRequest"/>
75         </Action>
76     </State>
77 </StateModel>
78
79
80 <Test name="UpnpNOTIFYRequestTest" description="Upnp NOTIFY Request Test">
81     <StateModel ref="State1"/>
82     <Publisher class="udp.Udp">
83         <Param name="host" value="239.255.255.250" />
84         <Param name="port" value="1900" />
85     </Publisher>
86 </Test>
87
88 <Run name="DefaultRun" description="Upnp NOTIFY Request Run">
89     <Test ref="UpnpNOTIFYRequestTest" />
90     <Logger class="logger.Filesystem">
91         <Param name="path" value="c:¥peach¥log¥" />
92     </Logger>
93 </Run>
94 </Peach>

```

5.2.3 「Peach」を使ったファジング

「Peach」の設定ファイルが準備できたら、あとはそれらを使ってファジングを実施します。探索リクエストの設定ファイル「upnp_msearch.xml」と、広告リクエストの設定ファイル「upnp_notify.xml」ごとに、2回ファジングを実施します（図 21）。

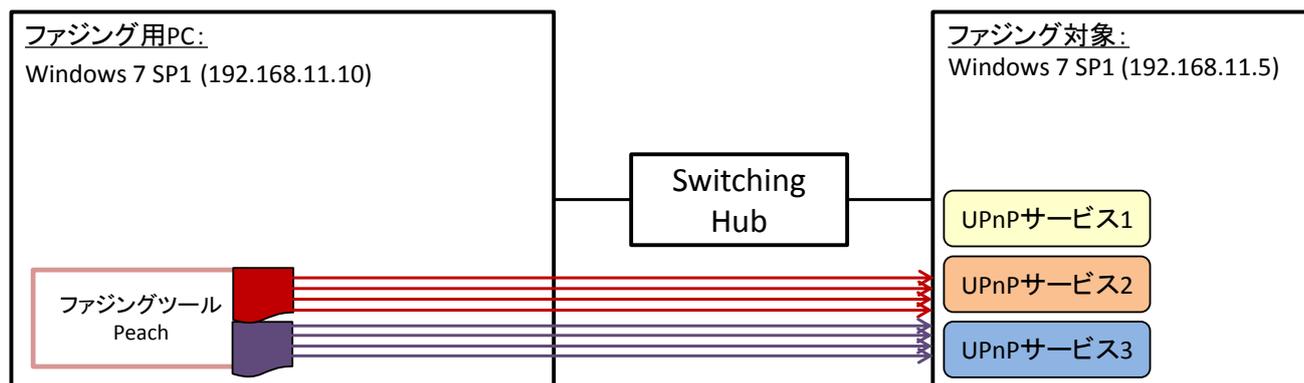


図 21 「Peach」を使ったファジング（イメージ図）

「Peach」を「C:¥peach」にインストールしていると仮定すると、探索リクエストの設定ファイル「upnp_msearch.xml」を指定して「Peach」を実行する場合、次のようになります。

```
C:¥peach> peach upnp_msearch.xml
```

広告リクエストの設定ファイル「upnp_notify.xml」を指定して「Peach」を実行する場合、次のようになります。

```
C:¥peach> peach upnp_notify.xml
```

以上で、【探索】における「Peach」を使ってファジングを実践できます。「Peach」の実行が完了したあとで、ファジング結果を分析する作業等がありますが、本手順書ではそれらの作業の説明を割愛します。

6 【制御】におけるファジング

UPnP 機能の【制御】に対しては、ファジングツール「Taof」または「Peach」を使ってファジングを実施します。このファジングでは、制御リクエストにおける制御データをテストデータに置き換えてファジング対象機器に送信します。

6.1 ファジングの流れ

UPnP 機能の【制御】におけるファジングは、大きく 4 つの作業に分かれます（図 22）。

【探索】におけるファジングの作業工程と比べると、ファジング対象とする UPnP サービスを特定する作業と、ファジングの基となるデータを生成する作業 2 つが追加されています。【制御】では、ファジング対象機器ごとに動作している UPnP サービスおよび制御リクエストが異なるため、まずテストデータを送信する UPnP サービスとファジングの基となる制御リクエストを準備する必要があります。

作業①、②の具体的な手順は「Taof」、「Peach」どちらも共通していますが、作業③、④の具体的な手順は「Taof」、「Peach」ごとに異なります。6.1.1 節から各工程について説明していきます。

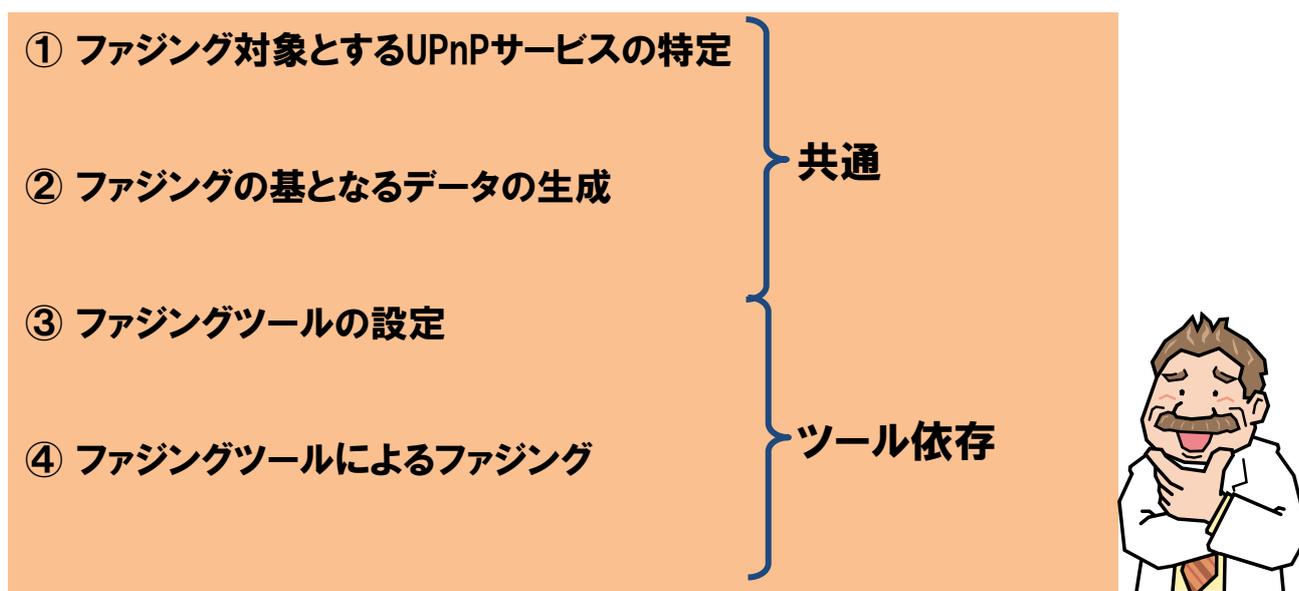


図 22 UPnP 機能の【制御】におけるファジングの作業工程

6.1.1 ファジング対象とする UPnP サービスの特定

まずファジング対象機器が提供している UPnP サービスを調べて、どの UPnP サービスに対してファジングを実施するかを決めます。

UPnP 機能を実装している機器では、複数の UPnP サービスを提供している場合があります。図 23 のファジング対象機器には「UPnP サービス 1」と「UPnP サービス 2」、「UPnP サービス 3」があることが分かります。

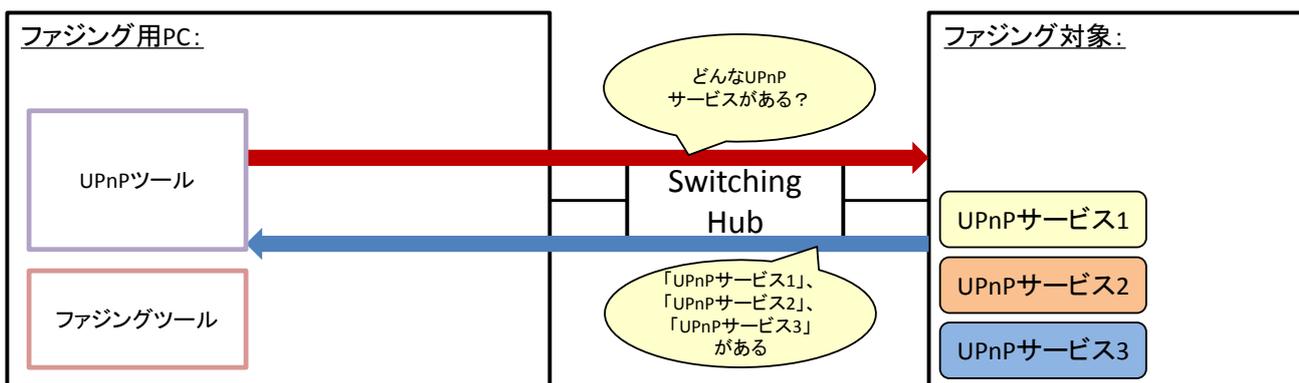


図 23 ファジング対象とする UPnP サービスの特定

6.1.2 ファジングの基となるデータの生成

ファジングを実施する UPnP サービスを決めたら、ファジングの基となるデータを生成します。このデータは、ファジング対象となる UPnP サービスに送信する制御リクエストとなります。

本手順書では、このデータを生成するためにファジング用 PC からファジング対象とする UPnP サービスに対して、実際に制御リクエストを送信します（図 24）。その制御リクエストを取得して、ファジングのテストデータとして使います。

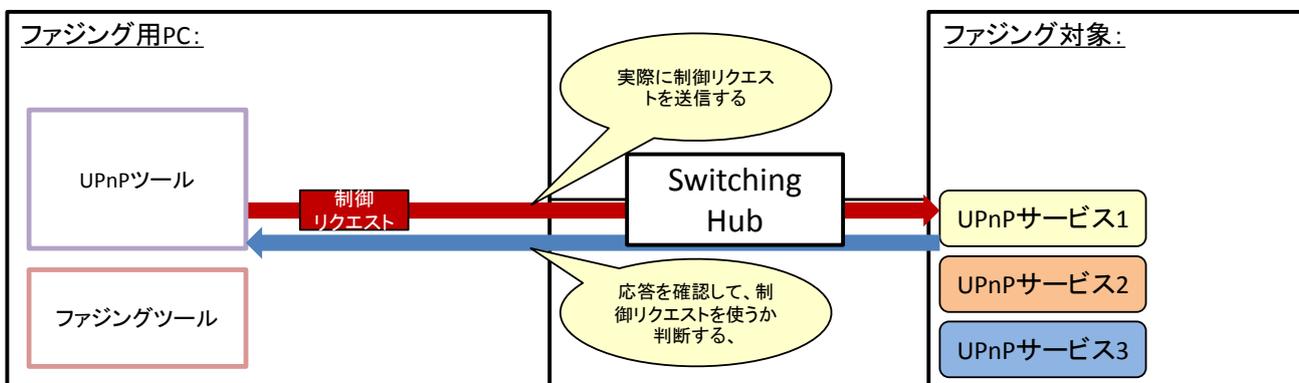


図 24 テストデータの基となるデータの生成

6.1.3 ファジングツールの設定

ファジングの基となる制御リクエストを生成したら、ファジングツール「Taof」、「Peach」ごとにそのデータを設定します（図 25）。

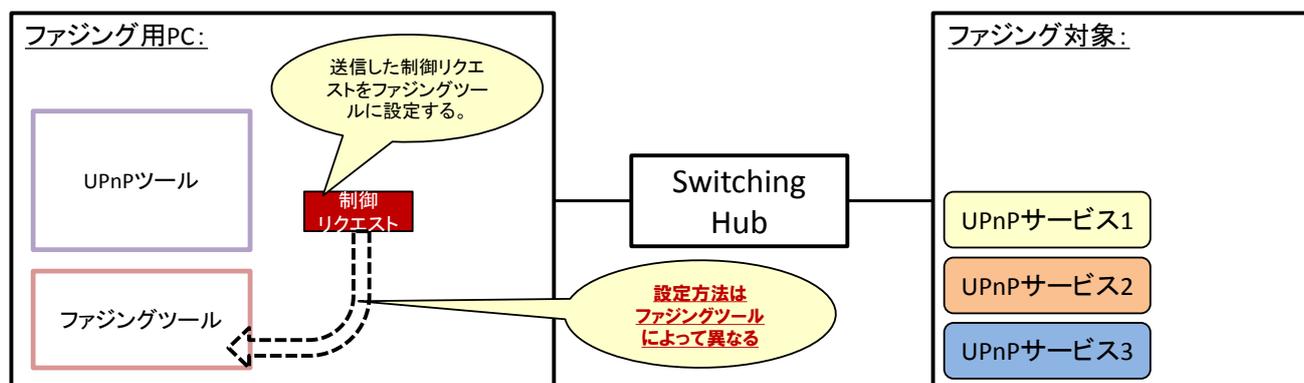


図 25 ファジングツールの設定

6.1.4 ファジングツールによるファジング

ファジングツールの設定が完了したら、あとはファジングを実践するだけです（図 26）。

ファジングツール「Taof」、「Peach」の使い方は、IPA 公開資料「ファジング実践資料¹⁸」で紹介しています。「Taof」、「Peach」に馴染みのない方は、その資料もご参照ください。

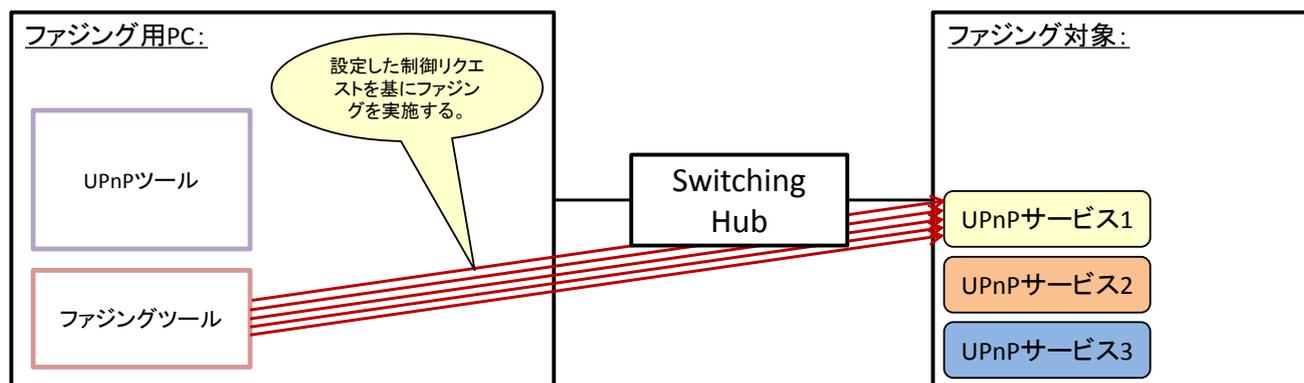


図 26 ファジングツールによるファジング

以上で、【制御】におけるファジングの流れの説明を終わります。この流れを基に、6.2 節からツールの具体的な操作を交えながら、ファジングの実践手順を説明していきます。

まずファジングツールに依存しない作業①、②の実践手順を 6.2 節で説明します。続いて、「Taof」の作業③、④を 6.3 節、「Peach」の作業③、④を 6.4 節で説明します。

¹⁸ IPA : 「ファジング実践資料」

<http://www.ipa.go.jp/security/vuln/documents/fuzzing-tool.pdf>

6.2 ファジング実践

本節から 6.4 節にかけて、「Windows 7 SP1」の UPnP サービスに対するファジングを例にとり、【制御】におけるファジング手順を説明します。

【制御】におけるファジングの作業①、②では、図 27 の 3 つの作業を実施します。この 3 つの作業を完了すると、続いてファジングツール「Taof」、「Peach」ごとにファジングを実践していきます。

- ① **ファジング対象とするUPnPサービスの特定**
 - (1) 「Device Spy」を使ったファジング対象とするUPnPサービスの特定
- ② **ファジングの基となるデータの生成**
 - (1) 「Device Spy」によるHTTPリクエストの送信
 - (2) 「Wireshark」によるHTTPリクエストの保存
- ③ **ファジングツールの設定**
- ④ **ファジングツールによるファジング**



図 27 作業①、②の作業項目

6.2.1 「Device Spy」を使ったファジング対象とする UPnP サービスの特定

【制御】におけるファジングでは、まずファジング対象機器で動作している UPnP サービスを調べて、それらの UPnP サービスのうち、どれをファジング対象とするか決定します。

この作業のイメージは図 28 の通りです。「Device Spy」を使って、ファジング対象機器から UPnP サービス一覧を取得します。「Device Spy」が整形して GUI 上に表示した UPnP サービスの一覧から、ファジング対象とするものを決定します。

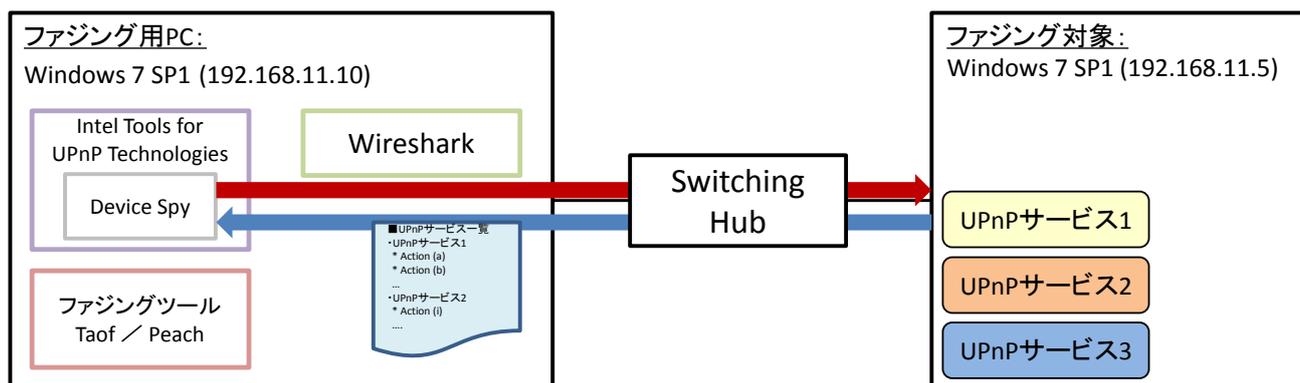


図 28 UPnP サービスの特定 (イメージ図)

それでは実際に「Device Spy」を起動して UPnP サービスを調べてみます。[スタート]メニューから[すべてのプログラム]→[Intel(R) Tools for UPnP(TM) Techonlog]→[Device Spy]を起動します (図 29)。



図 29 [スタート]メニュー

「Device Spy」を起動すると、図 30 のウインドウが立ち上がります。「Device Spy」は探索リクエスト、問い合わせリクエストを送信して、同じネットワーク上に存在する「UPnP 機能を実装した機器」を GUI 上に表示します (図 30)。図 30 の左ペインに「Win7SP1: fuzzer」というアイコンがありますが、これがファジング環境における Windows 7 SP1 (192.168.11.5) に該当します (図 30 の赤枠部分)。

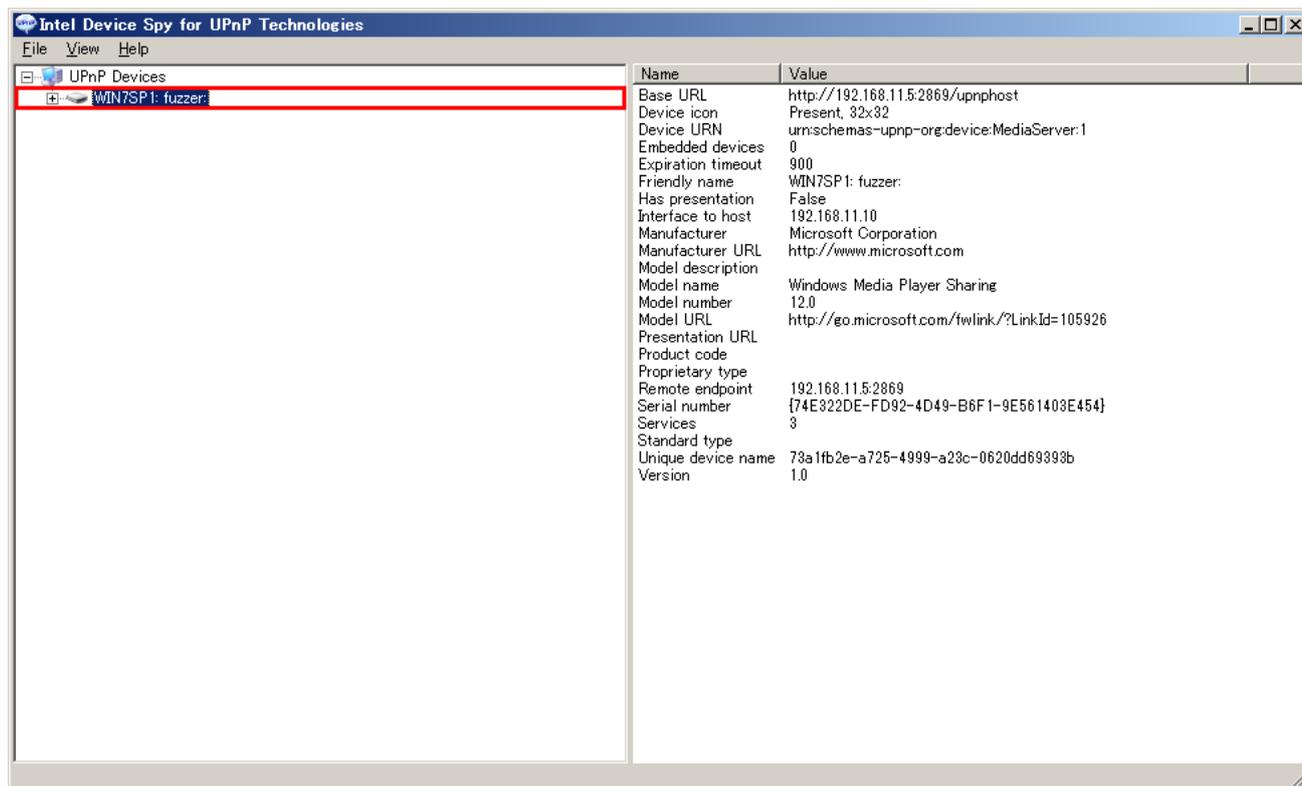


図 30 「Device Spy」

図 30 の「WIN7SP1: fuzzer」の[+]をクリックして展開すると、図 31 のように表示されます。「urn:●●●:service:■ ■ ■」という項目が UPnP サービスに該当します。図 31 では、次の 3 つの UPnP サービスを確認できました (図 31 の赤枠部分)。

- urn:microsoft.com:service:X_MS_MediaReceiverRegistrar:1
- urn:schemas-upnp-org:service:ConnectionManager:1
- urn:schemas-upnp-org:service:ContentDirectory:1

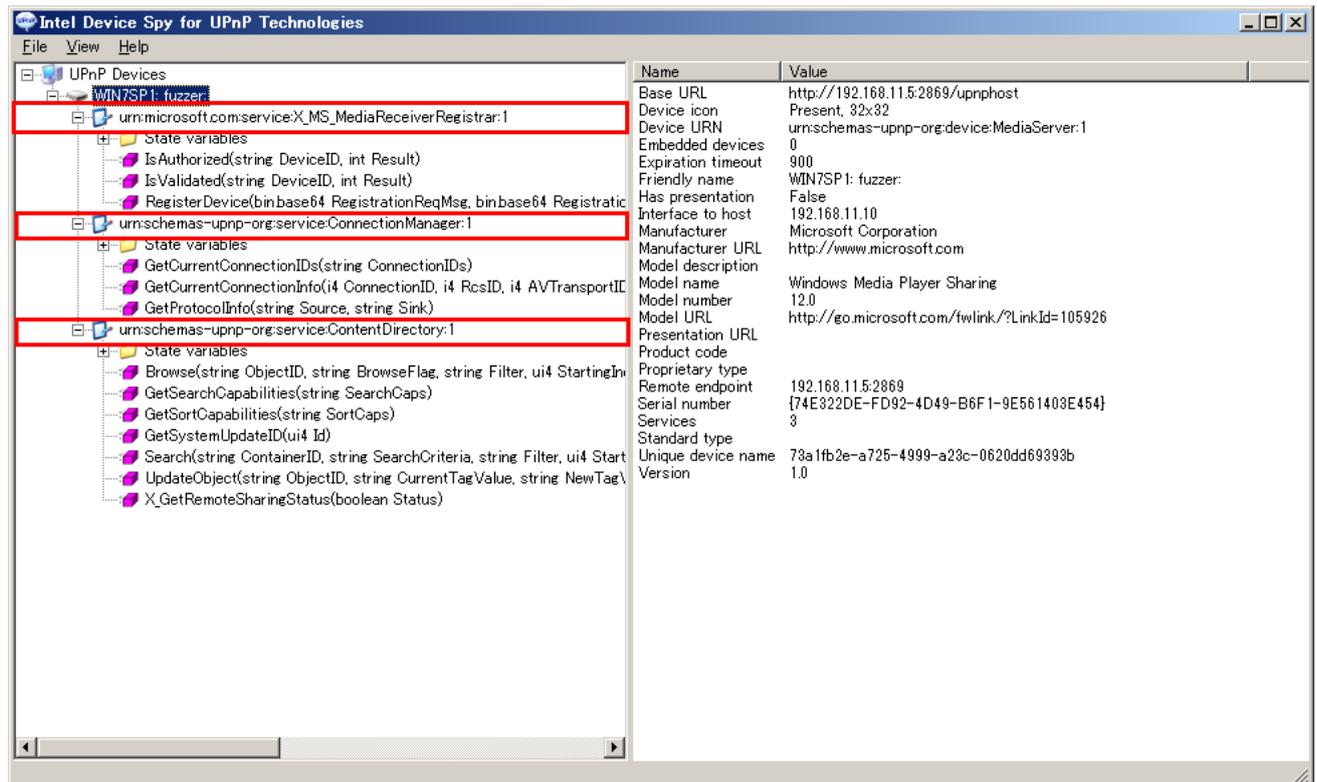


図 31 「Device Spy」による UPnP サービスの調査結果

本手順では、「urn:microsoft.com:service:X_MS_MediaReceiverRegistrar:1」(以降、MediaReceiver Registrar サービスと呼称) に対してファジングを実施することとします。

6.2.2 補助ツールを使ったファジングの基となるデータの生成

ファジング対象とする UPnP サービスを決定したら、ファジングの基となるデータを生成します。この作業には「Device Spy」と「Wireshark」を使います。この作業のイメージは図 32 の通りです。

まず「Device Spy」でファジング対象とする UPnP サービスに送信する制御リクエストを指定して、実際に制御リクエストを送信します。そのとき、「Wireshark」で送信した制御リクエストの packets を取得して、ファイルに保存します。

6.3 節の「Taof」によるファジング実践および 6.4 節の「Peach」によるファジング実践では、この作業で保存したファイルをファジングツールに設定します。

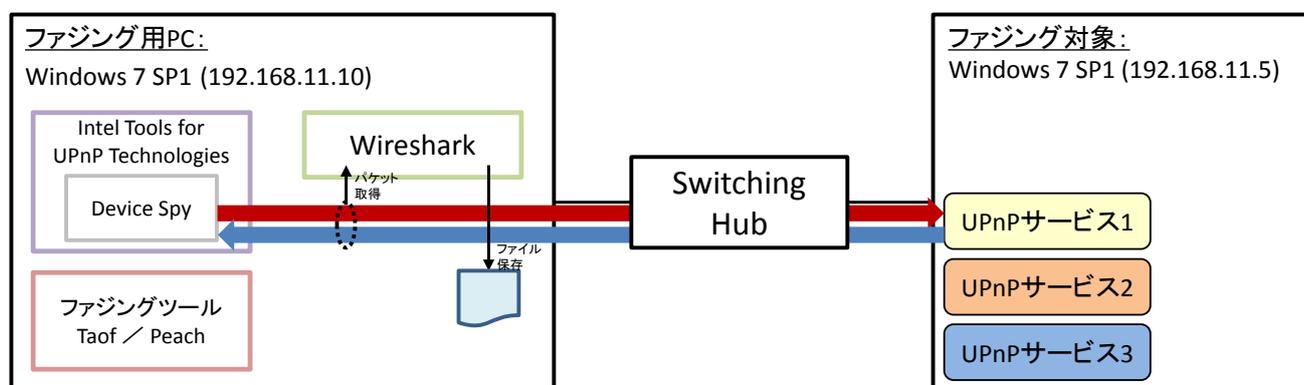


図 32 UPnP リクエストの生成・取得 (イメージ図)

それでは、実際にファジングの基となるデータを生成してみます。この作業は、「『Device Spy』による制御リクエストの送信」、「『Wireshark』による制御リクエストの保存」の順に説明します。

◆ 「Device Spy」による制御リクエストの送信

まずファジング対象とする MediaReceiverRegistrar サービスに送信する制御リクエストを決めます。図 33 において、MediaReceiverRegistrar サービスが受け付ける制御リクエストは次の 3 つが該当します (図 33 の赤枠部分)。この 3 つの制御リクエストのうち、「IsAuthorized(...)」をテストデータとして使用します。

- IsAuthorized(...)
- IsValidated(...)
- RegisterDevice(...)

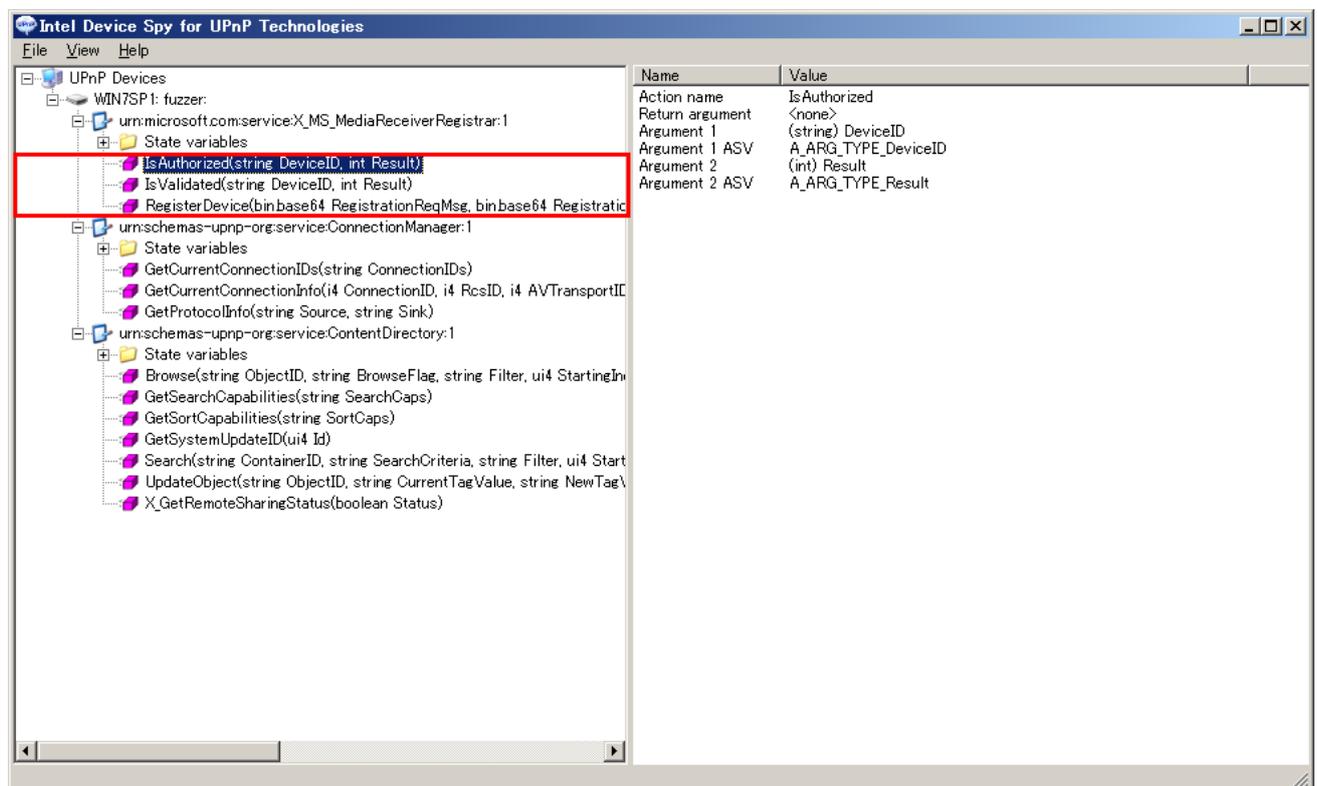


図 33 「Device Spy」による UPnP サービスの調査結果 (図 31 再掲)

図 33 左ペインの「IsAuthorized」をダブルクリックすると、図 34 のウインドウが開きます。「Device ID」のテキストフィールドに値を入力して[Invoke]ボタンをクリックすると、MediaReceiverRegistrar サービスに対して、制御リクエスト「IsAuthorized」を送信できます。

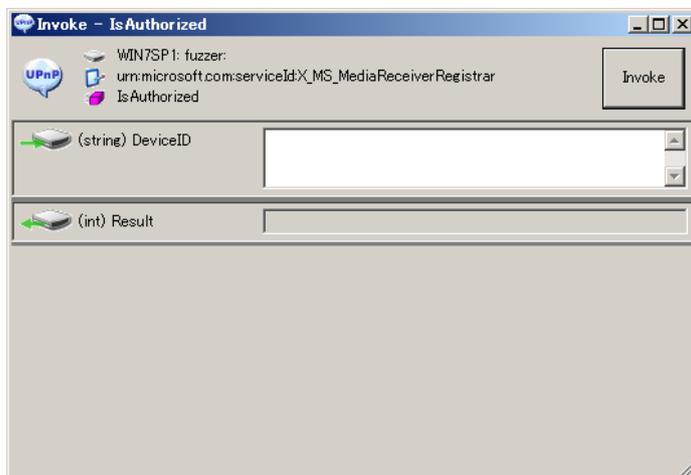


図 34 「Device Spy」 - 「IsAuthroized(...)」

★ 「Wireshark」による HTTP リクエストの保存

実際に「Device Spy」で制御リクエスト「IsAuthorized」を送信して、「Wireshark」でその制御リクエストをテキストファイルに保存します。

図 34 の[Invoke]ボタンをクリックして制御リクエストを送信する前に、まず「Wireshark」を起動してパケットキャプチャを開始しておきます。パケットキャプチャを開始したら、制御リクエストを送信します。本手順では、「Device ID」に「AAAAA」という値を入力して[Invoke]ボタンをクリックします(図 35)。本手順書では「AAAAA」という値を入力しましたが、ファジング時に「Device ID」の値をテストデータに置換するため、「Device ID」の値はどんな値でも構いません。

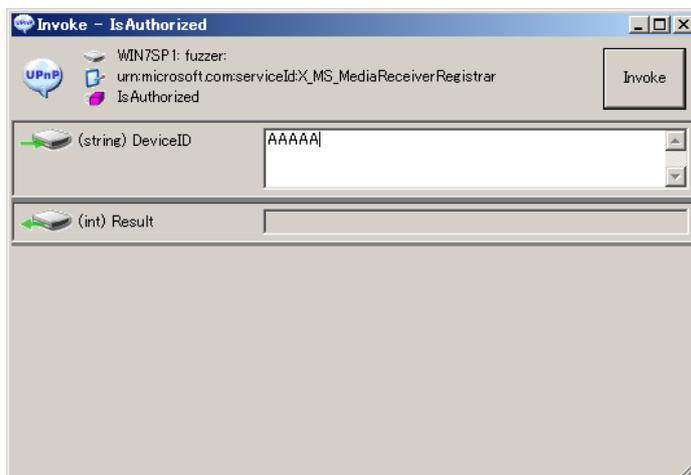


図 35 制御リクエスト「IsAuthorized(...)」の送信

図 35 で制御リクエストを送信すると、[Invocation Error Code 501 (187ms): Action Failed]というエラーが生じました (図 36 の赤枠部分)。このとき、送信した制御リクエストを「Wireshark」で確認してみます。

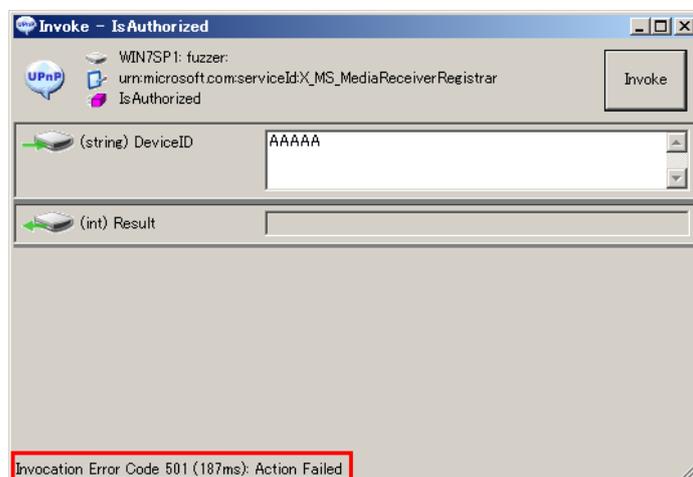


図 36 制御リクエスト「IsAuthorized(...)」の送信結果

図 35 で送信した制御リクエストを「Wireshark」で表示すると、図 37 のようになります¹⁹。図 37 の赤字部分に「AAAAA」という文字列が送信されていることが分かります。

図 36 にて[Invocation Error Code 501 (187ms): Action Failed]エラーが生じましたが、HTTP レスポンスを確認すると、UPnP における HTTP レスポンスの形式に則ってエラーメッセージが応答されています。この HTTP レスポンスから、ファジング対象とする UPnP 機能は制御リクエスト自体を処理していると考えます。

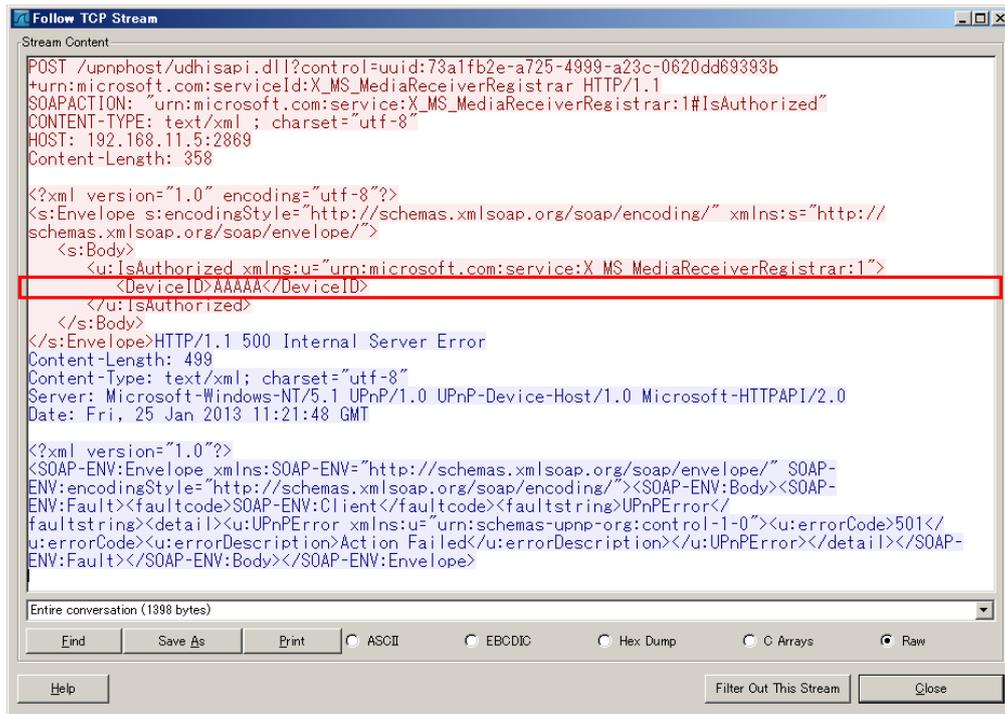


図 37 制御リクエスト「IsAuthorized」（図 35 で送信したもの）

以上の確認結果をふまえて、図 37 における制御リクエスト（[Follow TCP Stream]の赤字部分）をファジングの基となるデータに使用することとします。

¹⁹ 「Wireshark」にて該当制御リクエストのパケットを選び、マウス右クリックにて[Follow TCP Stream]を実行した結果です。

最後にファジングの基となる制御リクエストをテキストファイルに保存します。

図 37 を図 38 のように変更して（変更部分は図 38 の赤枠部分）、制御リクエストだけ抽出します。図 38 のように制御リクエストだけ抽出したら、[Save As]ボタンをクリックして制御リクエストをテキストファイルに保存します。本手順では、この制御リクエストを「C:\Yupnfuzz\httprequest.txt」で保存することとします。

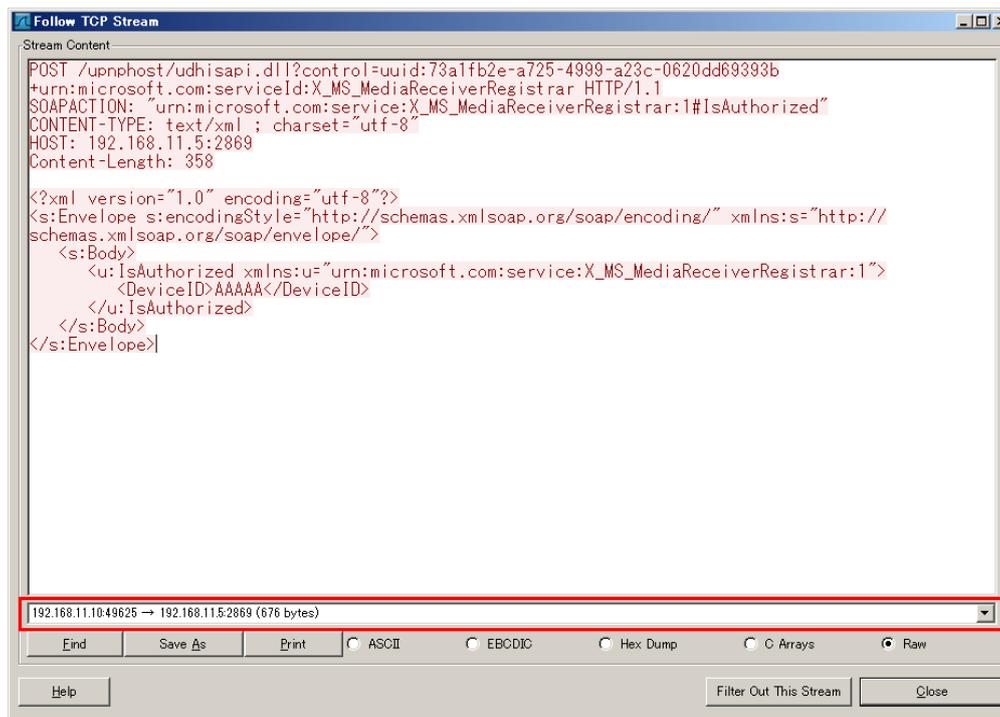


図 38 制御リクエスト「IsAuthorized」のみ抽出

以上で【制御】におけるファジングの作業①、②が完了となります。続いて、「Taof」と「Peach」ごとに「C:\Yupnfuzz\httprequest.txt」を設定して、ファジングを実施していきます。

6.3 ファジング実践（「Taof」編）

本節では、6.2.2 節で準備したデータ「C:\Yupnpfuzz\httprequest.txt」を「Taof」に設定して、それでファジングを実践していきます。「Taof」による【制御】のファジングでは、図 39 の 2 つの作業を実施します。

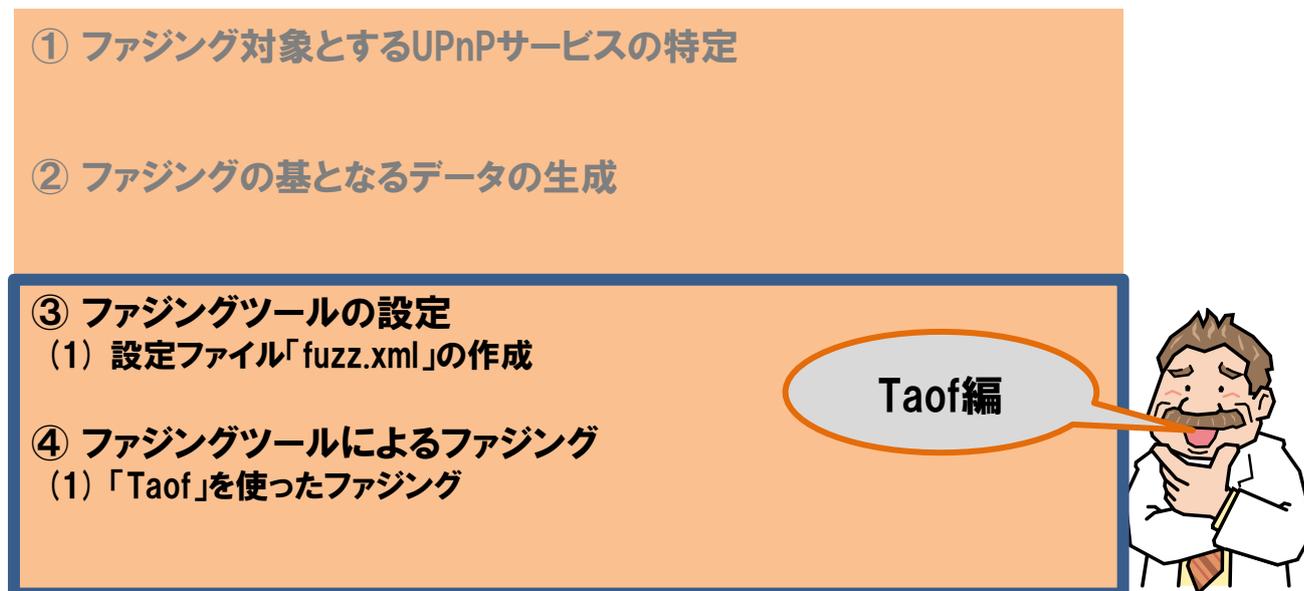


図 39 「Taof」による作業③、④の作業

6.3.1 「Taof」の設定

まずファジングを実施するために、「Taof」の設定ファイル群を作成します。「Taof」の設定ファイル群は、設定ファイル（XML ファイル）とリクエストファイル（テキストファイル）の 2 種類のファイルで構成されます（図 40）。

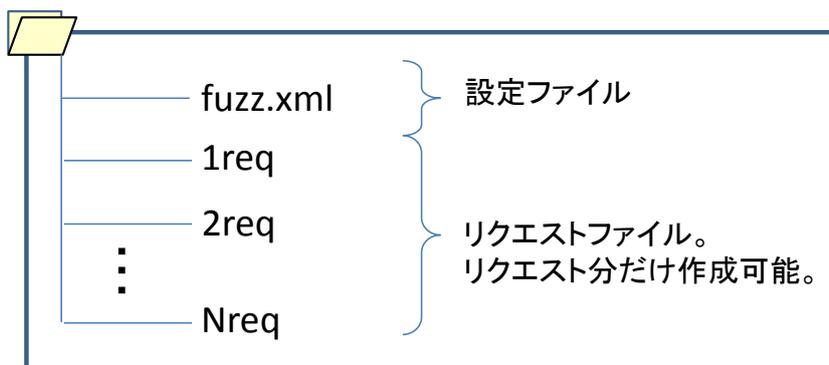


図 40 「Taof」の設定ファイル群

本手順書では、フォルダ「C:\¥unpnpfuzz¥」に次の2つのファイルを作成することで、ファジングの準備が完了となります。

- fuzz.xml (設定ファイル)
- lreq (リクエストファイル)

リクエストファイル「lreq」については、6.2.2 節で準備した「httprequest.txt」のファイル名を「lreq」に変更して保存するだけです。本手順で使用される HTTP リクエストファイルは1つだけですが、複数指定することもできます。

続いて、設定ファイル「fuzz.xml」を作成します。「fuzz.xml」を作成する前に、設定ファイルの概要を説明します。その後、今回のファジングにおける設定ファイル「fuzz.xml」を作成します。

※ 「Taof」の設定ファイル

「Taof」の設定ファイルは、XML で構成されます。「Taof」の設定ファイルを図 41 に例示します。図 41 は、ある設定ファイルをウェブブラウザ「Internet Explorer 9」で閲覧したものです。この設定ファイル例を基に設定ファイルの XML 構成を説明します。

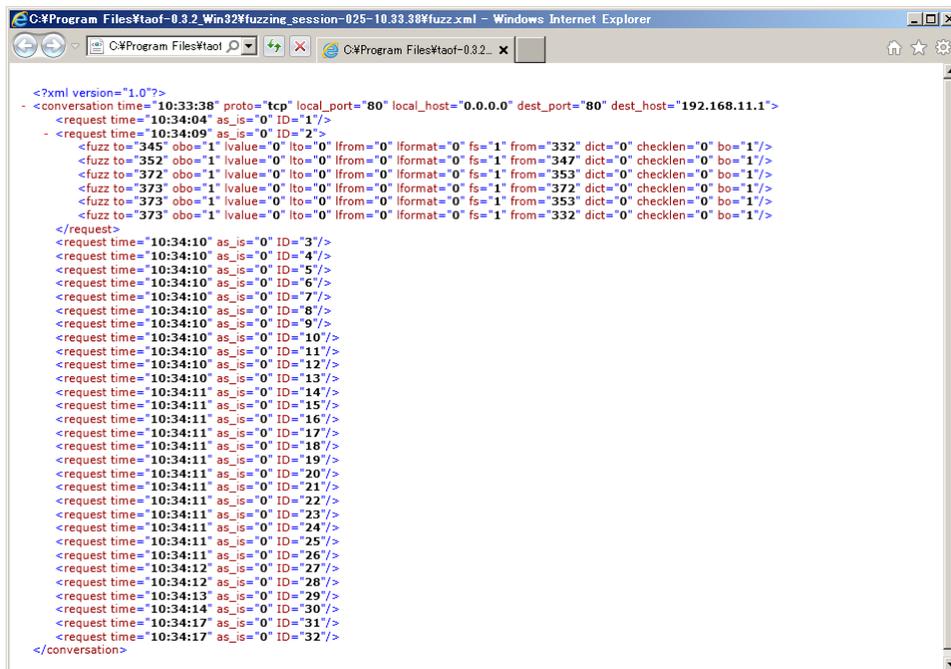


図 41 「Taof」の設定ファイルの例

「Taof」の設定ファイルは、表 7 の要素で構成されます。「Taof」自ら収集したリクエストデータを基にファジングを実施する場合、この設定ファイルを意識する必要はありません（自動的に作成されるため）。

しかし、「Taof」では設定ファイルとリクエストファイルを準備して、それらを基にファジングを実施することもできます。本手順では、この方法を使用するため手動で設定ファイルを作成します。

表 7 設定ファイルの構成要素

要素	説明
xml 要素	XML のヘッダ情報
conversation 要素	「Taof」と「ファジング対象機器」の通信情報 (IP アドレスやポート番号など)
request 要素	ファジングの基となるリクエストファイルの情報。 リクエストファイルの数だけ request 要素を作成する。
fuzz 要素	リクエストファイルごとのファジングポイントの情報。 リクエストファイルごとにテストデータに置換する箇所の数だけ作成する

● 設定ファイル「fuzz.xml」の作成

本節では、下記のテンプレートを使って設定ファイル「fuzz.xml」を作成します。テンプレートの●1、●2、●3には、表 8 の[本手順書の設定値]を入力します。

```
<?xml version="1.0" ?>
<conversation dest_host="●1" dest_port="●2" local_host="0.0.0.0" local_port="80" proto="●3" time="01:01:01">
<request ID="1" as_is="0" time="01:01:01"/>
</conversation>
```

表 8 fuzz.xml の設定値

項目	本手順書の設定値	設定値
●1	192.168.11.5	ファジング対象機器の IP アドレス 本手順では、図 14 右側の「Windows 7 SP1」の IP アドレス。
●2	2869	ファジング対象機器のポート番号を設定する。 本手順では、ファジング対象とする UPnP サービスのポート番号。
●3	tcp	ファジングにおける通信プロトコルを設定する。 本手順では、「tcp」。

本節における「Taof」の設定ファイル fuzz.xml は下記となります。この設定ファイルの構成要素のうち、表 7 で説明していない要素および属性について補足します。

```
<?xml version="1.0" ?>
<conversation dest_host="192.168.11.5" dest_port="2869" local_host="0.0.0.0" local_port="80" proto="tcp" time="01:01:01">
<request ID="1" as_is="0" time="01:01:01"/>
</conversation>
```

conversation 要素の local_host 属性および local_port 属性については、テンプレートの値のまま構いません。これらの属性は、「Taof」がリクエストデータを収集するときの待ち受け IP アドレスおよびポート番号となります。本手順では、手動で設定ファイルを作成しているため、どんな値でも構いません。

request 要素の ID 属性には、リクエストファイルのファイル名「●req」の●（数値）を設定します。本手順で使用するリクエストファイルは「1req」であるため、request 要素の ID 属性には「1」を設定します。もしリクエストファイルを複数準備する場合、そのリクエストファイル分だけ request 要素を作成します。

本節の手順が完了すると、「C:\¥upnpfuzz」フォルダの内容は図 42 のようになります。

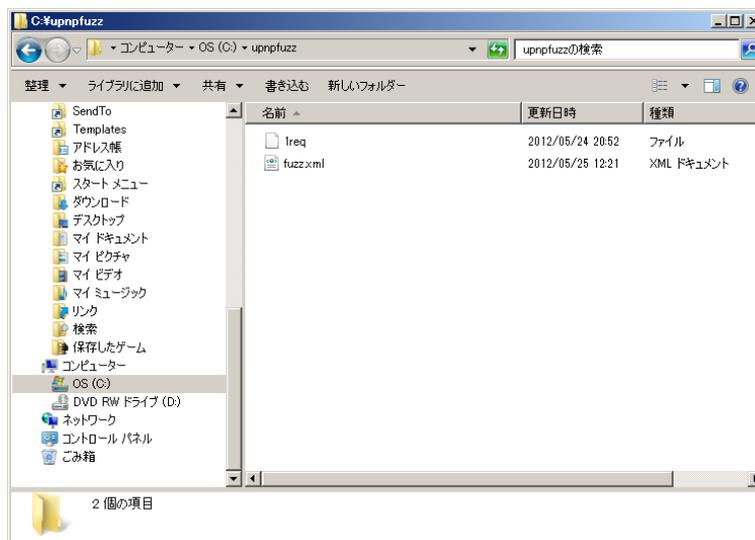


図 42 準備した「Taof」の設定フォルダ

6.3.2 「Taof」を使ったファジング

「Taof」の設定ファイル群を準備できたら、あとは「Taof」を使ってファジングを実施します。「Taof」を使ったファジング手順については、「ファジング実践資料²⁰」で紹介していますので、本手順書では割愛します。必要に応じてご参照ください。

「Taof」を起動したら、[File]メニューの[Open]を実行します（図 43）。

[Open]を実行すると、ファイル選択画面が表示されます。その画面では 6.3.2 節で準備した「C:\¥upnpfuzz¥fuzz.xml」を選択します。設定ファイル「fuzz.xml」を読み込んだあとで、図 44 のように[Request Contents]に 6.2.2 節で保存したリクエストファイルの内容が表示されていれば、問題ありません。

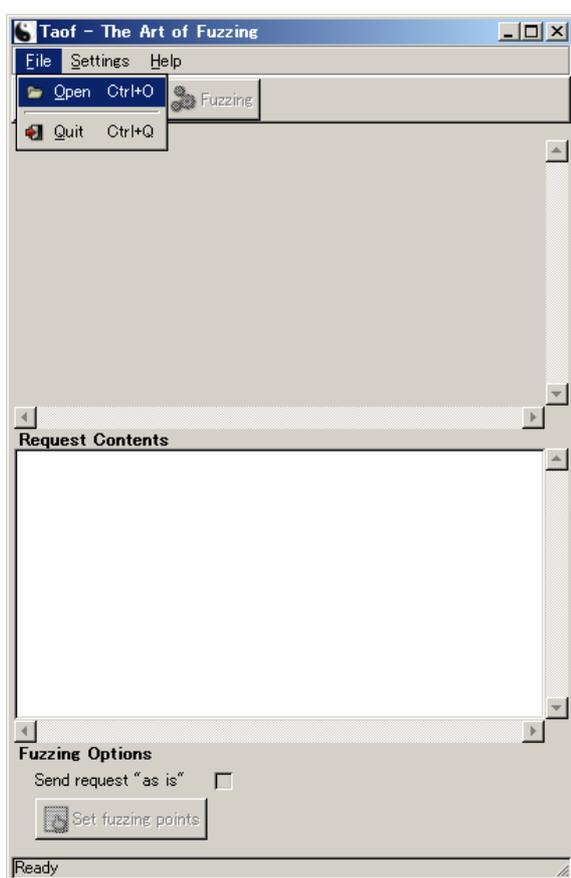


図 43 [Open]メニュー

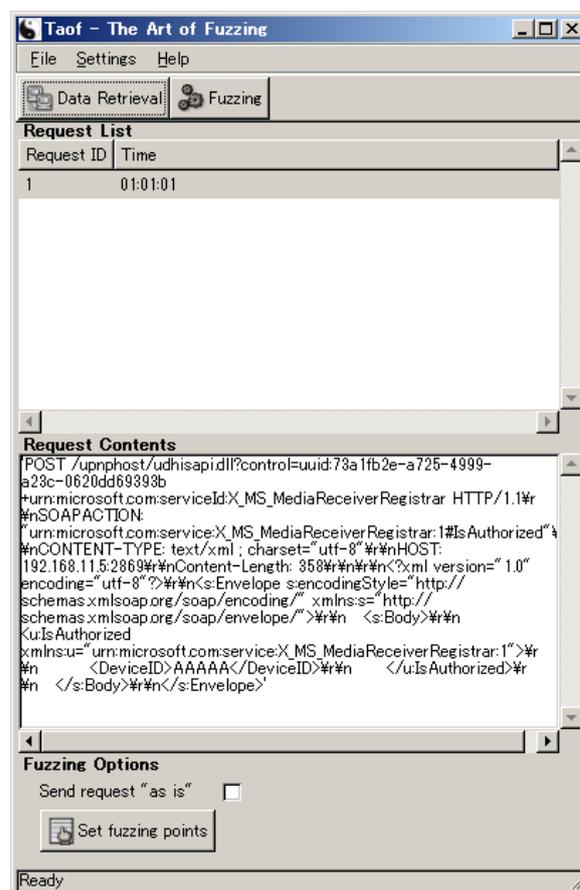


図 44 リクエストファイルの読み込み

²⁰ IPA : 「ファジング実践資料」

<http://www.ipa.go.jp/security/vuln/documents/fuzzing-tool.pdf>

あとは、制御リクエスト送信時に「Device Spy」で設定した値（DeviceID 要素の「AAAAA」）をファジングポイントに設定します（図 45 の赤枠部分）。その値をテストデータに置き換えることで、ファジングを実施します。

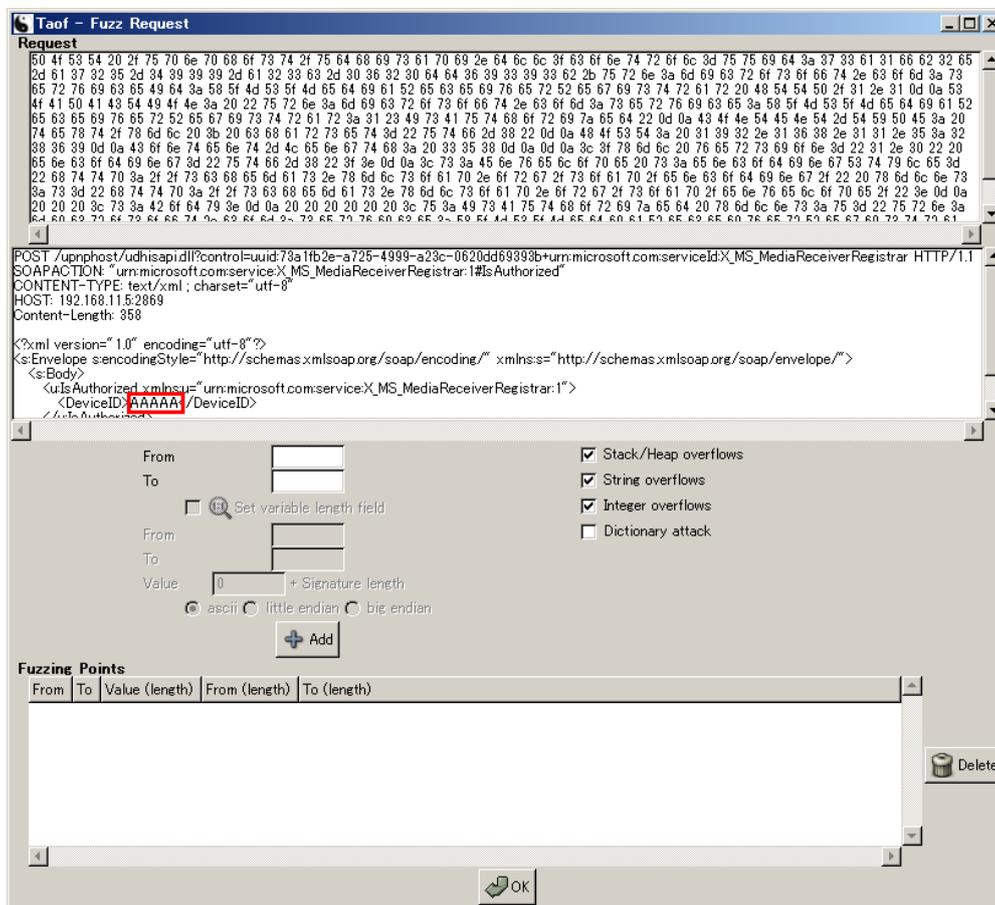


図 45 「Taof」におけるファジングポイント

以上で、【制御】における「Taof」を使ってファジングを実践できます。「Taof」の実行が完了したあとで、ファジング結果を分析する作業等がありますが、本手順書ではそれらの作業の説明を割愛します。

6.4 ファジング実践（「Peach」編）

本節では、6.2.2 節で準備したデータ「C:\upnpfuzz\httprequest.txt」を基に、「Peach」が使用する設定ファイルを作成して、「Peach」によるファジングを実施します。

「Peach」による【制御】のファジング実践は、「ファジングツールの設定」と「ファジングツールによるファジング」の2つの工程に分かれます。この2つのうち、「ファジングツールの設定」についてはさらに2つの大きな作業に分けることができます（図 46）。

6.4.1 節から順に図 46 の作業項目を説明していきますが、その前に本節で使用する「テンプレート設定ファイル」について説明します。

① ファジング対象とするUPnPサービスの特定

② ファジングの基となるデータの生成

③ ファジングツールの設定

(1) 設定ファイル用リクエストデータ作成

- ・ [制御リクエストの変換]
- ・ [変換データの検索]
- ・ [変換データの加工]

(2) 設定ファイルの作成

- ・ [リクエストデータの記述]
- ・ [機器パラメータの記述]

④ ファジングツールによるファジング

(1) 「Peach」を使ったファジング

Peach編

図 46 「Peach」による作業③、④の作業

◆ テンプレート設定ファイルについて

本実践手順では【制御】におけるファジングのためのテンプレート設定ファイルを準備しています。本節では、テンプレート設定ファイルに制御リクエストと、ファジング対象機器パラメータ（IP アドレス、ポート番号）を記述して、設定ファイルとして保存します。

図 47 に示すように、テンプレート設定ファイルの **DataModel** の定義部分（水色部分と橙色部分）と **Test** の定義部分（緑色部分）に、制御リクエストと機器パラメータを記述（置換、貼り付け）します。その他の定義部分については、基本的に変更不要です。

表 9 にテンプレート設定ファイルを示します。どの部分を記述するのかイメージをつかんでください。

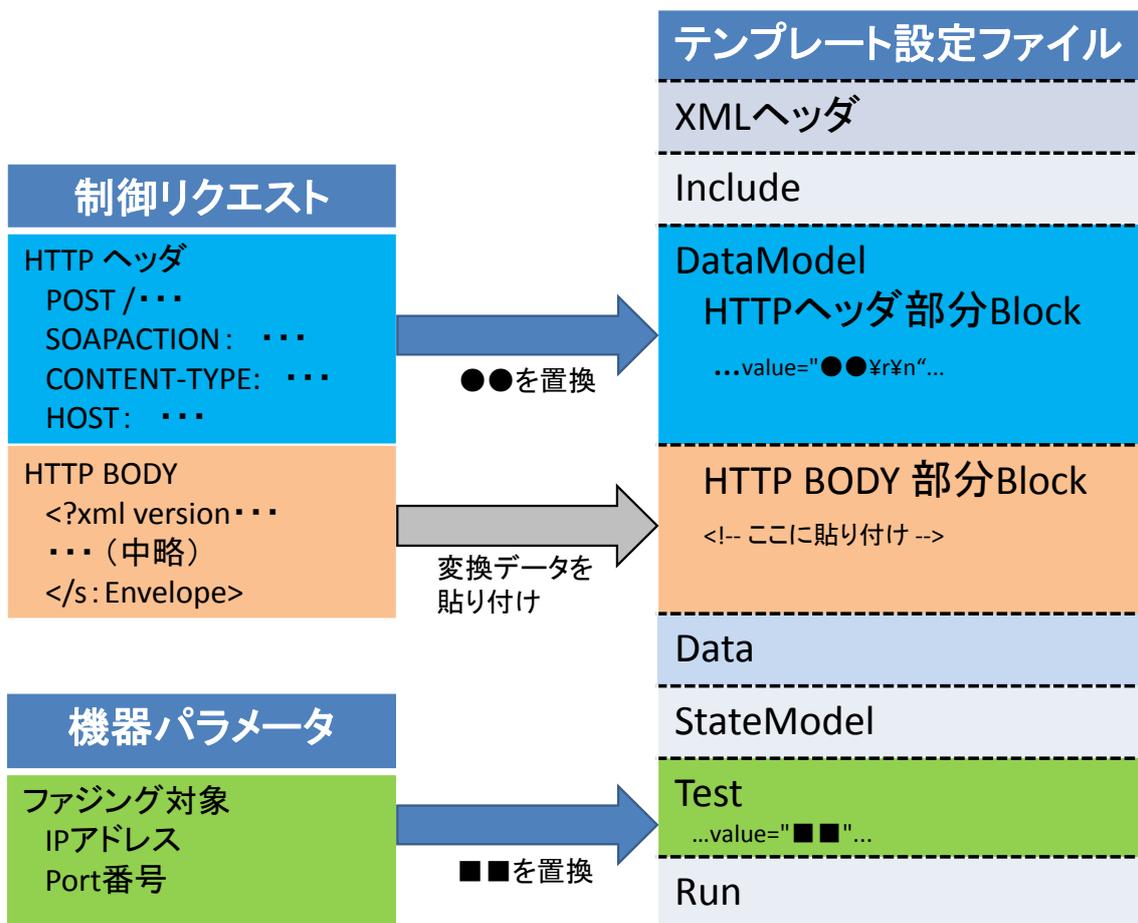


図 47 テンプレート設定ファイルの使い方

表 9 テンプレート設定ファイル「upnp_ctr_template.xml」

1	<?xml version="1.0" encoding="utf-8"?>
2	<Peach xmlns="http://phed.org/2008/Peach" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://phed.org/2008/Peach/peach.xsd" version="1.0" author="IPA" description="UPNP control">
3	<Include ns="default" src="file:defaults.xml" />
4	<DataModel name="UpnpRequest">
5	<Block name="RequestLine">
6	<String name="Method" value="●●¥r¥n" mutable="false" />
7	</Block>
8	<Block name="SOAPAction">
9	<String name="ACTION" value="●●¥r¥n" mutable="false" />
10	</Block>
11	<Block name="CONTENTStype">
12	<String name="CON-TYPE" value="●●¥r¥n" mutable="false" />
13	</Block>
14	<Block name="HeaderHost">
15	<String name="Host" value="●●¥r¥n" mutable="false" />
16	</Block>
17	<Block name="HeaderContentLength">
18	<String name="Header" value="Content-Length: " mutable="false" />
19	<String name="Value" mutable="false">
20	<Relation type="size" of="X_BODY" />
21	</String>
22	</Block>
23	<String value="¥r¥n¥r¥n" mutable="false" />
24	<Block name="X_BODY">
25	<!-- ここに貼り付け -->
26	</Block>
27	</DataModel>
28	<StateModel name="State1" initialState="Initial">
29	<State name="Initial">
30	<Action type="output">
31	<DataModel ref="UpnpRequest" />
32	</Action>
33	</State>
34	</StateModel>
35	<Test name="UpnpPostRequestTest" description="Upnp Request Post Test">
36	<StateModel ref="State1" />
37	<Publisher class="tcp.Tcp">
38	<Param name="host" value="■■" />
39	<Param name="port" value="■■" />
40	</Publisher>
41	</Test>
42	<Run name="DefaultRun" description="Upnp Request Run">
43	<Test ref="UpnpPostRequestTest" />
44	<Logger class="logger.Filesystem">
45	<Param name="path" value="c:¥peach¥log¥" />
46	</Logger>
47	</Run>
48	</Peach>

6.4.1 「Peach」の設定

ファジングツール「Peach」の設定は、「設定ファイル用リクエストデータ作成」と「設定ファイルの作成」の2つの作業に分かれます（図 48）。

まず「設定ファイル用リクエストデータ作成」にて、6.2.2 節で取得した制御リクエストのうち BODY 部分を「Peach」で読める書式に変換します。またこのとき、テストデータに置き換える箇所も指定します。この作業を完了すると、設定ファイルに記述するリクエストデータを準備できます。

そして「設定ファイル作成」にて、制御リクエストの HTTP ヘッダ、作成したリクエストデータ、機器パラメータをテンプレート設定ファイルに記述して、設定ファイルを作成します。

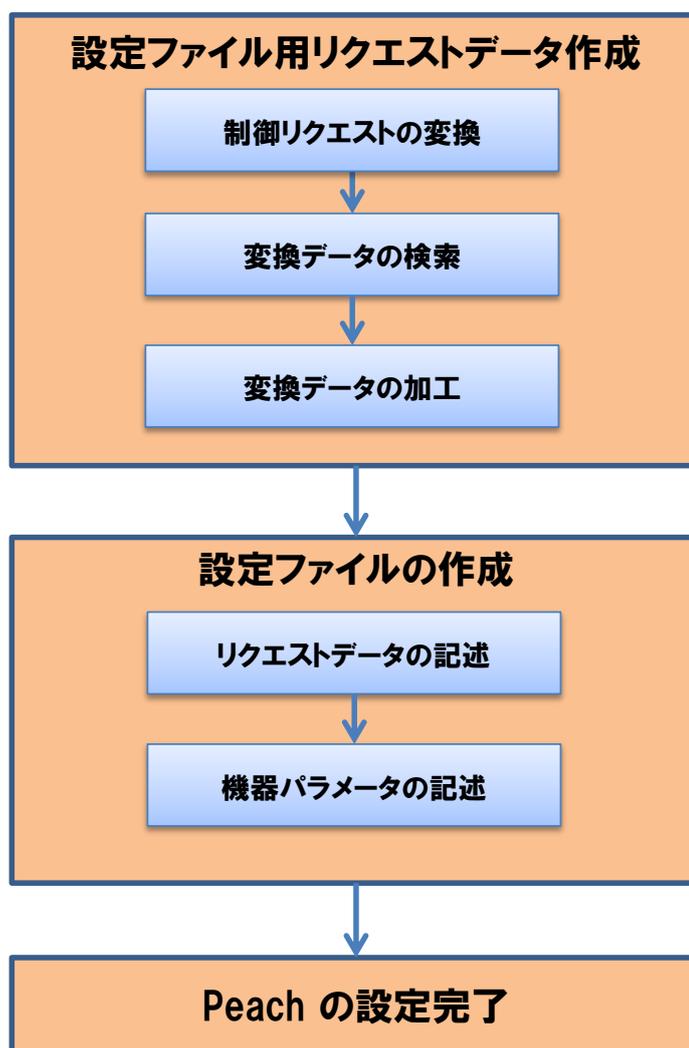


図 48 「Peach」の設定の流れ

※ 設定ファイル用リクエストデータ作成

本作業の流れは、図 49 になります。制御リクエストの BODY 部分を Peach で読める書式に変換して、【制御】におけるファジングで必要な部分のみ検索して抽出します。その取り出したデータにてテストデータに置換する箇所を指定したら設定ファイル用リクエストデータの作成が完了です。

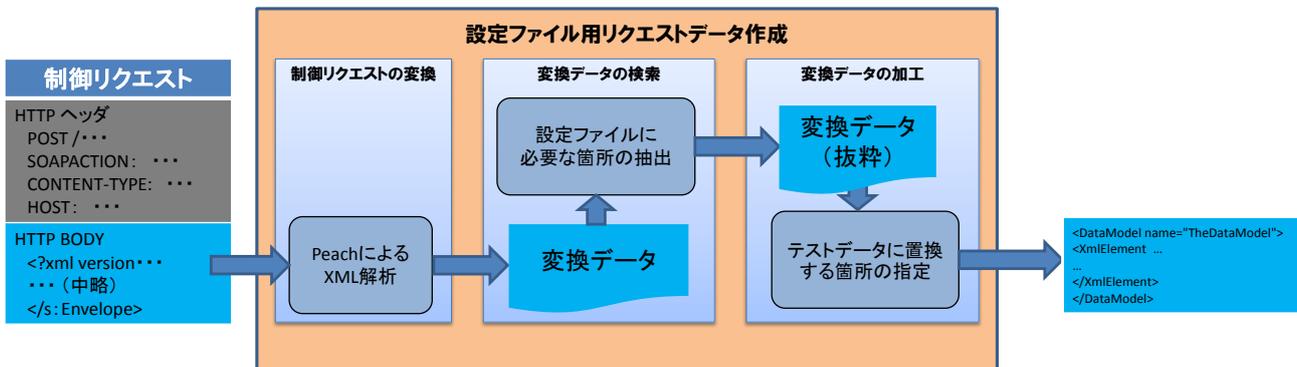


図 49 設定ファイル用リクエスト作成の流れ

[制御リクエストの変換]

制御リクエストの BODY 部分（図 50 の赤枠部分）を、変換用入力ファイル「upnp_body.xml」として保存します。

```
POST /upnp/udhisapi.dll?control=uuid:73a1fb2e-a725-4999-a23c-0620dd69393b
+urn:microsoft.com:serviceId:X_MS_MediaReceiverRegistrar HTTP/1.1
SOAPACTION: "urn:microsoft.com:service:X_MS_MediaReceiverRegistrar:1#IsAuthorized"
CONTENT-TYPE: text/xml ; charset="utf-8"
HOST: 192.168.11.5:2869
Content-Length: 358

<?xml version="1.0" encoding="utf-8"?>
<s:Envelope s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <u:IsAuthorized xmlns:u="urn:microsoft.com:service:X_MS_MediaReceiverRegistrar:1">
      <DeviceID>AAAAA</DeviceID>
    </u:IsAuthorized>
  </s:Body>
</s:Envelope>
```

図 50 制御リクエストの BODY 部分

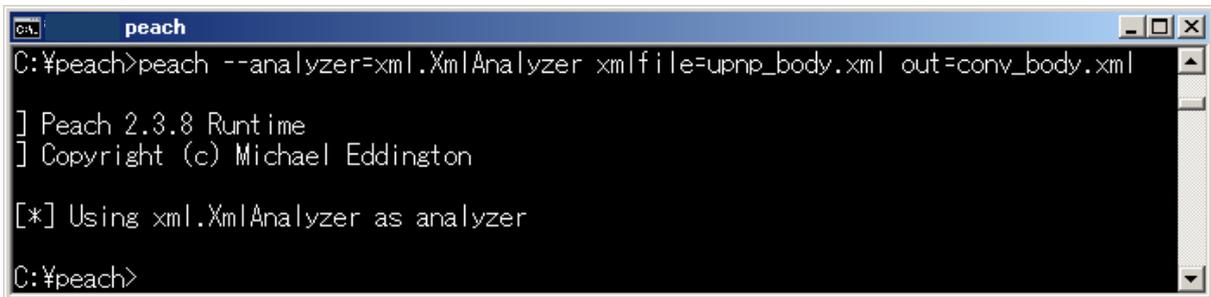
この「upnp_body.xml」を、「Peach」の XML 解析機能を使って「Peach」で読める書式に変換します。「Peach」の XML 解析機能については、「付録 2 : Peach の XML 解析による設定ファイルの出力」を参照してください。

Peach インストールフォルダ（本手順では C:\peach）に「upnp_body.xml」をコピーします。インストールフォルダに移動して、コマンドプロンプトを起動します。

コマンドプロンプト画面で「Peach」による XML 解析を実行します（図 51）。XML の解析引数（--analyzer=）には「xml.XmlAnalyzer」、入力ファイル名（xmlfile=）は「upnp_body.xml」、出力ファイル名（out=）は「conv_body.xml」を指定します。

XML 解析実行コマンド：

```
peach --analyzer=xml.XmlAnalyzer xmlfile=upnp_body.xml out=conv_body.xml
```



```
C:\peach>peach --analyzer=xml.XmlAnalyzer xmlfile=upnp_body.xml out=conv_body.xml
] Peach 2.3.8 Runtime
] Copyright (c) Michael Eddington
[*] Using xml.XmlAnalyzer as analyzer
C:\peach>
```

図 51 XML 解析機能の実行画面

Peach インストールフォルダに、「conv_body.xml」が作成されたら、次の作業に進みます。

[変換データの検索]

BODY 部分を変換した「conv_body.xml」から DataModel 要素内のデータを使用します。「conv_body.xml」をテキストエディタで開き、「<DataModel name="TheDataModel">」で始まる行を検索します。該当する部分を下記に示します。

<DataModel name="TheDataModel">から</DataModel>までが、制御リクエストの BODY 部分を変換したデータになります。

```
<DataModel name="TheDataModel">
<XmlElement elementName="s:Envelope" ns="http://schemas.xmlsoap.org/soap/envelope/">
  <XmlAttribute ns="http://schemas.xmlsoap.org/soap/envelope/" attributeName="s:encodingStyle">
    <String type="utf8" value="http://schemas.xmlsoap.org/soap/encoding/" />
  </XmlAttribute>
  <XmlAttribute ns="http://www.w3.org/2000/xmlns/" attributeName="xmlns:s">
    <String type="utf8" value="http://schemas.xmlsoap.org/soap/envelope/" />
  </XmlAttribute>
  <XmlElement elementName="s:Body" ns="http://schemas.xmlsoap.org/soap/envelope/">
    <XmlElement elementName="u:IsAuthorized" ns="urn:microsoft.com:service:X_MS_MediaReceiverRegistrar:1">
      <XmlAttribute ns="http://www.w3.org/2000/xmlns/" attributeName="xmlns:u">
        <String type="utf8" value="urn:microsoft.com:service:X_MS_MediaReceiverRegistrar:1" />
      </XmlAttribute>
      <XmlElement elementName="DeviceID">
        <String type="utf8" value="AAAAA" />
      </XmlElement>
    </XmlElement>
  </XmlElement>
</XmlElement>
</DataModel>
```

[変換データの加工]

続いて、変換データのうち、テストデータに置き換えない文字列を指定します。本手順では、制御リクエスト「IsAuthorized」に送信する制御データ「DeviceID」の値のみテストデータに置き換え、それ以外の文字列をテストデータに置き換えません。

「Peach」は、特に指定がない場合、設定ファイルで定義した文字列等のすべてをテストデータに置き換える対象とみなします。したがって、特定の文字列をテストデータに置き換えたくない場合は、その文字列の String 要素に「mutable="false"」²¹を設定します。

前述の変換データの3つの String 要素に「mutable="false"」を設定します（下記の橙色部分）。この加工が終了したら、冒頭「<DataModel name="TheDataModel">」と末尾「</DataModel>」を削除し、「conv_body.txt」として保存します。

```
<DataModel name="TheDataModel">
<XmlElement elementName="s:Envelope" ns="http://schemas.xmlsoap.org/soap/envelope/">
  <XmlAttribute ns="http://schemas.xmlsoap.org/soap/envelope/" attributeName="s:encodingStyle">
    <String type="utf8" value="http://schemas.xmlsoap.org/soap/encoding/" mutable="false" />
  </XmlAttribute>
  <XmlAttribute ns="http://www.w3.org/2000/xmlns/" attributeName="xmlns:s">
    <String type="utf8" value="http://schemas.xmlsoap.org/soap/envelope/" mutable="false" />
  </XmlAttribute>
  <XmlElement elementName="s:Body" ns="http://schemas.xmlsoap.org/soap/envelope/">
    <XmlElement elementName="u:IsAuthorized" ns="urn:microsoft.com:service:X_MS_MediaReceiverRegistrar:1">
      <XmlAttribute ns="http://www.w3.org/2000/xmlns/" attributeName="xmlns:u">
        <String type="utf8" value="urn:microsoft.com:service:X_MS_MediaReceiverRegistrar:1" mutable="false" />
      </XmlAttribute>
      <XmlElement elementName="DeviceID">
        <String type="utf8" value="AAAAA"/>
      </XmlElement>
    </XmlElement>
  </XmlElement>
</XmlElement>
</DataModel>
```

²¹ この mutable 属性については、「ファジング実践資料」の「2.6 Peach が生成するファズデータ」で説明しています。

<http://www.ipa.go.jp/security/vuln/documents/fuzzing-tool.pdf>

※ 設定ファイルの作成

テンプレート設定ファイルにおいて、制御リクエストの HTTP ヘッダと機器パラメータの部分置換して、「設定ファイル用リクエストデータ作成」で作成した変換データを貼り付けて、設定ファイルを作成します（図 52）。

※1: リクエストデータの記述

※2: 機器パラメータの記述

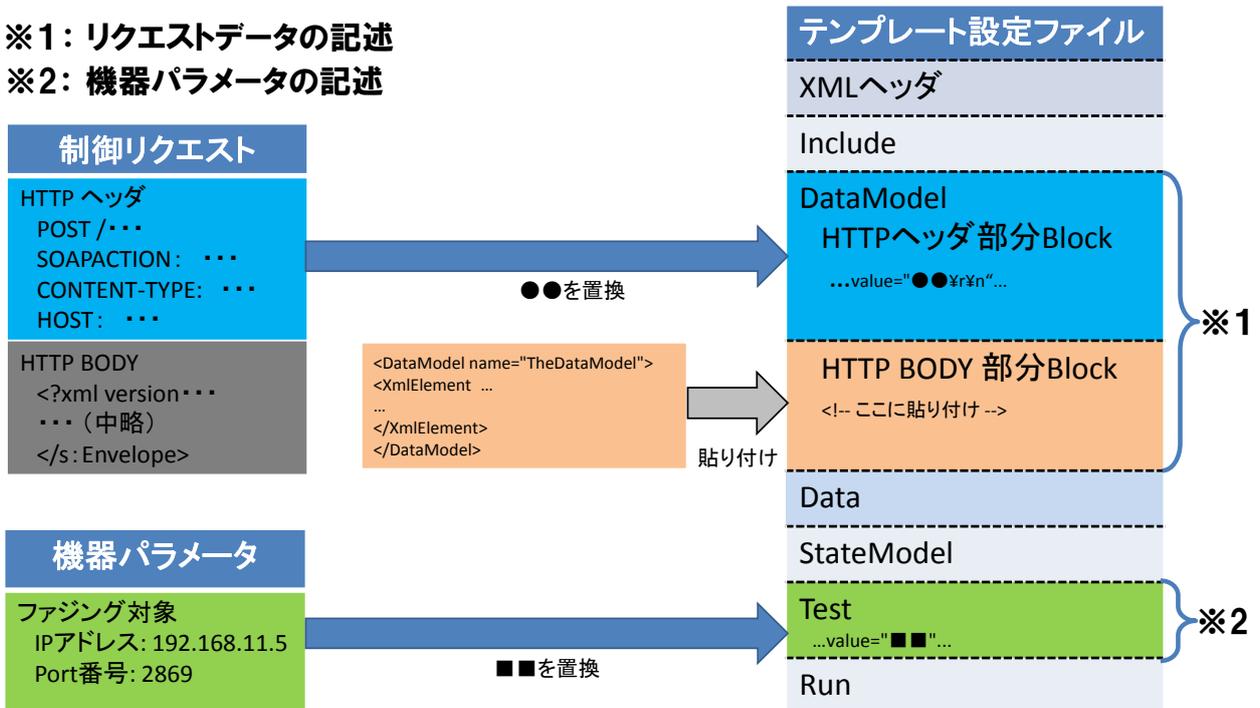


図 52 設定ファイルの作成の流れ

[リクエストデータの記述]

まず、テンプレート設定ファイル（図 52 の青色部分）に制御リクエストの HTTP ヘッダを記述します。テンプレート設定ファイル「upnp_ctr_template.xml」をテキストエディタで開き、制御リクエストの HTTP ヘッダ（図 53 の①～④）に該当する箇所を探します。



図 53 制御リクエストの HTTP ヘッダ部分

テンプレート設定ファイルには、String 要素として図 53 の①から④それぞれの HTTP ヘッダをすでに定義しています（表 9 青色部分 行番号 6, 9, 12, 15）。該当する String 要素の value 属性における●●部分を、図 53 の①から④に置換します。置換するときには、XML の文脈にあわせて「"」等の文字をエスケープする必要がある点に注意してください。

HTTP ヘッダ①～④の値を、テンプレート設定ファイルに置換したものが表 10 になります。●●部分を置換した部分は、表 10 の赤色部分が該当します。

表 10 HTTP ヘッダを置換したテンプレート設定ファイル抜粋（行番号は表 9 と連動）

行	テンプレート内容	図 53 番号
4	<DataModel name="UpnpRequest">	
5	<Block name="RequestLine">	
6	<String name="Method" value="POST /upnpghost/udhisapi.dll?control=uuid:73a1fb2e-a725-4999-a23c-0620dd69393b+urn:microsoft.com:serviceId:X_MS_MediaReceiverRegistrar HTTP/1.1%r%r" mutable="false" />	①
7	</Block>	
8	<Block name="SOAPAction">	
9	<String name="ACTION" value="SOAPACTION: "urn:microsoft.com:service:X_MS_MediaReceiverRegistrar:1#IsAuthorized"%r%r" mutable="false" />	②
10	</Block>	
11	<Block name="CONTENTStype">	
12	<String name="CON-TYPE" value="CONTENT-TYPE: text/xml ; charset="utf-8"%r%r" mutable="false" />	③
13	</Block>	
14	<Block name="HeaderHost">	
15	<String name="Host" value="HOST: 192.168.11.5:2869%r%r" mutable="false" />	④
16	</Block>	
17	<Block name="HeaderContentLength">	
18	<String name="Header" value="Content-Length: " mutable="false" />	
19	<String name="Value" mutable="false">	
20	<Relation type="size" of="X_BODY" />	
21	</String>	
22	</Block>	
23	<String value="%r%r%r%r" mutable="false" />	

続いて、[変換データの加工]で準備した制御リクエストの BODY 部分「conv_body.txt」をテンプレート設定ファイル（図 52 の橙色部分）に記述します。テンプレート設定ファイルで「<!-- ここに貼り付け -->」が記述されている箇所（表 9 橙色部分 行番号 25）を探し、そこに「conv_body.txt」を全て貼り付けます。テンプレート設定ファイルへ「conv_body.txt」の内容を貼り付けたものが表 11 になります。貼り付けた部分は、表 11 の赤色部分が該当します。

表 11 「conv_body.txt」を貼り付けたテンプレート設定ファイル抜粋（行番号は表 9 と連動）

行	テンプレート内容
24	<Block name="X_BODY">
25	<pre> <XmlElement elementName="s:Envelope" ns="http://schemas.xmlsoap.org/soap/envelope/" <XmlAttribute ns="http://schemas.xmlsoap.org/soap/envelope/" attributeName="s:encodingStyle"> <String type="utf8" value="http://schemas.xmlsoap.org/soap/encoding/" mutable="false" /> </XmlAttribute> <XmlAttribute ns="http://www.w3.org/2000/xmlns/" attributeName="xmlns:s"> <String type="utf8" value="http://schemas.xmlsoap.org/soap/envelope/" mutable="false" /> </XmlAttribute> <XmlElement elementName="s:Body" ns="http://schemas.xmlsoap.org/soap/envelope/" <XmlElement elementName="u:IsAuthorized" ns="urn:microsoft.com:service:X_MS_MediaReceiverRegistrar:1"> <XmlAttribute ns="http://www.w3.org/2000/xmlns/" attributeName="xmlns:u"> <String type="utf8" value="urn:microsoft.com:service:X_MS_MediaReceiverRegistrar:1" mutable="false" /> </XmlAttribute> <XmlElement elementName="DeviceID"> <String type="utf8" value="AAAAA"/> </XmlElement> </XmlElement> </XmlElement> </XmlElement> </pre>
26	</Block>

[機器パラメータの記述]

最後に、テンプレート設定ファイル（図 52 の緑色部分）にファジリング対象の機器パラメータを記述します。

テンプレート設定ファイルにおいて、name 属性が「host」の Param 要素の■■をファジリング対象の IP アドレスに、name 属性が「port」の Param 要素の■■をファジリング対象（の UPnP サービス）のポート番号に置き換えます。テンプレート設定ファイルの Param 要素を置換したものが表 12 になります。

表 12 機器パラメータを置換したテンプレート設定ファイル抜粋（行番号は表 9 と連動）

行	テンプレート内容
35	<Test name="UpnpPostRequestTest" description="Upnp Request Post Test">
36	<StateModel ref="State1" />
37	<Publisher class="tcp.Tcp">
38	<Param name="host" value="192.168.11.5" />
39	<Param name="port" value="2869" />
40	</Publisher>
41	</Test>

全ての記述が完了したら、設定ファイル「upnp_ctl_fuzz.xml」として保存します。保存後に、「upnp_ctl_fuzz.xml」に問題がないか、「Peach」を使って確認します。

Peach インストールフォルダに設定ファイル「upnp_ctl_fuzz.xml」をコピーします。Peach インストールフォルダで図 54 のように「peach -t 設定ファイル名」コマンドを実行します。「**File parsed without errors**」と表示されエラーが無いことを確認できたら、設定ファイルは完成です。

```
構文チェック実行コマンド：
peach -t upnp_ctl_fuzz.xml

C:\Windows\system32\cmd.exe
c:\%peach>peach -t upnp_ctl_fuzz.xml

] Peach 2.3.8 Runtime
] Copyright (c) Michael Eddington

File parsed without errors.

c:\%peach>
```

図 54 構文チェックオプション実行画面

6.4.2 「Peach」を使ったファジング

「Peach」による UPnP サービスに対するファジングの準備が整ったため、あとは「Peach」でファジングを実施するだけです。コマンドプロンプトを起動し、Peach インストールディレクトリに移動します。次に、コマンドプロンプト画面で図 55 の様に Peach コマンド、設定ファイル「upnp_ctl_fuzz.xml」を指定し、Peach によるファジングを実行します。

Peach ファジング実行コマンド
peach upnp_ctl_fuzz.xml



```
c:\peach>peach upnp_ctl_fuzz.xml

] Peach 2.3.8 Runtime
] Copyright (c) Michael Eddington

[*] Starting run "DefaultRun"
[-] Test: "UpnpPostRequestTest" (Upnp Request Post Test)
[1:?:?] Element: N/A
      Mutator: N/A

[2:7819:?] Element: N/A
           Mutator: DataTreeRemoveMutator
```

図 55 Peach によるファジング

以上で、【制御】における「Peach」を使ってファジングを実践できます。「Peach」の実行が完了したあとで、ファジング結果を分析する作業等がありますが、本手順書ではそれらの作業の説明を割愛します。

付録 1 : UPnP の仕組み

この付録では、UPnP 機能を実装したノートパソコンをネットワークに接続してから、UPnP 機能を実装した他のノートパソコンに制御リクエストを送信するまでの【探索】・【問い合わせ】・【制御】の HTTP 通信の流れを説明します。

【探索】・【問い合わせ】・【制御】の流れ

図 56 を実現する流れを、UPnP における【探索】・【問い合わせ】・【制御】に当てはめてみると、図 57 のようになります。なお、図 56 は「2 UPnP とは」で紹介した図 3 を再掲したものです。

図 57 で UPnP の流れをイメージしていただき、【探索】・【問い合わせ】・【制御】ごとに実際の HTTP 通信を例示しながら、HTTP 通信の流れを説明します。

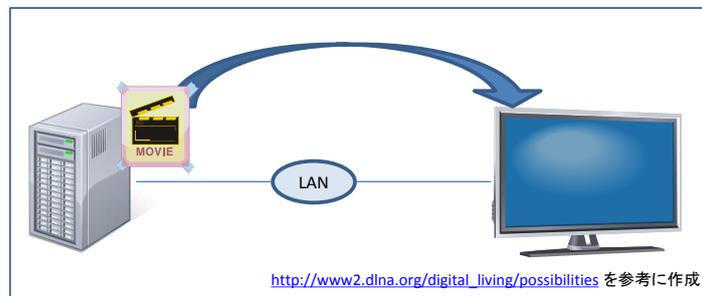


図 56 UPnP を使った動画の視聴 (図 3 と同じ)

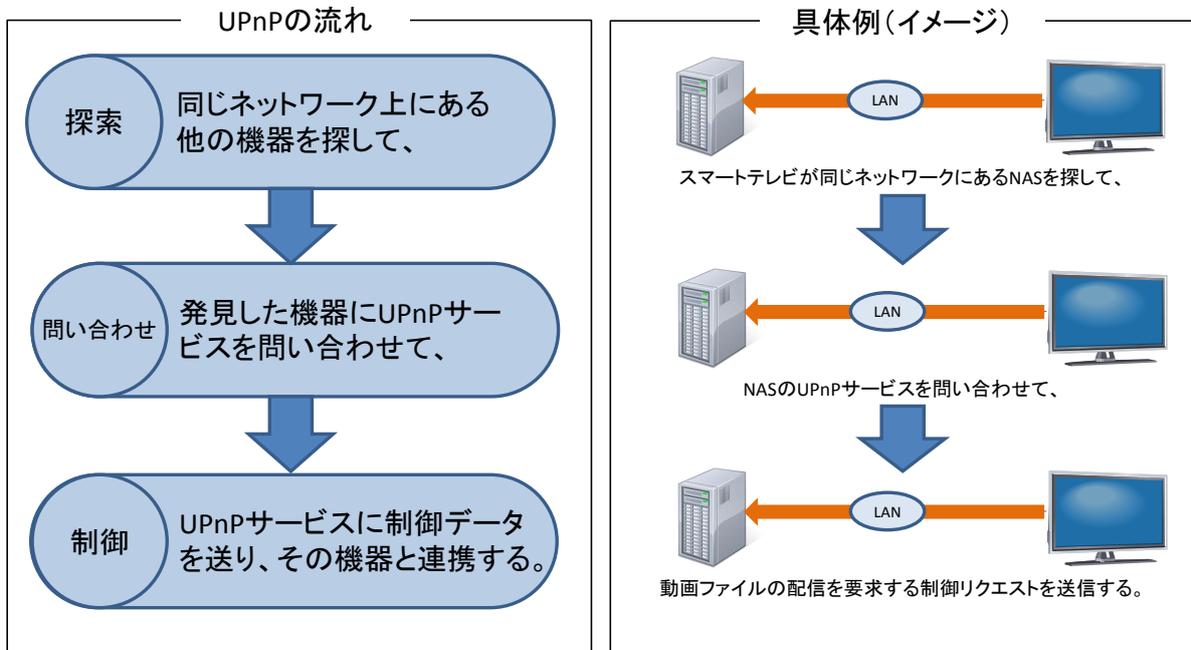


図 57 UPnP の流れ (具体的なイメージ)

【探索】

ノートパソコンをネットワークに接続すると、まず UPnP 機能が同一ネットワークに接続されている他の機器を探索します。同一ネットワークのすべての機器にむけて、ノートパソコンが探索リクエストを送信します (図 58)。UPnP の規格書では、この探索リクエストを「M-SEARCH リクエスト」と定義しています。

探索リクエストは、マルチキャスト IP アドレス「239.255.255.250」のポート番号 1900 番に対して、UDP で送信されます。探索リクエストを図 58 の左下に例示しました。

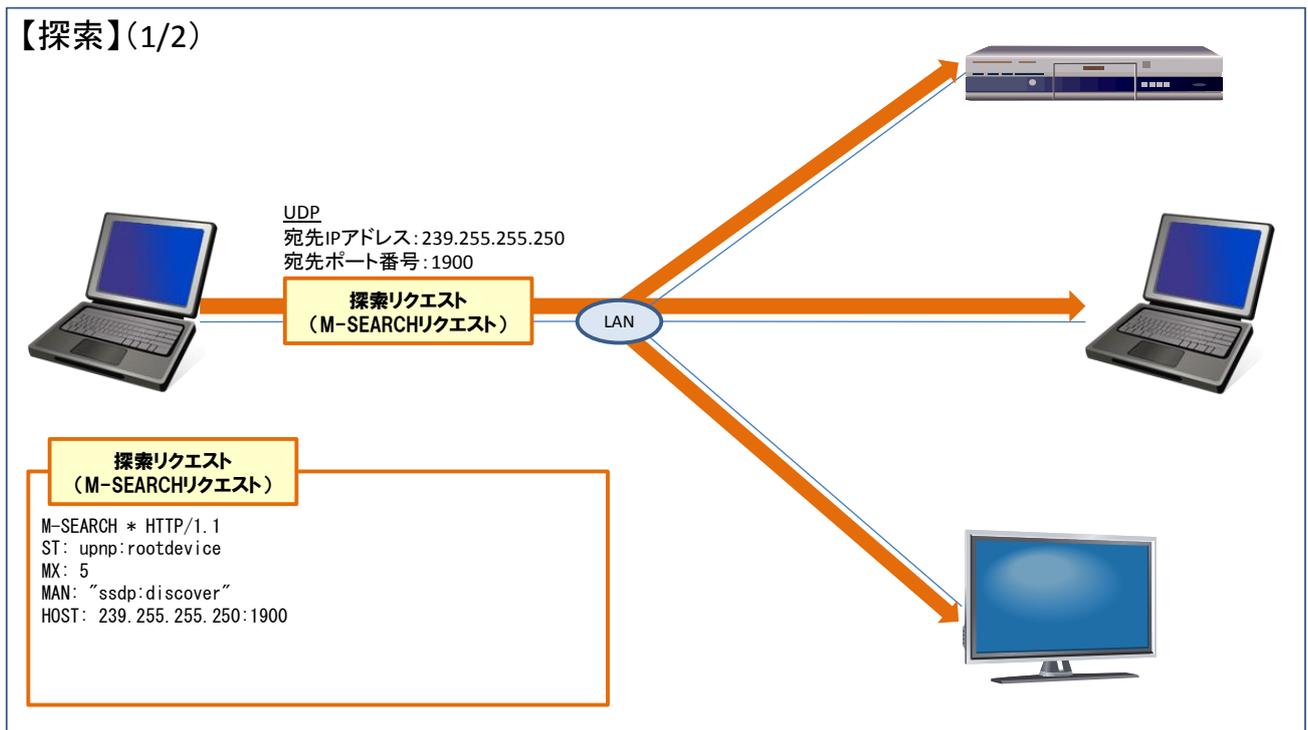


図 58 【探索】(1/2) : 同一ネットワークの機器を探索する

同一ネットワークに他の機器が存在すると、それぞれの機器から探索リクエストに対して応答があります (図 59)。図 59 では、他のノートパソコン、デジタルテレビ、Blu-ray/DVD プレーヤーから応答がありました。

ノートパソコンからの応答を図 59 の左下に例示しました。探索リクエストの応答には、Location ヘッダ (図 59 青色部分) が含まれます。この Location ヘッダでは、UPnP 機能に対する問い合わせ先が明示されます。

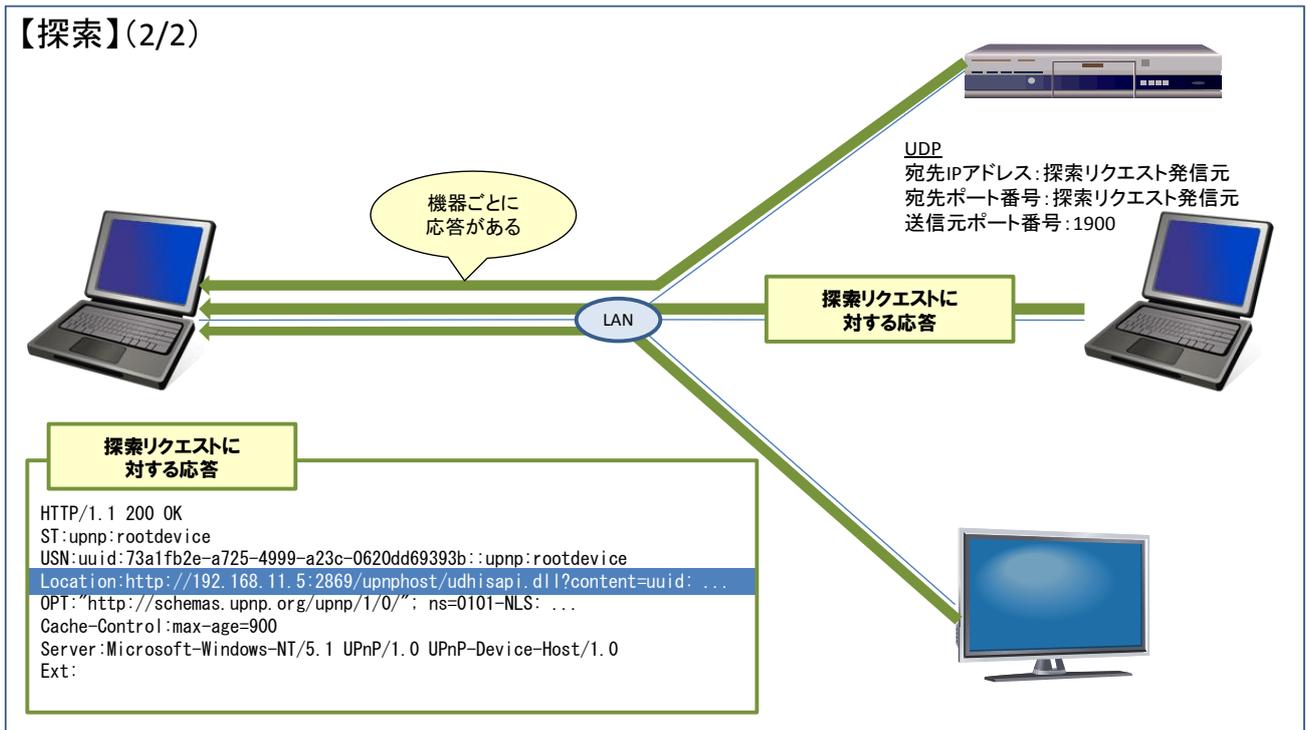


図 59 【探索】(2/2) : 他の機器からの応答

【問い合わせ】

ノートパソコンが同一ネットワークに存在する他のノートパソコンと連携する場合、まずそのノートパソコンが提供している UPnP サービスを問い合わせます。【探索】で得た UPnP 機能の問い合わせ先に対して、問い合わせリクエストを送信します（図 60）。

問い合わせリクエストを図 60 の左下に例示しました。この問い合わせリクエストは、ウェブサーバ等に送信する GET リクエストと同じ形式のものです。

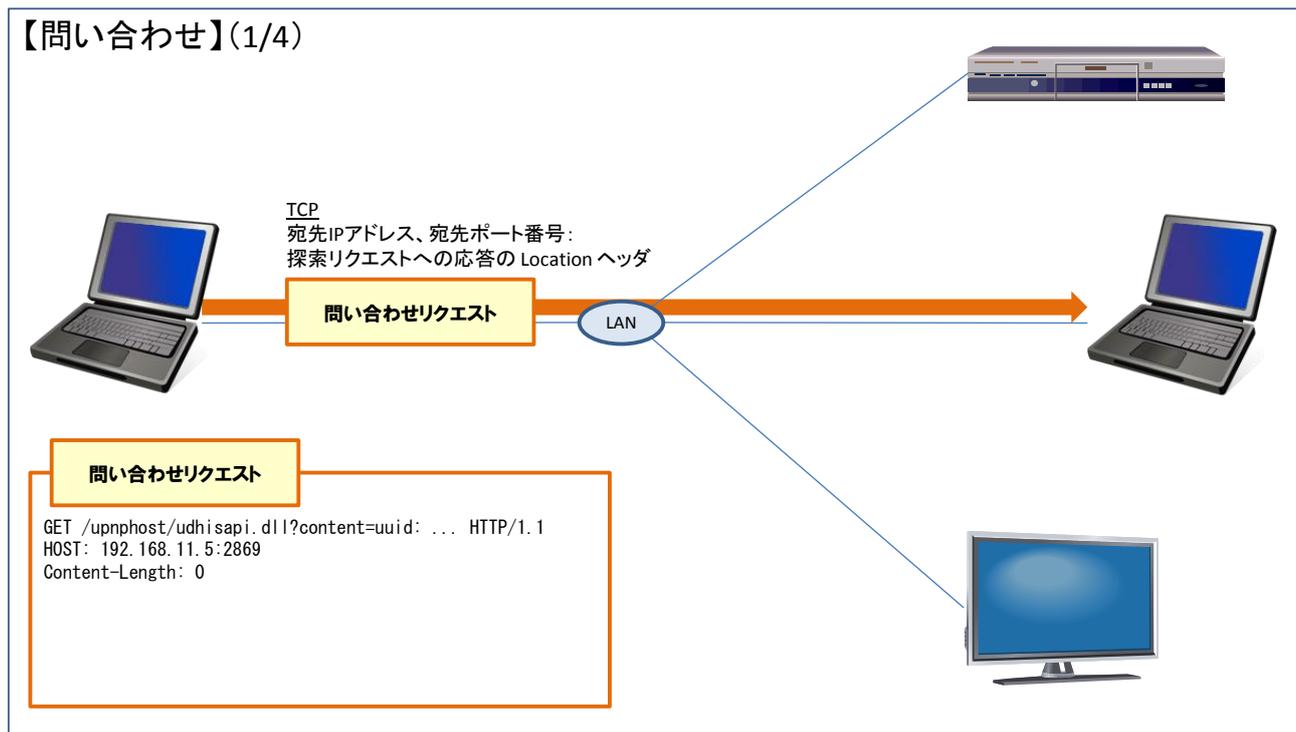


図 60 【問い合わせ】(1/4) : 他のノートパソコンに対する問い合わせ

他のノートパソコンへの問い合わせに対して、そのノートパソコンから応答があります（図 61）。この応答には、ノートパソコンの UPnP 機能に関する情報「Device description」が含まれます（図 61 の青色部分）。この「Device description」²²には、UPnP サービスの一覧とそれらに対する問い合わせ先が定義されています。

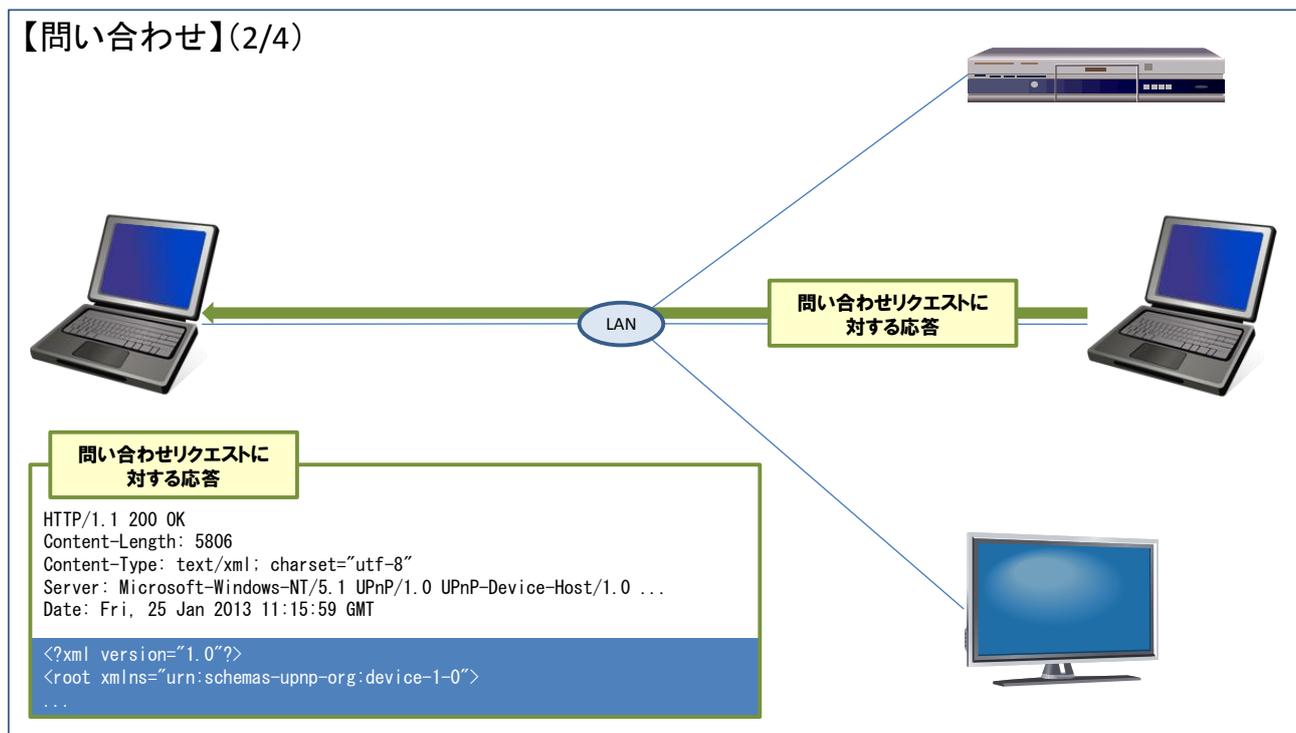


図 61 【問い合わせ】(2/4) : 問い合わせリクエストに対する応答

²² pp.43-48, 2.3 Device description, UPnP Device Architecture 1.1
<http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf>

ノートパソコンの「Device description」を図 62 に例示しました。

serviceList 要素には「UPnP サービスの一覧」が定義されています。serviceList 要素の子要素である service 要素一つ一つが UPnP サービスに該当します。この「Device description」で定義されている UPnP サービスは、「UPnP サービス 1」(赤色部分)、「UPnP サービス 2」(紫色部分)、「UPnP サービス 3」(緑色部分) の 3 つです。

service 要素には、controlURL 要素と SCPDURL 要素が定義されています (青色部分)。controlURL 要素は「UPnP サービスに対する制御リクエストの送信先」、SCPDURL 要素は「UPnP サービスに対する問い合わせ先」を示しています。

**「Device description」
(UPnP機能に関する情報)**

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <device>
    <serviceList>
      <service>
        ...
        <controlURL>/upnphost/udhisapi.dll?control=uuid: ... </controlURL>
        ...
        <SCPDURL>/upnphost/udhisapi.dll?content=uuid:4eddf88-da7d-4a83-ad03-07b00564e2b9</SCPDURL>
      </service>
      <service>
        ...
        <controlURL>/upnphost/udhisapi.dll?control=uuid: ... </controlURL>
        ...
        <SCPDURL>/upnphost/udhisapi.dll?content=uuid:69b92fa4-f260-4e66-80fe-dd943f163258</SCPDURL>
      </service>
      <service>
        ...
        <controlURL>/upnphost/udhisapi.dll?control=uuid: ... </controlURL>
        ...
        <SCPDURL>/upnphost/udhisapi.dll?content=uuid:69b92fa4-f260-4e66-80fe-dd943f163258</SCPDURL>
      </service>
    </serviceList>
  </device>
</root>
```

UPnPサービス1

UPnPサービス2

UPnPサービス3

図 62 「Device description」の例

続いて、ノートパソコンで動作する「UPnP サービス 1」、「UPnP サービス 2」と「UPnP サービス 3」のそれぞれの問い合わせ先の URL に、問い合わせリクエストを送信します。図 63 では、ノートパソコンが他のノートパソコンの「UPnP サービス 1」に問い合わせしている様子を例示しています。

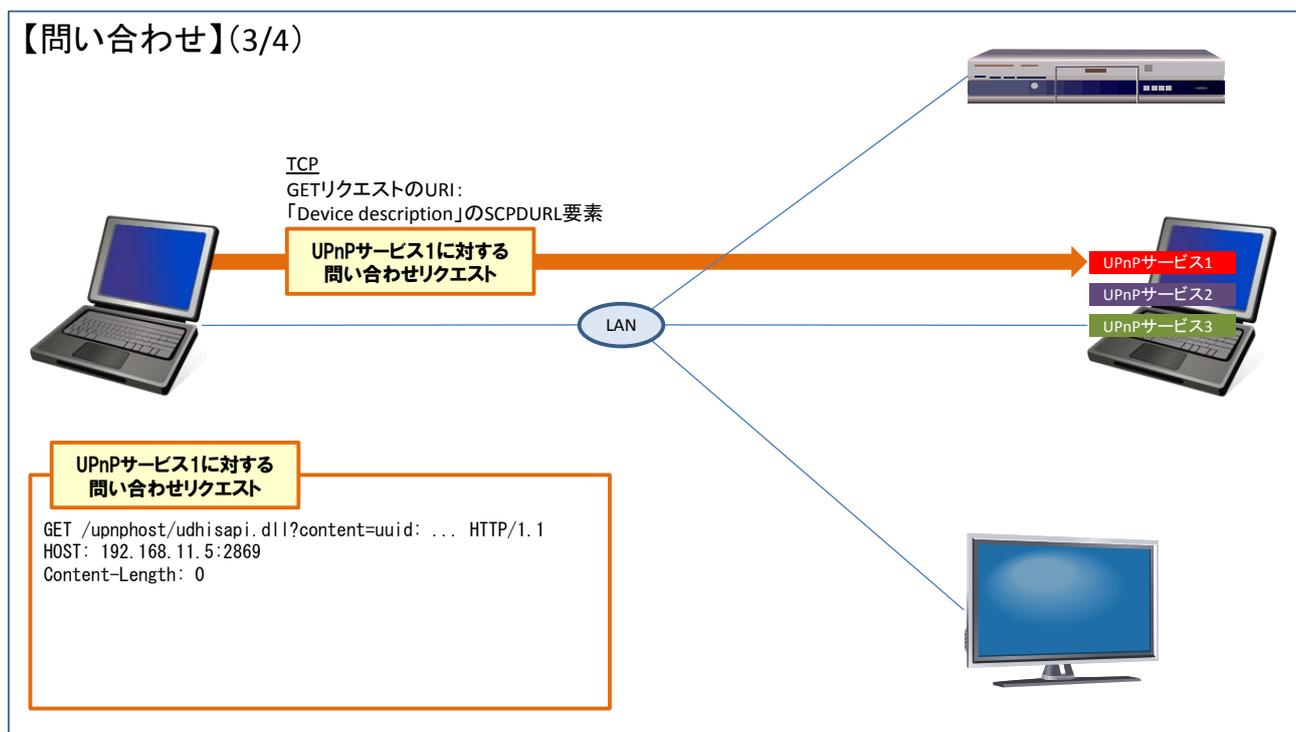


図 63 【問い合わせ】(3/4) : UPnP サービスに対する問い合わせ

他のノートパソコンの「UPnP サービス 1」に対する問い合わせについて、「UPnP サービス 1」から応答があります (図 64)。この応答には、「UPnP サービス 1」に関する情報「Service description」が含まれます (図 64 の青色部分)。この「Service description」²³には、その UPnP サービスが受け付ける制御リクエストの宛先や制御データの一覧が定義されています。

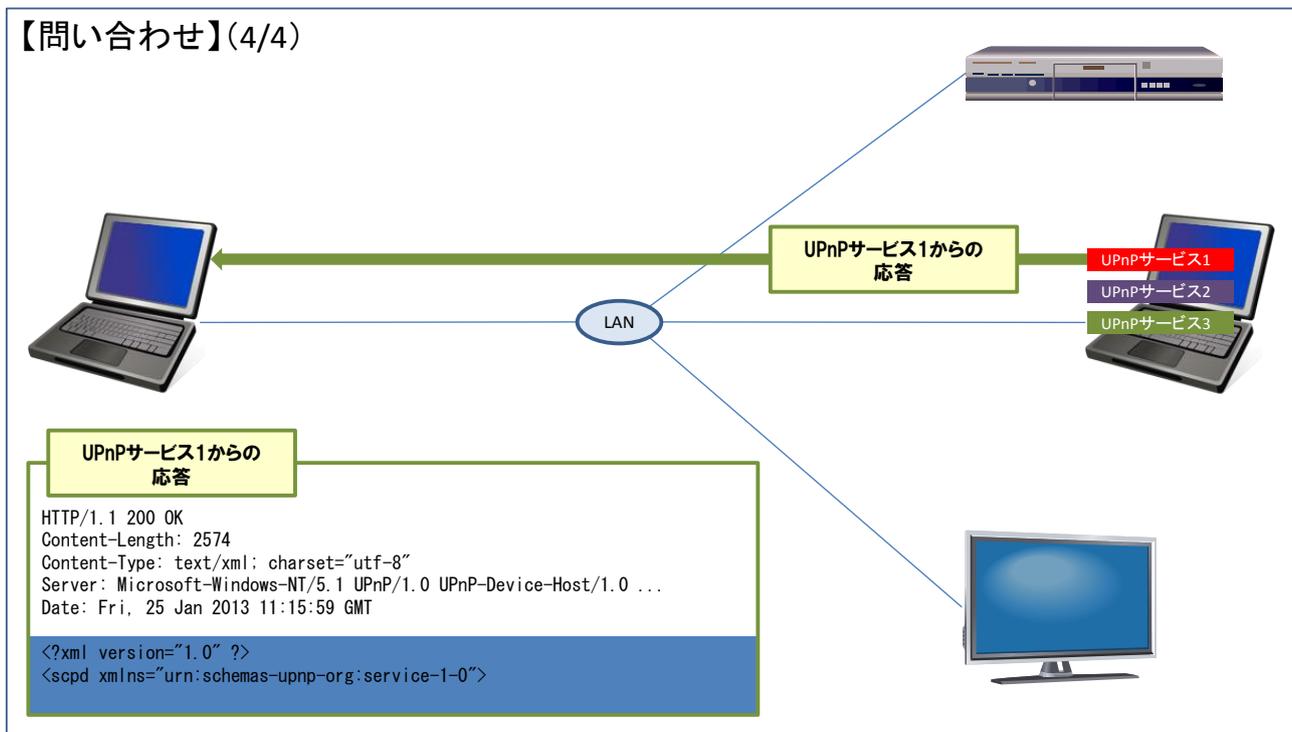


図 64 【問い合わせ】(4/4) : 問い合わせに対する応答

²³ pp.48-55, 2.5 Service description, UPnP Device Architecture 1.1
<http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf>

「UPnP サービス 1」の「Service description」を図 65 に例示しました。

actionList 要素には UPnP サービスが受け付ける制御リクエストの宛先や制御データの一覧が定義されています。actionList 要素の子要素である action 要素一つ一つが制御リクエストの宛先一つ一つに該当します。この「Service description」には「IsAuthorized」という名前の制御リクエスト（図 65 の赤色部分）が定義されています。

action 要素には、argumentList 要素が定義されています。argumentList 要素の子要素として argument 要素が定義されます（図 65 の青色部分）。この argument 要素が制御リクエストに含める制御データ（または制御リクエストの応答に含まれる制御データ）に該当します。

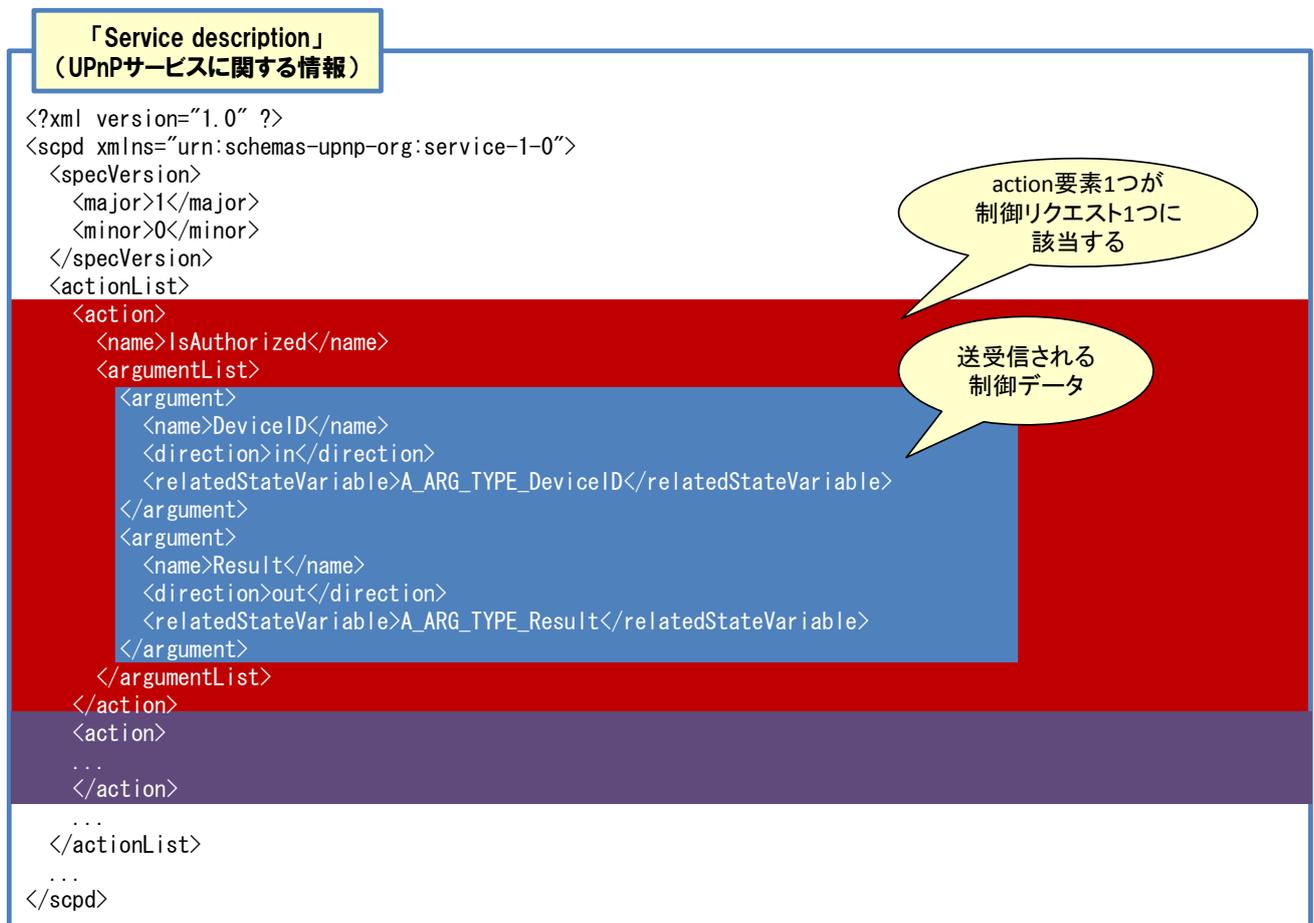


図 65 UPnP サービス 1 の「Service description」の例

【制御】

あとは、ノートパソコンが【問い合わせ】で得られた制御データを他のノートパソコンの UPnP サービスの問い合わせ先の URL に送信します。図 66 では、ノートパソコンが、他のノートパソコンの「UPnP サービス 1」に制御データを定義した制御リクエストを送信する様子を例示しています。

「UPnP サービス 1」に対して制御リクエストを送信する場合、【問い合わせ】の「Device description」で定義されていた「UPnP サービス 1」の control 要素の URL に、POST リクエストを送信します。

制御リクエストを図 66 の左下に例示しました。制御リクエストには、制御データとして【問い合わせ】の「Service description」で定義されていた argument 要素の値を定義しています。

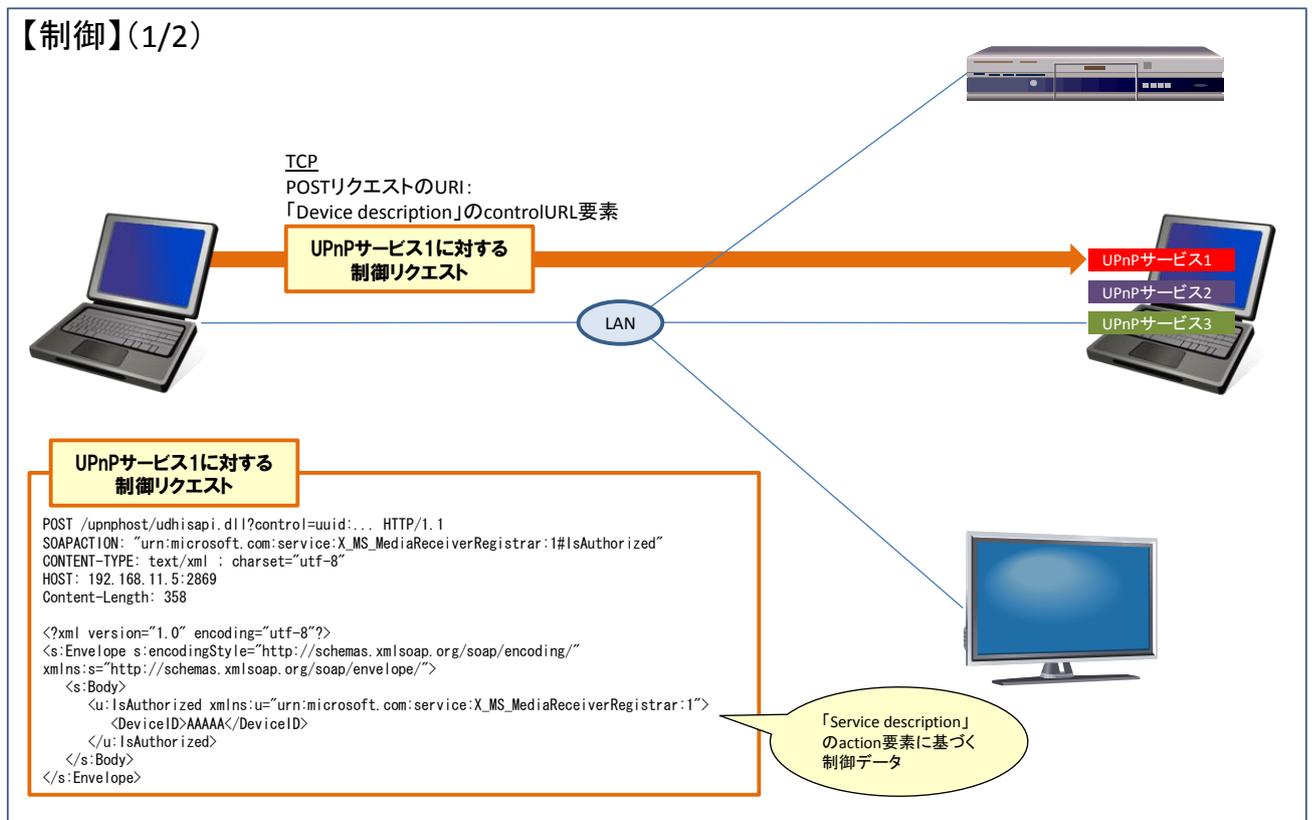


図 66 【制御】(1/2) : 他のノートパソコンの「UPnP サービス 1」に対する制御リクエスト

制御リクエストに対して、そのノートパソコンから応答があります (図 67)。この応答には、制御リクエストの結果が定義されています (図 67 の左下部分)。制御リクエストとして制御データ「DeviceID」に適切な値「AAAAA」を送信したため、図 67 の応答では UPnPError と定義されていることが分かります。制御データ「DeviceID」に適切な値を設定して制御リクエストを送信すれば、きちんとした応答があると考えます。

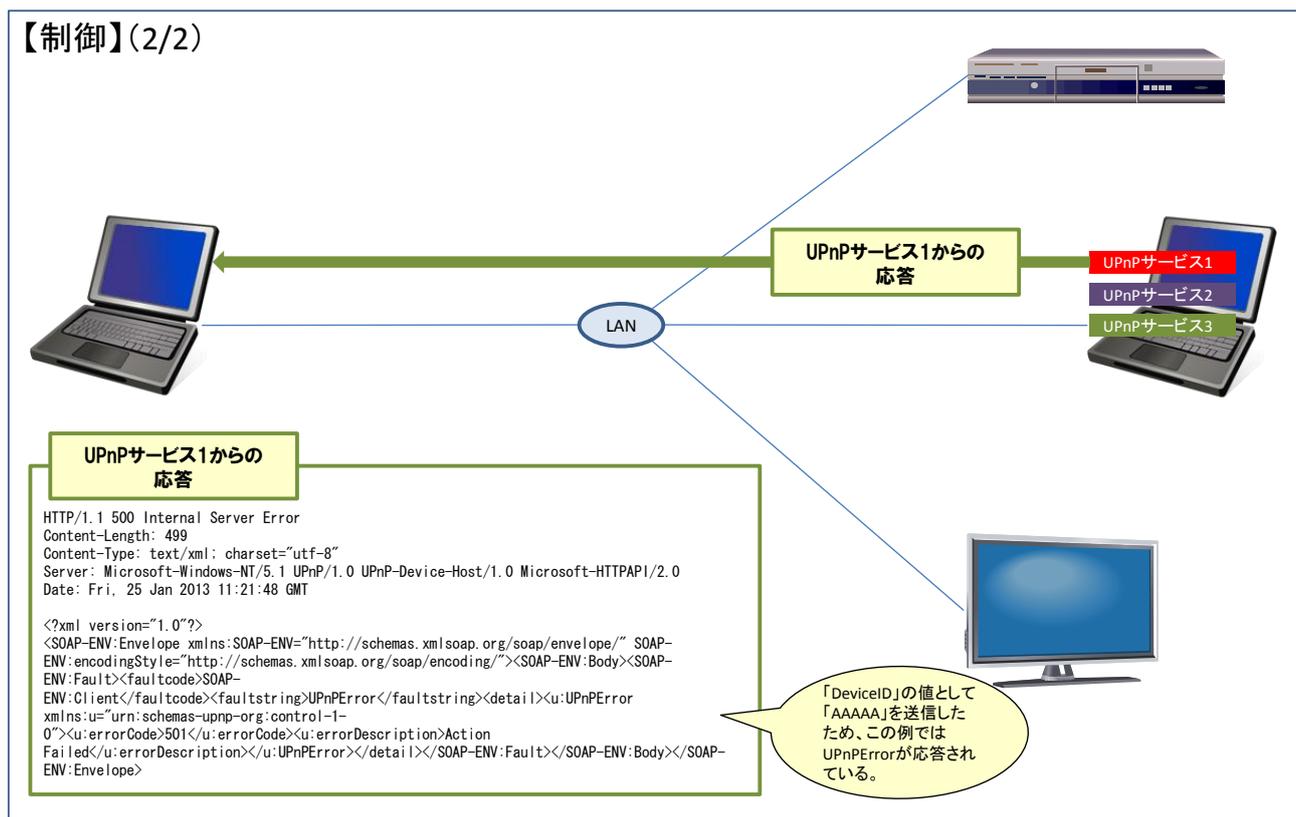


図 67 【制御】(2/2) : 制御リクエストに対する応答

あとはノートパソコンと他のノートパソコンが連携する目的にあわせて、適した制御リクエストを送受信することで、ノートパソコン同士が UPnP を通じて連携できます。

付録 2 : Peach の XML 解析による設定ファイルの出力

XML で記述されたリクエストを送信するファジングでは、「Peach」の XML 解析オプション²⁴を使って設定ファイルを作成できます。このオプションは、XML で記述された入力ファイルを指定すると、ファイル内容を解析して、設定ファイルを出力します。出力されたファイルは、ファジングの実行に必要な各種定義が記述されているため、必要に応じて設定ファイルの内容を変更するだけで、そのままファジングを実行できます。

XML 解析オプションを使用するには、コマンドプロンプトで Peach 実行コマンドの後に「--analyzer」オプションの値に「xml.XmlAnalyzer」、「xmlfile」パラメータの値に「解析ファイル名」、「out」パラメータの値に「出力ファイル名」を指定します。「解析ファイル名」と「出力ファイル名」には XML で記述されたファイルを指定してください。コマンド書式を以下に示します。

XML 解析オプション実行書式

peach --analyzer=xml.XmlAnalyzer xmlfile=解析ファイル名 out=出力ファイル名
--

※ 解析実行時の注意点

XML 解析オプションは、XML だけで記述されたファイルを解析します。そのため、解析ファイルに XML 以外の記述が含まれると、解析エラーとなり、設定ファイルが出力されないことを確認しています。

XML 解析オプション使用例

表 13 の XML ファイルを例にとり、「Peach」の XML 解析オプションの使用例を説明します。なお、表 13 の XML ファイルは、「6.4.1 「Peach」の設定」で使用した XML ファイルと同じものです。

表 13 XML ファイル「upnp_body.xml」

1	<?xml version="1.0" encoding="utf-8"?>
2	<s:Envelope s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
3	<s:Body>
4	<u:IsAuthorized xmlns:u="urn:microsoft.com:service:X_MS_MediaReceiverRegistrar:1">
5	<DeviceID>AAAAA</DeviceID>
6	</u:IsAuthorized>
7	</s:Body>
8	</s:Envelope>

²⁴ xml.XmlAnalyzer - Peach Fuzzer
<http://peachfuzzer.com/XmlAnalyzer.html>

Peach インストールフォルダ（本書では C:\¥peach）に XML ファイル「upnp_body.xml」をコピーします。コマンドプロンプトを起動して、下記の XML 解析オプションコマンドを実行します。出力ファイルには、「conv_body.xml」を指定しています。「Peach」の実行が終了すると、図 68 のように表示されます。

XML 解析オプションコマンド
<code>peach --analyzer=xml.XmlAnalyzer xmlfile=upnp_body.xml out=conv_body.xml</code>

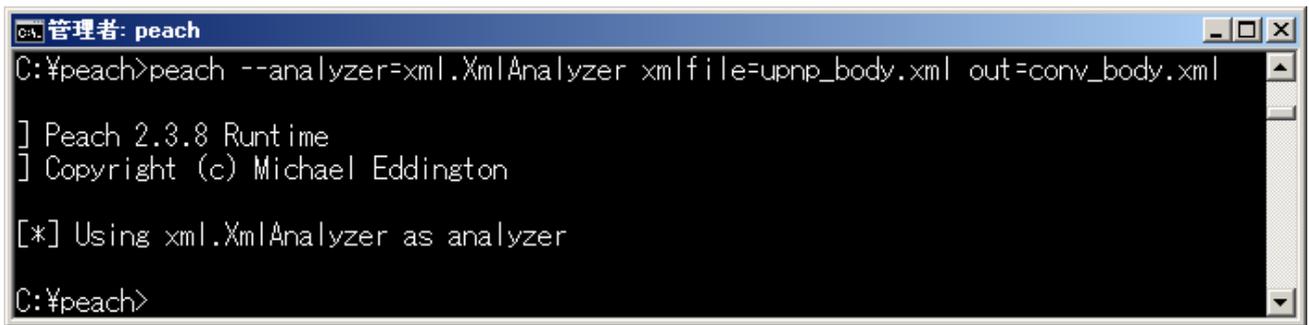


図 68 XML 解析オプションの実行画面

出力された設定ファイル「conv_body.xml」は、表 14 になります²⁵。XML ファイル「upnp_body.xml」の内容は、name 属性の値「TheDataModel」の DataModel 要素の子要素として定義されます（表 14 の緑色部分）。コメントにて「TODO」と記述されている部分（表 14 行番号 25, 33, 41）を修正することで、この設定ファイルを使ったファジニングを実施できます。

IPA では、XML 解析オプションで出力された設定ファイルをそのまま使用せずに、解析された XML ファイルの内容（表 14 の緑色部分）を別の設定ファイルに流用しました。「XML ファイルを『Peach』が読める書式に変換する」という目的にも、この XML 解析オプションを活用できます。

²⁵ 本書に掲載するにあたり、改行および空白、タブ等を中心に整形しました。

表 14 XML 解析オプションで出力された設定ファイル「conv_body.xml」

1	<?xml version="1.0" encoding="utf-8"?>
2	<Peach xmlns="http://phed.org/2008/Peach" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://phed.org/2008/Peach /peach/peach.xsd">
3	<!-- Import defaults for Peach instance -->
4	<Include ns="default" src="file:defaults.xml" />
5	<DataModel name="TheDataModel">
6	<XmlElement elementName="s:Envelope" ns="http://schemas.xmlsoap.org/soap/envelope/">
7	<XmlAttribute ns="http://schemas.xmlsoap.org/soap/envelope/" attributeName="s:encodingStyle">
8	<String type="utf8" value="http://schemas.xmlsoap.org/soap/encoding/" />
9	</XmlAttribute>
10	<XmlAttribute ns="http://www.w3.org/2000/xmlns/" attributeName="xmlns:s">
11	<String type="utf8" value="http://schemas.xmlsoap.org/soap/envelope/" />
12	</XmlAttribute>
13	<XmlElement elementName="s:Body" ns="http://schemas.xmlsoap.org/soap/envelope/">
14	<XmlElement elementName="u:lsAuthorized" ns="urn:microsoft.com:service:X_MS_MediaReceiverRegistrar:1">
15	<XmlAttribute ns="http://www.w3.org/2000/xmlns/" attributeName="xmlns:u">
16	<String type="utf8" value="urn:microsoft.com:service:X_MS_MediaReceiverRegistrar:1" />
17	</XmlAttribute>
18	<XmlElement elementName="DeviceID">
19	<String type="utf8" value="AAAAA" />
20	</XmlElement>
21	</XmlElement>
22	</XmlElement>
23	</XmlElement>
24	</DataModel>
25	<!-- TODO: Create state model -->
26	<StateModel name="TheState" initialState="Initial">
27	<State name="Initial">
28	<Action type="output">
29	<DataModel ref="TheDataModel" />
30	</Action>
31	</State>
32	</StateModel>
33	<!-- TODO: Configure agent/monitors
34	<Agent name="LocalAgent" location="http://127.0.0.1:9000">
35	<Monitor class="test.TestStopOnFirst" />
36	</Agent>
37	-->
38	<Test name="TheTest">
39	<!-- <Agent ref="LocalAgent" /> -->
40	<StateModel ref="TheState" />
41	<!-- TODO: Complete publisher -->
42	<Publisher class="stdout.Stdout" />
43	</Test>
44	<!-- Configure a single run -->
45	<Run name="DefaultRun">
46	<Test ref="TheTest" />
47	</Run>
48	</Peach>
49	<!-- end -->

著作・制作 独立行政法人情報処理推進機構（IPA）

編集責任 小林 偉昭

執筆者 勝海 直人
岡崎 圭輔
澤田 迅

協力者 鵜飼 裕司
金野 千里
板橋 博之

相馬 基邦
谷口 隼祐

「ファジニング活用の手引き」別冊

ファジニング実践資料（UPnP 編）

— DLNA に対応した情報家電に対するファジニングの実践 —

[発行] 2013年 3月18日 第1版

[著作・制作] 独立行政法人情報処理推進機構 セキュリティセンター