卒業論文　2017年度（平成29年度）

定点観測プローブを用いたコミュニティベースのIoT DDoSボットネットアラートシステム

慶應義塾大学 環境情報学部

河口 綾摩

徳田・村井・楠本・中村・高汐・バンミーター・植原・三次・ 中澤・武田 合同研究プロジェクト

2017年12月

卒業論文　2017 年度（平成 29 年度）

定点観測プローブを用いたコミュニティベースの IoT DDoS ボットネットアラートシステム

論文要旨

大量のリクエストを標的のサーバに送信することによってリソースの枯渇を目的とした DDoS 攻撃は、非常に単純かつ古くから知られている攻撃手法である。IoT 機器をターゲットにした DDoS ボットネットや、それらのボットネットをインフラとして持つ DDoS 代行業者の存在により DDoS 攻撃はより身近に大規模かつ高度化してきている。現在の DDoS 攻撃の監視と対策の連携には、ウェブサイトやメールなどのツールを使った注意喚起による情報共有が主流である。しかし、増え続ける DDoS 攻撃に対して今後これらの手法では迅速な対応が困難になると予想されるため、これらの手法にとって代わるより自動化との親和性の高い手法を検討する必要がある。また IoT ボットネットに焦点を当てた場合、ボットネットは非常に多くの感染ノードが標的サーバに対して大量の攻撃を行うという方式である。多くの管理者にとってこれらの感染ノードは正常な通信を行なっているように見えるため、感染ノードや感染しているマルウェアに関する情報を迅速に把握し、対策に役立てる必要がある。本研究では、IoT ボットネットとそれらのボットネットの地理的な感染デバイス数の偏りに着目し上記の二つの問題点を解決することを目的とした DDoS ボットネットアーキテクチャの提案を行う。本研究のアプローチは現在 IETF で標準化が勧められている DOTS(DDoS Open Threat Siganling) プロトコルのユースケースを拡張し、インターネット定点観測観測プローブを用いることでコミュニティーベースの IoT ボットネットのための DDoS アラートシステムのアーキテクチャを提案する。評価としては、擬似パケットデータを用いて中央サーバアーキテクチャの処理時間の計測を想定した規模でシミュレーションを行い、実際に観測データを処理できることを確認できた。

キーワード

DDoS 攻撃, Mirai, ハニーポット, Internet of Things

慶應義塾大学 環境情報学部

河口 綾摩

Abstract Of Graduation Thesis Academic Year 2017

A Community-Based Alert System Against IoT DDoS Botnet with Internet-Probes

Summary

A DDoS(Distributed Denial of Service) attack, which aims to exhaust a target's server resource and make it unavailable temporarily by sending huge amount of requests has been known for a long time as one of the most popular cyber-attack technique. DDoS attack has been getting easier to pull off and more sophisticated because of the presense of a DDoS botnet and DDoS provider service called "booter" or "stresser".

Currently, The mainstream approach to notify users and network administrators of cyber threats is announcement of them on a website or via e-mail alert. It is neccesary for us to consider adopting a new automated-friendly approach against incresingly substaintial amount of DDoS attacks because we will not be able to cope with the attacks swiftly in the near future.

Unlike most DDoS botnets, an IoT botnet takes a form of attacking its target with a huge amount of volume by taking advantage of a lot of vunerable IoT devices scattered on the Internet. It is difficult to identify compromised devices because these attack traffic look normal to network administrators

In this research, we set these problems our goals to achieve and propose an architecture as my contribution. Concreately, we expand a use-case of DOTS(DDoS Open Threat Signaling Protocol) being worked on towards standardization in IETF, propose a community-based DDoS alert system with internet-probes against IoT-botnet

In terms of evaluation, we conducted simulation to measure required time to process randomly generated packets with this architecture on an estimated scale and confirmed that the proposed architecture can process data from probes.

Keywords

DDoS, Mirai, Honeypot,Internet of Things

Faculty of Environment and Studies

Keio University

Ryoma Kawaguchi

目 次

第1章　序論

1.1　背景

DDoS 攻撃が増加の一途を辿っており、NICT のレポート [14] によると 2015 年と比較して観測された DDoS 攻撃の数は二倍以上にもなっており、攻撃の高度化や大規模化が問題になっている。そのような背景として、IoT 機器の増加に伴ってそれらの脆弱性を利用した IoT 型の DDoS ボットネットワークの登場が挙げられる。IoT 型 DDoS ボットネットワークとは一部の IoT 機器に搭載されている管理用 telnet/ssh コンソールを経由して感染する自己増殖型のボットネットワークである。レポート [11] によると、telnet を通じてデフォルトの認証情報や、空文字でコンソールにログイン可能なノードの数は 120 万台を超えるという。従来の DDoS ボットネットワークではアンプ型の DDoS 攻撃が主流であり、これらは比較的少数のノードが大量のトラフィックを生成する形であったが、IoT 機器の攻撃は攻撃ノード数が非常に多く増幅を必要としない所が異なっている。この IoT 機器により構成されるボットネットにより、1Tbps 以上の攻撃トラフィックが生成され主要な CDN サービスや、DNS サービス事業者が一時的に停止に追い込まれる自体に発展した。

1.2　現状の課題

現在のセキュリティの監視と対策の連携には、メール・ウェブサイト等で注意喚起を目的として掲載・送信されるのが主流だがこれらの手法では今後増加すると考えられる IoT 機器を用いた DDoS 攻撃に対して迅速な対応が困難になると考えられる。また、DDoS 攻撃に対する対処としてフィルタルールに攻撃元 IP アドレスの追加などの手段が用いられるが、攻撃元 IP アドレスが動的に変化するためフィルタルールの管理が難しい。IoT ボットネットの登場により、さらに流動性が増し増加するフィルタルールに対して適切な管理手法が必要である。そのため、アラートの方式をメール等の手法に変わる自動化との親和性の高い手法を検討する必要性がある。また、IoT 機器はそのほとんどが組み込み機器であり、ユーザーによって機器がマルウェアに感染しているかどうかの判断が難しい事が問題として挙げられる。このため、攻撃を行ったノードをネットワーク管理者が初期段階で把握、監視、対策をマルウェアの感染の初期段階で行える仕組みが必要である。現在感染している多くの IoT デバイスの分布には地理的な特徴を持っており、各地域のネットワーク環境に対するセキュリティの認知度が比較的低く、かつ脆弱な IoT 機器の市場シェアの高い国や地域においての感染がレポート [21] による報告で目立っている。管理用として telnet サービスがハードコーディングされた認証情報として動作している場合もあり、ファームウェアのアップデートが

行えない等の事情からユーザが対処できず結果的にリコールが行われた IoT 機器 [20] も存在する。今後 IoT 機器をターゲットにしたマルウェアの自己増殖のためのスキャン活動が活発になると予想される中で、初期感染確率の向上の観点から新しい攻撃に関してもこれらの国と地域での初期感染活動が行われやすいため、この条件を仕組みを構築する上で考慮する事が必要になってくる。

1.3 本論文の目的

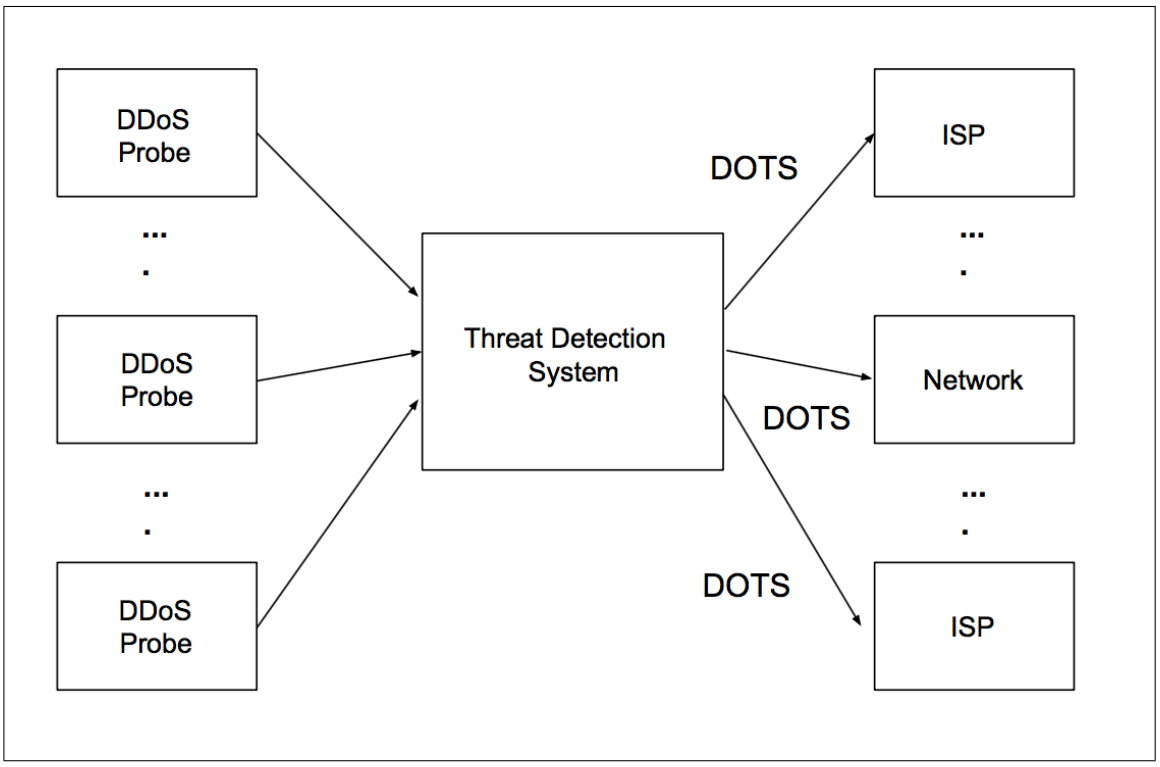

図 1.1: 本論文の最終的な目標

本論文の目的では、現状の課題で述べた二つの問題を解決するために、IoT 機器を用いた DDoS ボットネットによる攻撃を初期段階で迅速に観測し、メール等に挙げられる必ず人が介在する手法を使わずに自動化との親和性の高い手法を採用することによってプローブを設置している参加者間で迅速に通知を行える観測アラートシステムのアーキテクチャを提案する事である。この目的を達成するために本論文ではアプローチとして上図のような観測プローブから収集されたデータの収集・分析を脅威判定システム上で行い、受取手の設定した閾値等の条件を設定した上で条件に該当した場合はシグナリングは標準化された DDoS 攻撃のためのシグナリングプロトコルを用いて即座にシグナリングを行うシステムが本論文提案手法の全体像である。このため、本論文における議論を以下の三つの項目に大まかに区切った上で行う。

- DDoS 攻撃観測プローブの構成と脅威判定システムとの間の通信

- 脅威判定システム内部での動作とアーキテクチャ
- 参加者によるデータを取得するための条件の記述、脅威判定システムからのシグナリング

観測プローブを物理的に広域で設置することによって、IoT ボットネットの多くの感染デバイスを持つ国や地域に対してより集中的にデータを収集可能である。プローブとしては安価なハードウェアを用いて設置の際にも回線の提供者が設置を行う際も手間がかからず、セキュリティリスクもある程度防止することが可能であることも利点として挙げられる。本論文ではこの図を最終的な達成する目標としてその一部分であるプローブ部分と中央サーバアーキテクチャの実装と評価を行う。評価としては、擬似的に生成したパケットデータを一定の pps(packet per second) で供給し、サーバアーキテクチャの性能評価のための計測を評価として行う。

1.4 本論文の構成

本論文の構成について述べる。2 章では、本論文の提案手法を構成するための分野を分類し、それらの既存手法や関連技術について記述を行う。具体的には、IoT 機器を用いたボットネットのアーキテクチャの解説を行い、関連研究や過去の著者の研究から得た成果を合わせて解説する。3 章では、本論文の提案手法について第 1 章の研究の目的で述べた事項を確認しながら、ユースケースを 3 つ挙げた上でアーキテクチャを提案する。4 章では、評価を行うために中央のサーバアーキテクチャを実装し、その実装のために必要な認証プロトコルや HDFS 上での処理について記述を行う。5 章では、4 章で実装を行なった中央のサーバアーキテクチャを想定した観測規模での評価実験を行い、そのデータを示す。6 章では、1-5 章までの結果をまとめた上で本提案手法を用いた今後の課題と展開について議論する。

第2章　既存技術・関連研究

2.1　概要

本論文の提案手法は複数の分野の関連研究やプロジェクトの成果が関連して構成されているため、それぞれの関連研究の説明を事前準備として全体としての分野の繋がりをここで明確にする。以下は本論文で用いられるそれぞれの関連研究と、提案手法におけるアーキテクチャとの関連を示したものである。

2.2　IoT 機器を標的にした DDoS ボットネット

本節では、2016 年に出現した IoT 機器を標的にした DDoS 攻撃のためのボットネットの中でも、特に大きな被害をもたらしたボットネットであり、ソースコードが公開されていてアーキテクチャに対する評価が行いやすい Mirai を例に、アーキテクチャやその亜種の移り変わり、関連研究から得た Mirai の攻撃の特徴や、感染デバイスの種別、地理的配置によるデータを元に考察を記述する。

2.2.1　BASHLITE

BASHLITE[1](別名 Gafgyt,ELFBASHLITE.A) は 2014 年 9 月に出現した Linux デバイスをターゲットにしたもので出現当時は IoT 機器を重点的にターゲットにしたものではなかった。しかし、Qbot[7] などに代表される BASHLITE の亜種の出現によってより悪意のある方法によって IoT 機器や脆弱なインターネット接続された組み込みデバイスがターゲットにされ、攻撃インフラとして利用されるようになった。現在、オリジナルの BASHLITE のソースコードは公開されている。

オリジナルの BASHLITE では、クライアントコードとサーバーコードで構成されておりサーバーコードは C&C サーバ上で動くプログラムとして設計されており、IRC ベースのプロトコルによってクライアントコードとのコミュニケーションを行うようになっている。クライアントは telnet スキャンを用いて、.1 .255 範囲での/24 のサブネット単位で IP アドレスをランダムに探索することで、BASHLITE が持っているユーザー ID とパスワードの組を用いてログイン試行を行う。この時、IANA が指定している特別目的の IP アドレスブロックは探索対象から外すように処理が行われており、ボットネットの探索効率を上げ、不要なバグを発生させないようにする工夫が見られる。

2014 年 10 月ごろに出現した BASHLITE の亜種である ELFBASHLITE.SMB[6] では、ログイン後にターミナルのシェルの一種である Bash の Shellshock 脆弱性 [5] を用いて、root 権限を取得して後マルウェアを外部の HTTP サーバからダウンロードするように改良され、その後多数の亜種が出現した。

2.2.2 Mirai

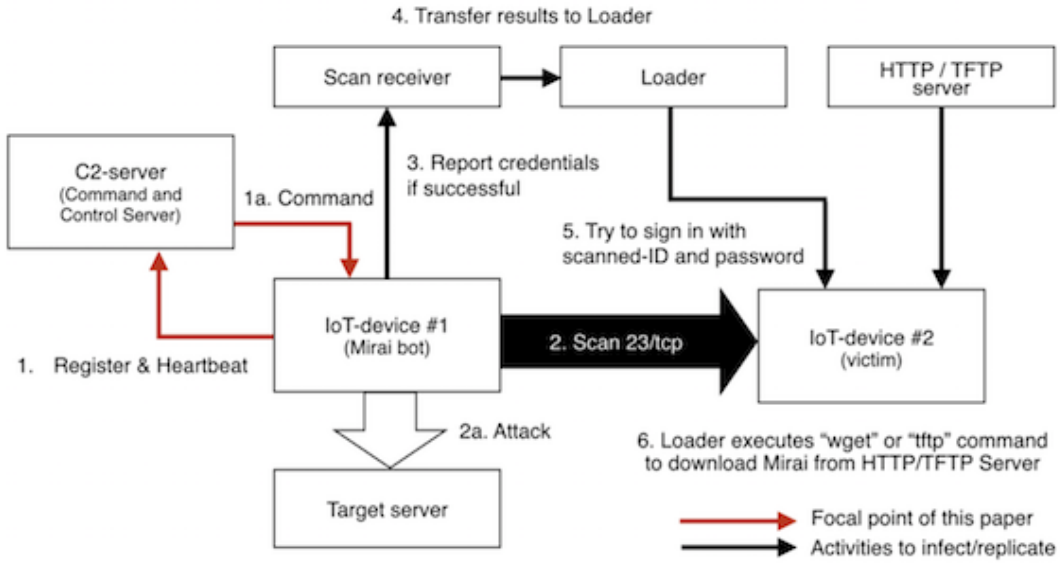

図 2.1: Mirai の感染と拡散

IoT ボットネットのアーキテクチャの代表例として、Mirai のアーキテクチャを公開されているソースコードをを用いながら解説する。Mirai は C2 サーバ、スキャンレシーバ、ローダー、マルウェアのダウンロード元である HTTP/TFTP サーバより構成される。ここではすでに Mirai ボットネットに感染しているノードがあるという前提で話を進める。すでに感染しているノード (IoT-device 1) は、定期的に C&C サーバからの攻撃司令の受信・C&C 死活確認サーバへの自身の死活報告、およびインターネット上に存在する IoT デバイスを IPv4 アドレス空間よりスキャンを行うという活動の三つを行う。スキャン活動では、TCP プロトコルのポート番号 23 番と 2323 番 (以降 tcp/23, tcp/2323) に対して tcp/23 : tcp/2323 = 9:1 の割合でスキャンを事前にマルウェア自体にハードコードされた認証情報によってログイン試行を行う。Mirai にも Bashlite で使われている辞書データは利用されており、Mirai ではマルウェアが保持している辞書攻撃に用いられる認証情報の組の数は 62 個まで増加した。

この時、スキャンする IP アドレスは開発者の意向によって組織ごとのアドレスブロックにおいて例外的に除外されているものも存在しており、以下の図に示す IP アドレスに関してはスキャンを行わないようにプログラムされている。これは、攻撃者が感染活動を行うにあたって無意味であると判断したアドレス空間や、攻撃者への危害を加えられる可能性を排除したいという思惑があったという説がこのようなコードを記述した理由として有力である。

表 2.1: 探索除外リスト

IP アドレス	ネットワーク部	詳細
127.0.0.0	8	Lookback
0.0.0.0	8	Invalid
3.0.0.0	8	General Electric
15.0.0.0	7	Hewlett-Packard
56.0.0.0	8	US Postal Service
10.0.0.0	8	Internal Network
192.168.0.0	16	Internal Network
172.16.0.0	14	Internal Network
100.64.0.0	10	IANA NAT Reserved
169.254.0.0	16	IANA NAT Reserved
198.18.0.0	15	IANA Special Use
224.x.x.x		Multicast
下位 1byte が 6/7/11/21/22/ 26/28/29/30/ 33/55/214/215		Department of Defense

また、辞書攻撃として利用するハードコードされた認証情報は上記のような認証情報が利用されており、root, password などの一般的な初期設定のパスワードや、特定のベンダーの製品のみを狙ったものも存在している。レポート [11] によると、一般的な初期設定のパスワードや、空文字でログインすることのできるセキュリティ対策の行われていないデバイスは 120 万台に上るという。

辞書攻撃に成功すると Scan Receiver にその認証情報を通知、Scan Receiver は Loader と呼ばれるマルウェアをダウンロードさせるサーバに対してログイン試行で認証に成功した認証情報を送信する。

Loader は Scan Receiver から受け取った認証情報を使ってホストにログインし、対象感染ホストのアーキテクチャを判断した後、適切な対象アーキテクチャ向けにコンパイルバイナリを TFTP 経由でダウンロードする。TFTP コマンドが busybox に存在しなかった場合は echo コマンドで自作の http ダウンローダを送り込み、そこから HTTP 経由でマルウェアをダウンロードする。この時にアーキテクチャの判断手法として、/bin/echo のバイナリファイルを読み込んでそこからアーキテクチャを判断している記述が見られる。

マルウェア本体のダウンロードが完了すると自分自身のバイナリファイルを実行してメモリにロードした後、自分自身を削除。IP アドレス空間をスキャンしてログイン試行を行おうとしてくる他の IoT マルウェアにコントロールを奪われないよう、自身のプロセスのみを起動できるようにシステムコールを感染ホスト上で発行、プロセスの締め出しを行う。揮発性メモリにのみ Mirai は存在する事になるためデバイスの再起動を行うと Mirai の活動は停

表 2.2: 認証情報一覧

ユーザー ID	パスワード	ユーザー ID	パスワード
root	xc3511	admin1	password
root	vizxv	administrator	1234
root	admin	666666	666666
admin	admin	888888	888888
root	888888	ubnt	ubnt
root	xmhdipc	root	klv1234
root	default	root	Zte521
root	juantech	root	hi3518
root	123456	root	jvbzd
root	54321	root	anko
support	support	root	zlxx.
root	(none)	root	7ujMko0vizxv
admin	password	root	7ujMko0admin
root	root	root	system
root	12345	root	ikwb
user	user	root	dreambox
admin	(none)	root	user
root	pass	root	realtek
admin	admin1234	root	00000000
root	1111	admin	11111111
admin	smcadmin	admin	1234
admin	1111	admin	12345
root	666666	admin	54321
root	password	admin	123456
root	1234	admin	7ujMko0admin
root	klv123	admin	1234
Administrator	admin	admin	pass
service	service	admin	menism
supervisor	supervisor	tech	tech
guest	guest	tech	tech
guest	guest	mother	fucker
guest	12345	(none)	(none)

止する。

具体的なプロセスの締め出し手段については以下のような動作が行われる。

- 1. 22/tcp,23/tcp,80/tcp ポートを bind() しているプロセスを終了し、自分自身のプロセスに bind し直す.
- 2. 48010/tcp ポートを自分自身のポートに bind する.
- 3. 現在感染対象にしているデバイス上で動作している全てのプロセスを探索して、以下の競合のマルウェアが動作していないかチェック。動作していたら停止させる。
- 4. .anime を持つ拡張子を探索し (競合の IoT デバイスを対象としたマルウェアと見られる)。こちらのプロセスも停止させる。

```
m qbot reportREPORT %s:%s
m qbot httpHTTPFLOOD
m qbot dupLOLNOGTFO
m upx str\x58\x4D\x4E\x4E\x43\x50\x46\x22
m zollardzollard
```

マルウェアが起動するとマルウェア内にハードコードされている C&C サーバに接続し、自分自身を感染ホストとして登録後 C&C サーバからの攻撃指令を待つという次のステップへと以降を行うのが、Mirai の大まかな動きとなっている。

(1) C&C サーバとのコミュニケーション

マルウェア内部で行われる、攻撃者からの指令を感染ホスト群にブロードキャストする役目を持つ C&C(Command and Control) サーバとのコミュニケーションについて解説を行う。開発者によって公開された Mirai のソースコードの中には C&C サーバのソースコードも含まれており、それに基づいた C&C サーバの動作についての記述を行う。

C&C サーバは Go 言語で実装されており、起動すると 23/tcp(telnet) 番ポートでサービスを起動する。起動すると、以下のような画面が表示され、現在配下に置かれている接続済みボットの数が表示される。攻撃者はこのプロンプトを用いて攻撃コマンドを発行することになる。攻撃コマンドでは、攻撃先のサブネットと攻撃に使用する攻撃タイプ、および攻撃の間隔を指定する。実際に利用されるコマンドのフォーマットは以下のようになる。

コマンド実行例

```
# -1 udp 12.34.56.78/24 10
# -10 syn 12.34.56.78/24,87.65.43.21/24 3600
```

マルウェア内部にハードコードされている文字列には、CNC, SCAN CB と呼ばれる値が存在し、CNC は C&C サーバに利用されるドメイン。SCAN CB は IPv4 空間をスキャンした結果を報告するためのサーバの IP アドレスとなっている。

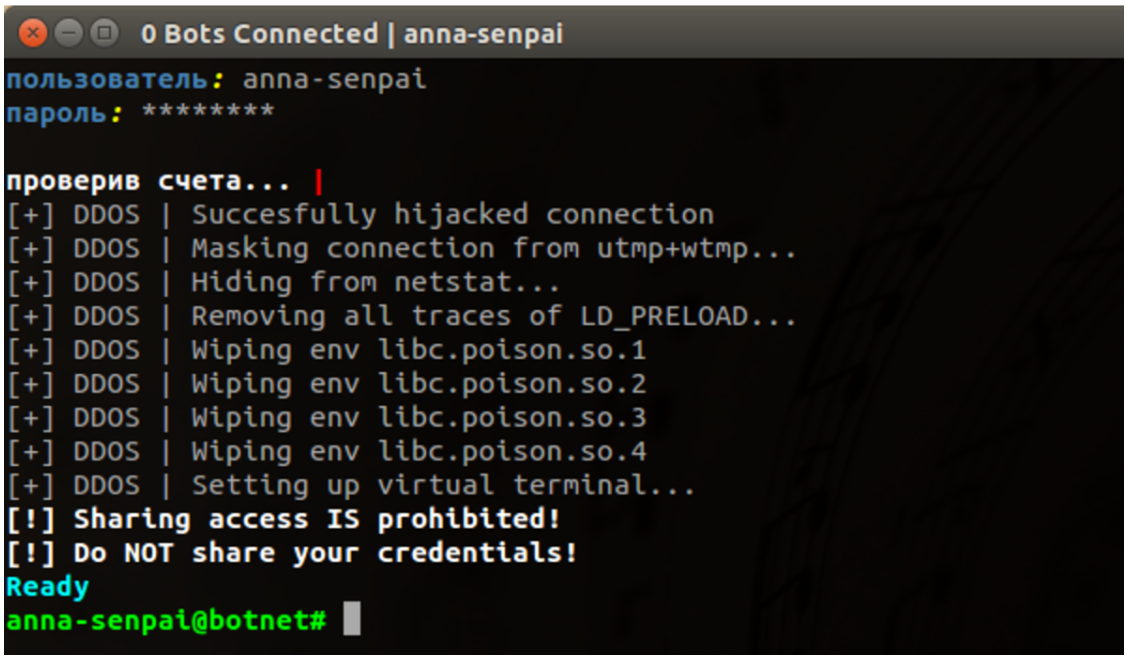

図 2.2: C&C サーバの管理者コンソール

これらのハードコードされた IP アドレスは内部の XOR 演算を用いた難読化処理によって難読化されており、マルウェア本体から直接確認でくるものではない。Mirai ソースコード中の mirai/tools/enc.c にはこのエンコードを行うツールのソースコードが記述されており、以下のように 0xdeadbeef とキーとして XOR 演算を 4 回繰り返して難読化を行うという処理になっている。

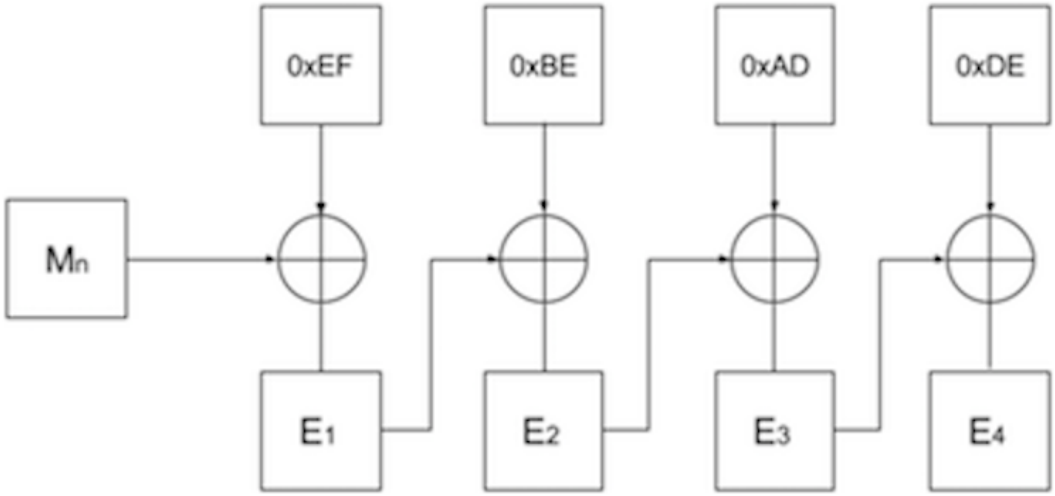

図 2.3: オリジナルの Mirai で利用されていた XOR 暗号化アルゴリズム

アドレス難読化ルーチン

```
void *x(void *_buf, int len)
{
    unsigned char *buf = (char *)_buf, *out = malloc(len);
    int i;
    uint8_t k1 = table_key & 0xff,
            k2 = (table_key >> 8) & 0xff,
            k3 = (table_key >> 16) & 0xff,
            k4 = (table_key >> 24) & 0xff;
    for (i = 0; i < len; i++)
    {
        char tmp = buf[i] ^ k1;
        tmp ^= k2;
        tmp ^= k3;
        tmp ^= k4;
        out[i] = tmp;
    }
    return out;
}
```

C&C サーバとマルウェアの間でやりとりされるメッセージには以下の三つのメッセージがある。

- 1. Register メッセージ
- 2. HeartBeat メッセージ

- 3. Attack メッセージ

Register メッセージは、感染した IoT 機器が C&C サーバに対して攻撃命令をこれから受信するリクエストを行う。C&C サーバに対して送信されるパケットは以下のようなフォーマットになっている。

表 2.3: Register パケット

パケット	データ長	バイト列
1	4	\x00\x00\x00\x00
2	1	起動引数のデータ長 (n とする)
3	n	起動引数

Mirai は自身がまだ攻撃者の配下に置かれていることを報告するために 60 秒ごとに定期的に Heartbeat メッセージを送信する。

表 2.4: Heartbeat パケット

パケット	データ長	バイト列
1	2byte	\x00\x00

Register コマンドが送信され、なおかつ定期的に Heartbeat メッセージが受信されていることを確認したら C&C サーバにより攻撃者の意向に応じて攻撃メッセージが発行される。

表 2.5: C&C サーバからの攻撃指令の内容

データ長	バイト列
2byte	パケット長
4byte	攻撃の実行時間
1byte	攻撃タイプを指定する ID
1byte	攻撃先の数
4byte	攻撃先の IP アドレス
1byte	攻撃先のネットマスク
1byte	フラグの数
1byte	フラグの ID
1byte	フラグに指定するデータ長 (n とする)
nbyte	フラグに指定するデータ

(2) C&C サーバの観測に関する関連研究

上記で解説した C&C サーバと Mirai 本体のコミュニケーションプロトコルでは、仕様上必ずしも IoT 機器をマルウェアに感染させる必要がないため攻撃がないため、上記で解説した

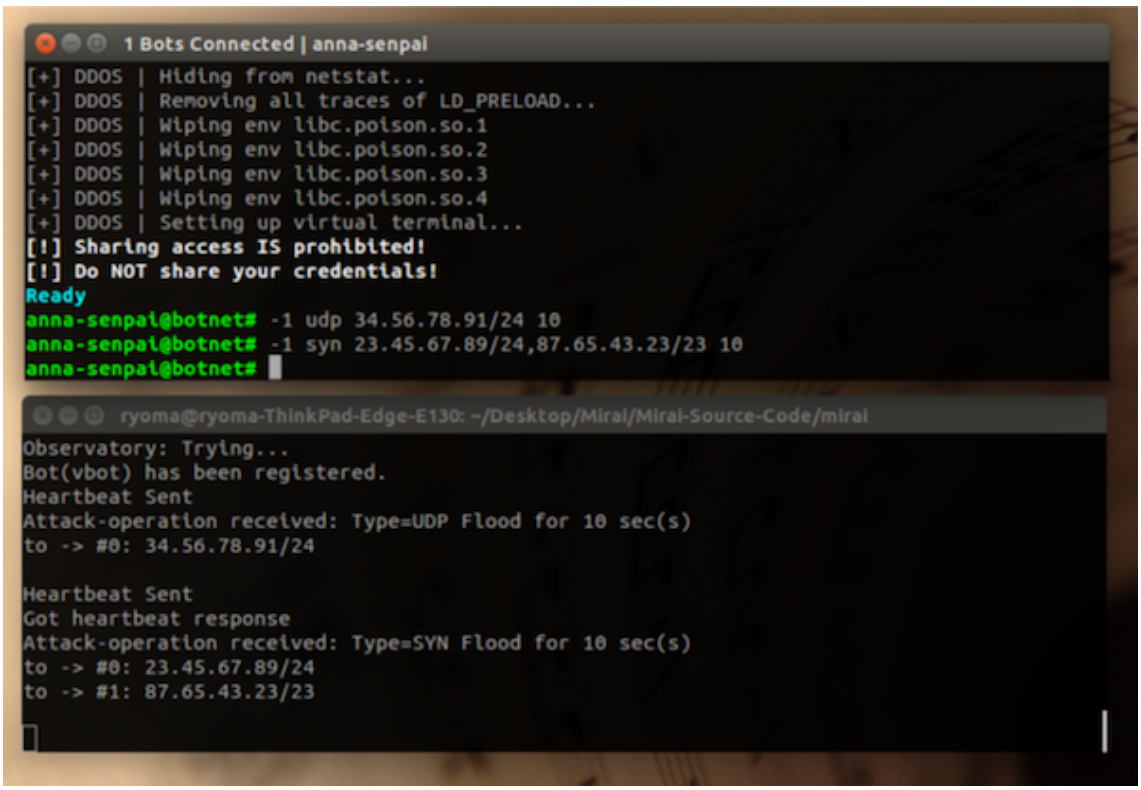

図 2.4: C&C サーバからの司令を受信するプログラム

コミュニケーションプロトコルを実装することによって、C&C サーバから送信される攻撃指令を観察することが可能である。そこで、2017 年の 7 月に北海道で開催された DICOMO'18 にて「Mirai 型 DDoS ボットネットの監視環境の構築」というタイトルで発表を行い、その論文の中で Mirai のコミュニケーションプロトコルを実装したモニタリングプログラムをインターネット上で公開されている Mirai の C&C サーバのアドレス群に対して接続させ、観測を行なった。

上図は実際に観測プログラムを動作させている時のテスト画面であり、ローカル上で動作させている C&C サーバのプログラムから送信された指令がきちんと受信できていることが確認できる。

Mirai の C&C サーバに接続後、攻撃指令を取得する事ができ、インターネットで得られた C&C サーバの IP アドレスだけでも 6 つの C&C サーバから応答を得る事ができた。応答が得られた 6 つの中から実際にコントロールメッセージが来たのは 3 つの C&C サーバで、1 つのサーバから 84 もの IP アドレスに対しての攻撃命令が観測できた (実際スクリーンショットを参照：図 8)。攻撃手法としては、公開された Mirai のソースコードには Proxy Knockback connection と記述されている未実装の攻撃手法を利用しており、おそらくソースコードを入手した攻撃者が独自に命令を実装したものだと考えられる。また、攻撃指令パケットの中に含まれている攻撃先 IP アドレスは 3 つとも同じ IP アドレスのリストだったが、攻撃先 IP アドレスの順序、攻撃パケットが配布される時間間隔が異なってた。特に時間間隔については、3 分おきに定期的に同じ攻撃 IP アドレスリストが配布されるもの、ボット

の登録パケットを C&C が受信してすぐに攻撃指令がくるものも観測できた。同じ攻撃命令を複数のサーバで発信していることから、攻撃者はより高速に攻撃命令を出すために C&C サーバのクラスタリングを行なっている可能性がこの研究により判明した。

(3) 亜種の移り変わり

Mirai のソースコードが公開され [8]、ダウンロードリンクが有効だったのは 48 時間と非常に短い期間であったが Github などのウェブサイト上でミラーが存在し手軽に入手することが可能になっている。この Mirai のソースコードの公開を皮切りとして、現在非常に多くの亜種がネットワーク上に出回っている。亜種は公開されているソースコードを単純にコンパイルして IoT DDoS ボットネットを自前で構築しているものから、探索アルゴリズムの効率化を計ったもの、C&C サーバに変わるアーキテクチャを採用しているものもある。

(4) Windows への対応

オリジナルの Mirai ボットネットは、本来 Linux ベースのファームウェアを搭載して IoT 機器をターゲットにしたものであったが、この亜種ではターゲットとして Windows が動作するデバイスも含まれている。この亜種は、攻撃対象のプロトコルを SSH,Telnet,DCE,Active Directory,MSSQL,MySQL,RDP とオリジナルの Mirai から拡張を行い、感染した Windows デバイスを Linux デバイスを探索するためのスキャナーとして利用を行う。現時点では、Windows デバイスを用いた攻撃は確認されておらずスキャナーの探索性能を向上させる目的での動作しか行われていない。

(5) C&C サーバアドレスの DGA(Domain Generate Algorithm) 化

C&C サーバとマルウェア間のコミュニケーション内でも解説を行なったように、オリジナルの Mirai ではホスト名やポート番号といった C&C サーバへの接続情報は XOR 演算を用いた難読化ルーチンによって行われていたため、バイナリエディタ等では直接それらの情報を抽出する事はできなかった。しかし、XOR 演算に用いられるキーが固定で内部に書き込まれており、一度サーバを特定するとそのサーバがダウンしない限り自由に C&C サーバから攻撃指令の内容を観測する事が可能だった。

DGA(ドメイン生成アルゴリズム) は、動的に C&C サーバのドメインを日付などのあるシードに基づいてドメイン名を生成するルーチンを経由させることによって C&C サーバのコントロールを行いやすくするためのものである。Netlab360 らの報告 [15] によると、tcp/7547,tcp/5555 で動作するルーターのリモート制御に使われる CWMP(CPE WAN Management Protocol) プロトコルをターゲットにした Mirai の亜種が DGA を利用している。

この亜種で利用されている DGA アルゴリズムが利用されるには、ある特定の条件を満たさなければならない。マルウェア本体には C&C サーバへの接続情報が 3 つハードコードさ

れており、それらの接続情報が有効である場合には DGA アルゴリズムは利用されない。具体的な呼び出しの条件の図を以下に示す。

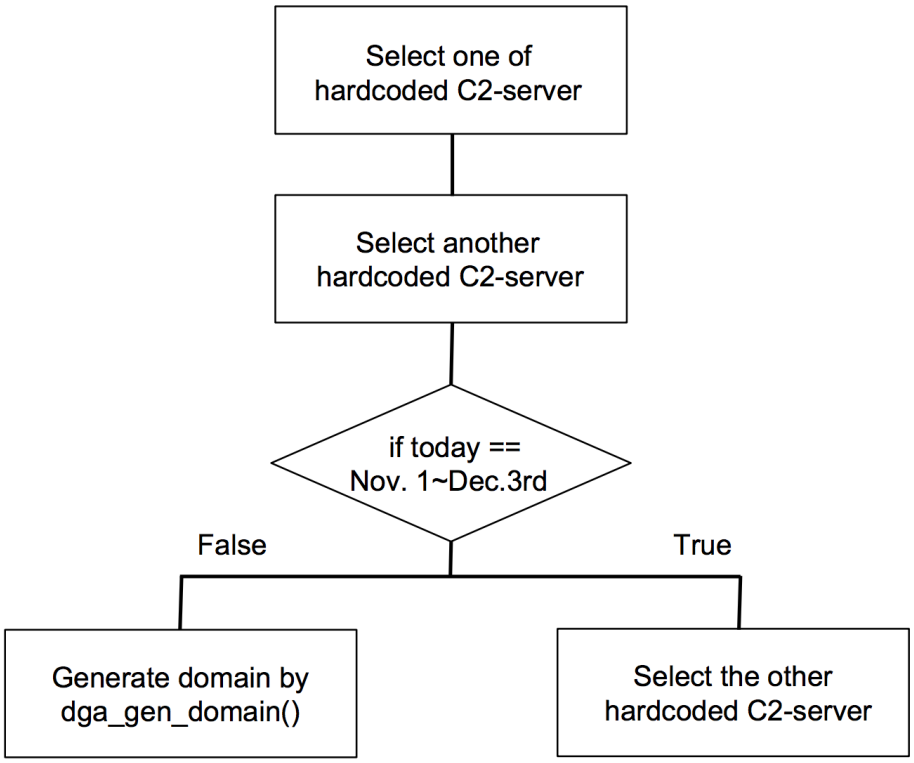

図 2.5: Mirai 亜種に実装されている DGA アルゴリズム

3 つの中の C&C サーバからランダムな値によって一つのサーバが選択される。しかし、接続に失敗した場合はまたランダムな値によってまた新しい C&C サーバが残りの二つから選択されることになる。実装されている DGA 生成関数が呼び出されるタイミングは、この二つの C&C サーバへの接続が失敗した場合に加えて日付が 11/1～12/3 以降に呼び出されることになっている。つまり、C&C サーバへのフォールバックとして DGA アルゴリズムが利用されているという事になる。実装されている DGA のための生成関数は、入力として日付とハードコードされたシード値を利用して第二レベルドメインとしてアルファベットの”a”から”y”を利用した 12 バイト固定長の文字列を生成され、TLD ドメインとしては.online, .tech, .support の三つとなっている。このドメインは 1 日に一回決まったドメインが生成されるようになっている。

Netlab 360 はリバースエンジニアリングによって DGA 生成アルゴリズムを再現し、生成ドメインに対して whois 解決を行った結果を報告している。以下に登録者の確認が行えたドメイン名の表を引用する。この表からは日付によってはドメインが登録されていない状態がある事も知る事ができる。このように、マルウェア全体の活動を C&C サーバが停止してしまってもすぐにマルウェア側で代理のサーバに接続しにいくメカニズムによって継続的に活動が行えるよう機能が追加されている。

date	Level 2 domain	TLD	Registrant Email
12-04	vmdefmnsndoj	.tech	beaba23f49bd4c688faec8a6e5b22a23.protect@whoisguard.com
12-05	xpknpxmyqsr	.online	dlinchkravitz@gmail.com
12-06	lvfjcwwobycj	.tech	ac4ca107e04e4d58ad1348d5d759b3b0.protect@whoisguard.com
12-07	nympompksmfx	.tech	dlinchkravitz@gmail.com
12-08	kedbuffigfjs	.online	dlinchkravitz@gmail.com
12-14	bwhrdaumwuvn	.online	dlinchkravitz@gmail.com
12-19	bpmsfckfkrpr	.online	dlinchkravitz@gmail.com
12-20	oornsduuwjli	.tech	dlinchkravitz@gmail.com
12-21	qjqubpciajoc	.tech	dlinchkravitz@gmail.com
12-22	exvfaajegjur	.online	dlinchkravitz@gmail.com
12-24	poorcetnmjfc	.online	dlinchkravitz@gmail.com
12-31	vtrndmhsgada	.online	vtrndmhsgada.online@domainsbyproxy.com

図 2.6: Mirai 亜種に実装された DGA アルゴリズムの日毎の出力結果と whois 結果 (Netlab 360 Blog[15] から引用)

(6) DHT/uTP を C&C サーバに用いたもの (Hajime, Rex 等)

Hajime は RapidlityNetworks[10] によって発見された IoT ボットネットであり、Mirai と非常によく似た構成を持っている。このボットネットは主に TR-069 と呼ばれるルーターで用いられるプロトコルをターゲットに感染を行う。しかし、アーキテクチャ的に Mirai と大きく異なるのは、Mirai DDoS ボットネットは C&C サーバを利用するのに対して Hajime では BitTorrent Mainline DHT[2] を用いたボット間のコミュニケーションを行う。DHT(Distributed Hash Table, 分散ハッシュテーブル) とは、P2P ネットワーク上のノードで構成される分散化されたハッシュテーブルである。

DHT は、Key-value ペアをネットワークを構成しているピアが持ち合って構成されている。DHT としては様々な Chord[31], Kademlia[24] 等様々なアルゴリズムが提案されているが、共通点としてノードへの到達オーダーが定義できるデータ構造をトポロジーとして採用し、ノード ID が割り振られている点、トポロジー内でノード間の距離メトリックが定義されている点、その距離メトリックにおいてファイルのハッシュ関数を取ったものとしてキーとして扱い、算出されたキー値に一番近いノード ID を持つノードに key-value ペアが格納される点が共通している。

Hajime では、Kademlia DHT をベースに本来、BitTorrent におけるトラッカーの役割を P2P ネットワークによって分散化し、トラッカーサイトのテイクダウンによる単一障害点を除去するために設計された BitTorrent Mainline DHT を利用する形で C&C サーバの役割を果たしている。そのため、Hajime においても同様の仕組みが採用されている事は攻撃指令を行うサーバが存在せず攻撃を停止させる事が非常に難しくなっていることを意味する。

Kademlia の探索アルゴリズムを以下に示す。Kademlia は二分木ベースのトポロジーを採用しており、ファイルに関する情報を取得したい場合はファイルのハッシュ値を取り (Kademlia では SHA-1)、自身のルーティングテーブルの中で一番近いノード ID(Kademlia の場合にはノード間の XOR を距離として定義) を持つノードに対して問い合わせのメッセージを送る。

受信したノードは自身の中に対象の Key-value ペアがないことを確認すると、自身のルーティングテーブルのエントリの中から受信したハッシュ値に最も距離の近いノードを見つけ、問い合わせてきたノードに送信する。問い合わせ結果が返ってきたノードは、反復的に同じメッセージを結果として返ってきたノードに対して送信する事を繰り返すことによってメッセージを発見する仕組みになっている。

2.3 DOTS(DDoS Open Threat Signaling) プロトコル

DOTS(DDoS Open Threat Signaling) プロトコル [25][28][26][29] は、DDoS 攻撃に関する情報を素早く伝達を行うためのプロトコル仕様であり、現在 IETF ドラフトとして標準化に向けた提案が行われている。従来ではメール等を用いた手法が DDoS 攻撃のシグナリング手法として主に用いられていたが、メールによる DDoS 攻撃情報の伝達には攻撃を受けている個人または組織と、攻撃に対処する組織との間で必ず人間の目を介在させる必要性があり、メールを用いた仕組みは攻撃の大規模化と高度化が進んだ DDoS 攻撃に対応するためには大きな時間とコストがかかってしまう。DOTS はこのような問題を解決することを目的として策定が行われており、それぞれの組織に DOTS プロトコルを実装したソフトウェアまたはハードウェアを設置することによって、機器を通じてシグナリングを受信することができる。ここでは、IETF ドラフトによって策定が行われている DOTS プロトコルを用いたアーキテクチャとドラフト内で記述されているユースケースについて記述を行う。

2.3.1 概要

DOTS プロトコルには主に二つのコンポーネントがあり、その二つは DOTS サーバと DOTS クライアントと呼ばれている。ドラフト内で DOTS プロトコルの定義する範囲は原則としてプロトコルの拡張等を除いてこの DOTS サーバと DOTS クライアントの間のコミュニケーションのみを定義する、としている。[26]

DOTS プロトコルにおいて、DOTS クライアントと DOTS サーバの確立するリンクは二種類存在する。これはシグナルチャンネルと、データチャンネルと呼ばれておりシグナルチャンネルは主にリンクの生存確認、攻撃緩和要請を送信するためのリンクであり、一方のデータチャンネルはシグナルチャンネルを用いて攻撃の緩和要請等を送信する前に事前に 2 コンポーネント間で共有する設定情報等を送受信するためのチャンネルになっている。DOTS サーバと DOTS クライアント間で確立されたリンクは二種類あるものの、そのリンクがアクティブであることを確認するためには、たとえデータチャンネルが切断されていても、シグナルチャンネルがアクティブであれば、リンクがアクティブであるという判定を行うことになっている。[25] この 2 コンポーネント間における認証手法等は、DOTS プロトコルのアーキテクチャドラフト内では定義されていないがシグナルチャンネルの認証には CoAP over DTLS が用いられており、証明書を用いて相互認証を行うことが可能になっている。

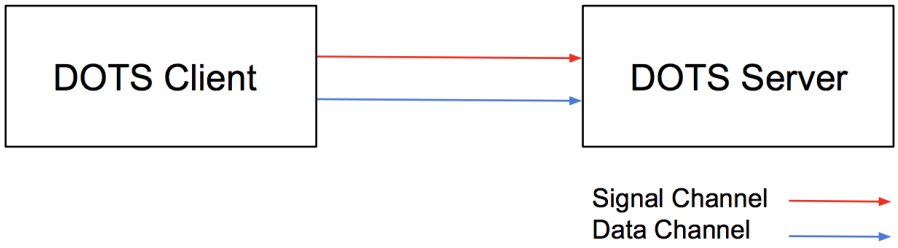

図 2.7: DOTS の最も基本的なユースケース

2.3.2 DOTS プロトコルを用いたユースケース

ドラフトの中では、DOTS プロトコルを用いた代表的なユースケースについていくつか言及されている。DOTS のプロトコル仕様を理解するためにいくつかの例を紹介する。

(1) 上流トランジットプロバイダが DOTS サーバを提供する想定

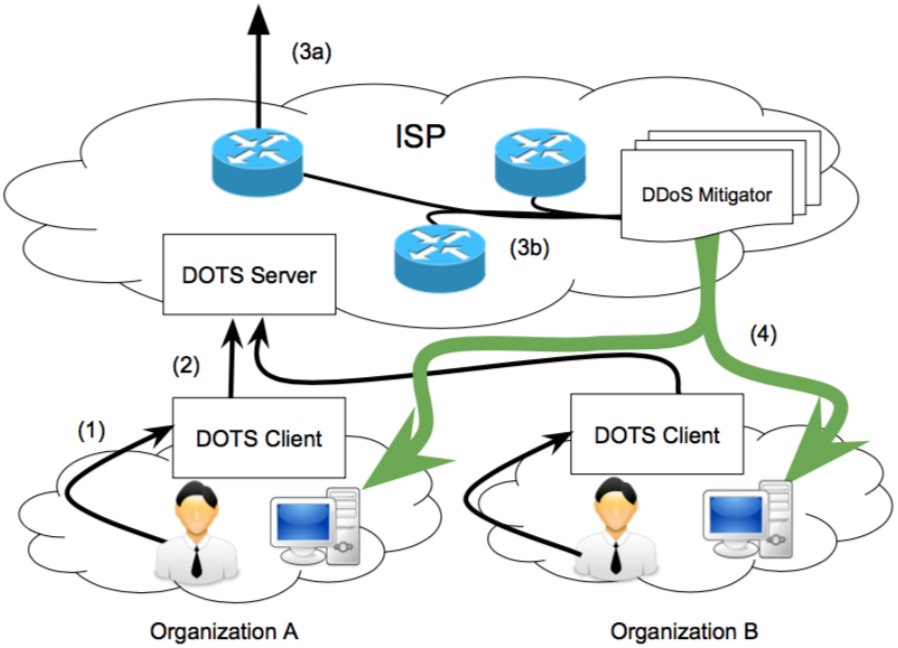

図 2.8: BGP 経路注入を用いたトラフィックの緩和手法の DOTS の活用例

Web サーバ、DOTS クライアント、権威 DNS、IDMS(DDoS 緩和システム) があるネットワーク環境と、DDoS 緩和サービスを代理してくれるトランジットプロバイダとの契約を行なっているという前提環境において、DDoS 攻撃を受けた場合最初は自ネットワーク内の IDMS 製品を用いて攻撃の緩和を行うが、その後自ネットワークの設備だけでは対処できな

くなくなった場合はDOTSクライアントを用いてトランジットプロバイダのDOTSサーバに対して緩和を行うサブネットや、プロトコル種別等を報告する。報告を受けとったトランジットプロバイダは、RTBH(Remotely-Triggered Black-Hole) の技術を用いて経路情報の書き換えを行い、該当攻撃トラフィックはトランジットプロバイダ内のDDoS緩和設備へとルーティングを行なうことによってDDoS攻撃の緩和を援護する形になる。

また、上の例では単一のトランジットプロバイダの前提でDOTSプロトコルを用いたユースケースを紹介したが、複数のDOTS対応のトランジットプロバイダとの接続を行うことにより、マルチホーム構成でのDOTSプロトコルを用いたDDoS緩和サービスを利用する例もユースケースとして挙げられている。

(2)　複数プロバイダが DOTS サーバを提供するマルチホーム環境の想定

複数の冗長化されたインターネット接続を持つネットワークの環境において、DOTSによるDDoS攻撃をシグナリングを行って接続を遮断。ドラフト内ではこの複数のDOTSサーバ間でステータス情報をやりとりし、効率的なDDoS攻撃を行うという記述があるが、現在の技術仕様の中ではこの機能は確認されていない。

(3)　スモールオフィスや SOHO などの環境における DDoS 攻撃の対処を行う想定

スマートフォン向けのアプリケーション等でDOTSプロトコルを実装したクライアントアプリケーションを作り、スモールオフィス等の実際にそのようなアプライアンスを設置する事が困難な場合でも、ネットワーク事業者もしくはMSSP(DDoS Mitigation Managed Security Service Provider) 業者に対して報告を行う手動によるシグナリングとしても機能できる。

(4)　内部ネットワーク内での DDoS 攻撃ノードの停止要求を行う想定

上記のユースケースでは、インタードメインにおけるシグナリングのユースケースが主であった。DOTS自体はDOTSサーバとDOTSクライアントをDOTSプロトコルで接続したものであるため、あらゆる箇所での応用が可能になっている。この例ではユーザーのアクセス回線からDDoS攻撃トラフィックが生成されている環境を想定して、DOTSクライアントによってDDoS攻撃トラフィックを内側から防止できるというようなユースケースが定義されている。

2.3.3　シグナルチャンネル

シグナルチャンネルは、DOTSにおけるリンクの確立確認 (ハートビート)、DDoS攻撃の緩和を伝達するためのメッセージを送信するためのチャンネルである。DLTS over UDP/TLS

over TCP 上の伝送路の上で CoAP プロトコル [30] が用いられる、デフォルトポート番号は 5684 と定義されている。シグナリングのメッセージ形式は全て JSON メッセージを CBOR(Concise Binary Object Representation) 形式 [32] によってシリアライズされた上で送信される。CoAP は HTTP に似たメソッドを持っており、このメソッドの使い分けによって DOTS クライアントと DOTS サーバはシグナリングを行い、ハートビートのインタバールやタイムアウト等の設定等を変更できる。

2.3.4 データチャンネル

データチャンネルは、DDoS 攻撃の緩和要請メッセージがやりとりされる前段階で交換されるべきメッセージを送受信するためのチャンネルであり、TLS 上に確立された NETCONF データストアを操作するための RESTCONF(RFC8040) と呼ばれる HTTP ベースのプロトコルが採用されている。

2.3.5 先行実装

DOTS クライアント、DOTS サーバ、DOTS プロトコルの先行実装として 2018 年 1 月 17 日現時点では、Go 言語で実装された go-dots[9] が存在する。DOTS には具体的な実装が存在しなかったため、go-dots に実装された仕様等は実際に DOTS IETF ドラフトのほうへフィードバックが行われ更新が頻繁に行われている。今回は、DOTS クライアントやサーバの実装等は行わないためシグナリングを行う際にはこの実装を利用して評価を行うことにする。

2.4 RIPE Atlas

この節では、本論文の提案手法の一部である定点観測プローブに対して先行事例として RIPE Atlas プロジェクトについての記述を行う。

2.4.1 概要

RIPE Atlas[16] は、定点観測プローブを用いた分散型のインターネット計測プロジェクトである。2010 年に RIPE Network Coordination Centre(以下 RIPE NCC) によって開始された。基本的なプロジェクトのモデルとしては、ボランティアに対してインターネットに接続されたプローブを無償にて配布する。配布したプローブを元に RIPE NCC は世界中のインターネットの疎通を確認している。現在 2017 年 12 月現在で 1 万人を超えるユーザーがプローブを設置してプロジェクトに参加しているが、参加者には参加へのインセンティブとして稼働した時間に比例してクレジットが与えられる。この与えられたクレジットを消費することによってその参加者は RIPE Atlas プロジェクトに参加している他の定点観測プローブ

表 2.6: RIPE Atlas プローブスペック表

	Device	OS	CPU	Memory	Ethernet Interface
V1	Lantronix XPort Pro	uClinux-based	Lantronix DSTni-FX (no MMU)	8MB RAM, 16MB Flash	100Base-TX
V2	Lantronix XPort Pro	uClinux-based	Lantronix DSTni-FX (no MMU)	16MB RAM, 16MB Flash	100Base-TX
V3	TP-Link TL-MR3020	OpenWRT-based	MIPS + MMU	32MB RAM, 4MB-flash, 4GB-USB stick	100Base-TX
V4	NanoPi NEO Plus 2	Unknown	ARM Cortex-A53x4 + MMU	512MB RAM	100Base-TX, 1000Base-T
Anchor	Soekris net6501-70	CentOS-based			

に対して DNS や ICMP パケットなどを送信するようにリクエストする事ができる。このような非常に巧妙なインセンティブ設計によって分散型のプローブネットワークを自主的に設置を行なってくれるボランティアを募る形で形成している。

2.4.2 プローブ

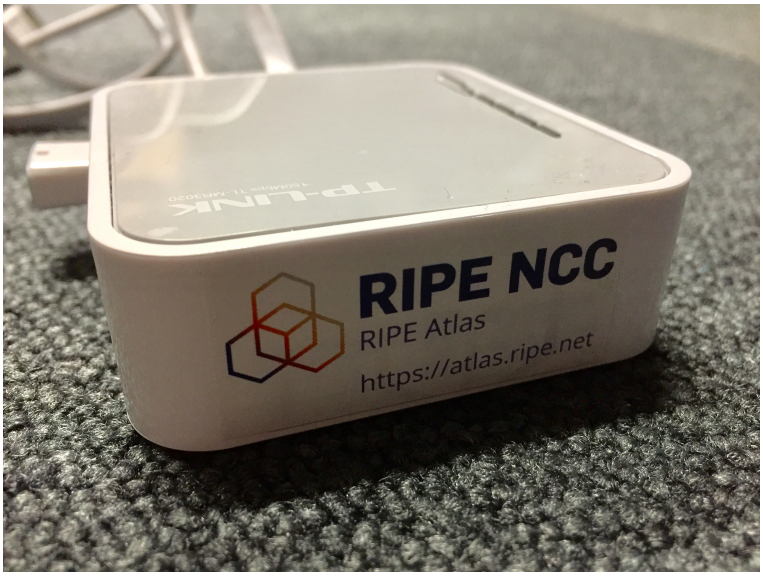

図 2.9: RIPE Atlas のプローブ (現在のバージョン 3)

RIPE Atlas で用いられているプローブにはいくつかのバージョンが存在しており、2017 年 12 月現時点では 3 番目となるバージョン 3 のプローブの配布が行われている。

当初、RIPE Atlas プロジェクトはプローブとして Lantronix 社製の XPort を採用していた。しかし、MMU が搭載されていないことから初期の 8MB メモリ版ではメモリフラグメントの問題が頻繁に生じたことで 1 日おきに再起動を繰り返さなければならなかった。16MB メモリ版への変更を行なったが XPort を使ったプローブでは SSH コネクションを確立するのに 30 秒程度の時間がかかるため、インターネット計測を目的とするこのプロジェクトではスペック要件を満たせなかったことが考えられる。

第二世代では、TP-LINK 社製の TL-MR3020 を採用している。CPU 性能は以前のプローブと比較して大幅に向上したが内部フラッシュメモリが 4MB 程度だったため搭載されていた外部の USB スティックにソフトウェアが格納されている。内部フラッシュメモリには、新

しいバージョンのファームウェアを取得し USB スティックへと書き込むコードが書き込まれており、この分離された仕組みにより USB スティックが破損した場合でもファームウェアを取得して正常な動作を回復できるようになるという利点がある。[17]

2.4.3 中央サーバ

1 万台を超える世界中に散らばったプローブからの頻繁に送信されるデータを処理する基盤の設計に関して RIPE Atlas の方針としては以下の図で示されたアーキテクチャを採用するとしている。プローブがネットワークへ参加するためには最初に登録サーバと呼ばれるサーバにおいて認証を行い、登録サーバはプローブに対して結果の送信を行うためのサーバ情報を返信し、プローブはこれに接続を行う。内部では、プローブから収集されたデータはメッセージキューに蓄積され、各コンポーネントがメッセージキューと接続しストリーミングやデータ解析、データベースへの蓄積を行うために利用されている。

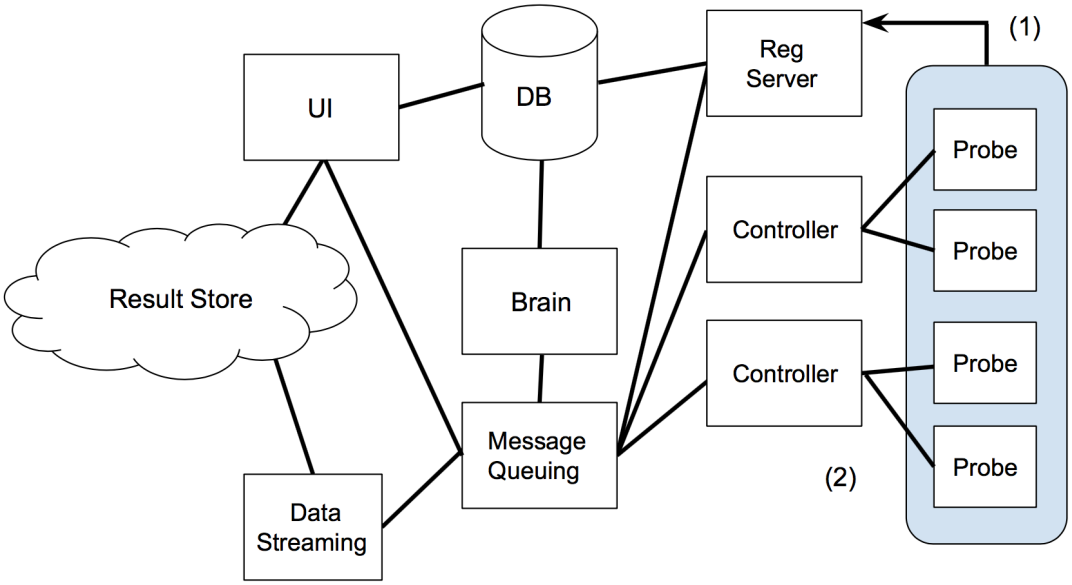

図 2.10: RIPE Atlas 全体のアーキテクチャ図

2.4.4 セキュリティ

定点観測プローブを用いることによって集計した情報をいかにして信用するか、虚偽の問題に対して対処するかといった問題は同様のアーキテクチャにおいて常に浮上してくる問題である。MDSec による RIPE Atlas のリバースエンジニアリング [13] において RIPE Atlas では以下の点においてセキュリティ対策が行われている事が判明した。

1. ファームウェアアップデート時におけるアップデートデータの改ざんの防止

2. OS と USB メモリ内のファイルシステムの暗号化
3. 内部のソフトウェアはすべて RIPE Atlas のクラウドを経由して実行される
4. 結果の送信時に機器ごとに個別の暗号鍵を保持しており、万が一の場合でも全体の影響を最小限に食い止める

このようなプローブ型の計測ネットワークにおけるセキュリティの問題は常に存在しており、RIPE Atlas では必要最低限のセキュリティ対策を行いつつ、なおかつ被害があった場合でも固有の暗号鍵を機器ごとに割り当てる等の対処によって被害端末を特定し、迅速に停止させる事が可能になっている。

第3章　提案手法

この章では、本論文で挙げられた問題点を解決し目的を達成するための手法として、DOTSと観測プローブを用いてIoT機器を用いたDDoSボットネットによる攻撃を観測し、迅速に通知を行うコミュニティーベースのアーキテクチャを提案する。本提案手法におけるユースケースを定義し、本論文の提案アーキテクチャとそのメッセージング等について議論を行う。

3.1　提案手法の概要

提案を行うにあたって、第1章の研究の目的で述べた本提案手法の目的とするゴールを再度確認することに加え、本提案手法の大まかな処理の流れについて記述を行う。

本論文では、IoT機器を用いたDDoSボットネットによる感染・攻撃活動を初期段階で迅速に観測し、自動化との親和性の高い観測アラートシステムを構築することを目的としている。本論文で提案する観測アラートシステムでは、図1.1に示されている通り、ネットワークへのスキャン活動を監視するための環境を提供する参加者に対してDDoS観測プローブを設置してもらい、そのインセンティブとして参加者間で観測プローブを用いたデータを利用したDDoS攻撃に関するシグナリングを受け取るためのシステムである。

観測プローブを導入することによって、以下の利点が挙げられる。

- 初期感染活動が行われやすい国や地域に対して集中的な観測活動が可能にになる
- 参加ユーザーに対しての設置コストが非常に少ない
- ソフトウェアで動作する観測プログラムと比較して、データの改ざんなどが比較的難易度が高い

図1.1を分類すると、(1) プローブから中央のサーバへの観測結果の送信、(2) 中央のサーバでのデータの解析、参加ユーザーのシグナリング対象のデータかどうかのマッチング処理、(3)DOTS等のシグナリングプロトコルを用いた参加者に対するシグナリングの三つに分類される。

プローブから送信されるデータとしては、プローブで動作しているハニーポットと観測プローブに対して到達したものからデータを得る。取得するデータに関しての詳細は各コンポーネントごとの項目にて解説を行うが、マルウェアのダウンロード先として指定されたIPアドレスや、感染のためのスキャン活動を行うための探索パケットなどである。データの送信を中央のサーバに対して行うと、中央サーバはそれらのデータに対して集計処理を行い、参加者があらかじめ設定した条件に合わせてシグナリングを行うという構成になっている。

これによって、参加者はマルウェアの配布サーバのダウンロード先情報やポート番号ごとのスキャン活動動向などをリアルタイムに把握することができ、それらをフィルタールールとして応用することや、攻撃ノードを外部からの観測を受けて停止を行う手助けを行うシグナリングも可能になる。

3.2 本提案アーキテクチャのユースケース

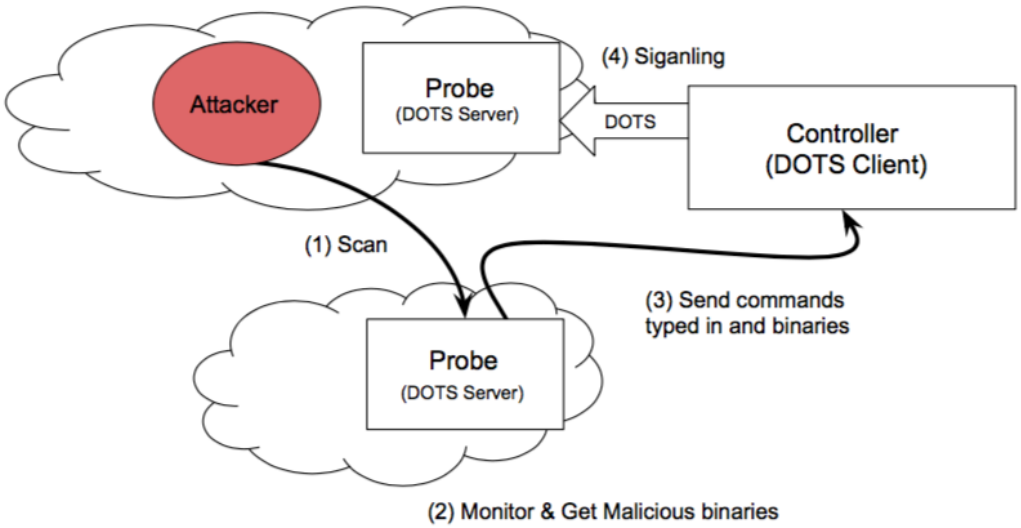

図 3.1: 提案アーキテクチャの基本的な流れ

本手法のターゲットとするユースケースを定義する。ユースケースとしては以下の 3 つが考えられる。

3.2.1 特定のポートに対する攻撃パケットの動向の把握と対策

ネットワーク上に流れる不審なパケットの中には、時間とともにネットワーク上に出回るパケットの種類は特徴がある。例としては、特定のポートで動作するあるサービスに対して重大な脆弱性が発見された場合には、非常に多くの探索パケットがネットワーク上のまったく関係のないホストに到達する。観測プローブはこの動きを中央のサーバに定期的に送信し、IP アドレスやポートにおけるレピュテーションを行う。これによって、素早くネットワークの上での不審な動向を把握することが可能になる。

3.2.2 マルウェアのダウンロードサーバや経由サーバの遮断

定点観測を行うプローブの中には、ハニーポットモジュールが内臓されており一連の IoT マルウェアの動きをトレースすることが可能になっている。このトレース情報から配信サー

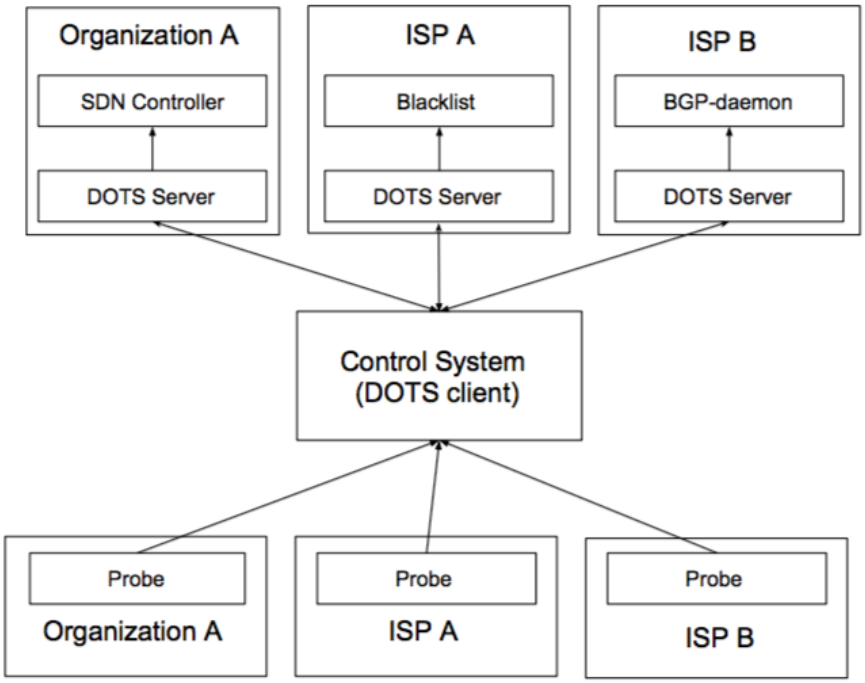

図 3.2: 提案プローブの各設置者におけるデータの取り扱い

バからマルウェア本体をダウンロードする際の URL を取得し、プローブは中央のコントロールサーバへと送信を行う。この取得したダウンロード URL を利用して ISP やネットワーク管理者は自ネットワーク内の IoT 機器に対して不審な IP アドレスへの接続を遮断でき、IoT 機器が新しくボットネットへと感染する可能性を未然に防げる可能性を上げることができる。

3.2.3　ネットワーク管理者への DOTS を用いたシグナリング

IoT ボットネットに感染している IoT 機器が IP アドレスを詐称していない前提で、感染している機器が所属しているネットワークに対して所属ネットワーク内に IoT ボットネットに感染してスキャンを行なっていることを DOTS を用いて通知を行う。相手が DOTS サーバを中央システムに登録して、自分のネットワークからの攻撃パケットを見ることによって攻撃ノードの影響を最小限に止めることが可能になる。詐称した IP アドレスをどのように特定するかなどは、本論文の対象外とする。

3.3　提案アーキテクチャ

IoT 機器を利用したボットネットが、スキャン活動や攻撃活動を行なった際に IoT 機器に特化したハニーポットを動作させているプローブにその記録が残ると、プローブは各プローブのハニーポットに関する設定情報や鍵を管理している中央システムにデータを送信する。

中央システムはデータを集計し、プローブを所有して継続的に利用可能なデータを送信しているプローブ群に対して、その記録を行なったプローブのデータを公開する。もし参加しているネットワーク事業者またはネットワーク管理者の中に、観測した攻撃を送信した感染ノードを管理している管理者が存在している場合は、プローブに実装されている DOTS サーバに対して中央サーバ内に実装されている DOTS クライアント経由で緩和リクエストを送信する。受け取った DOTS リクエストは、プローブ上で動作している HTTP サーバ上でアクセス可能な他、中央システム上の設定において緩和リクエストを送る DOTS サーバの IP アドレスとポート番号を記述しておくことによりプローブを経由せずに DOTS サーバからのシグナリングを受信することも可能。取得後、得られた情報を利用して各管理者は自社のブラックリストへの追加等の対処が可能になる。

3.4 観測プローブ

観測を行うプローブには、大まかに以下のモジュールを搭載する。

- IoT ボットネットのスキャン活動を観測するハニーポットモジュール
- パケットを観測するためのモジュール
- デジタル署名などを検証するための鍵情報

3.4.1 IoT ボットネットの活動を観測するハニーポットモジュール

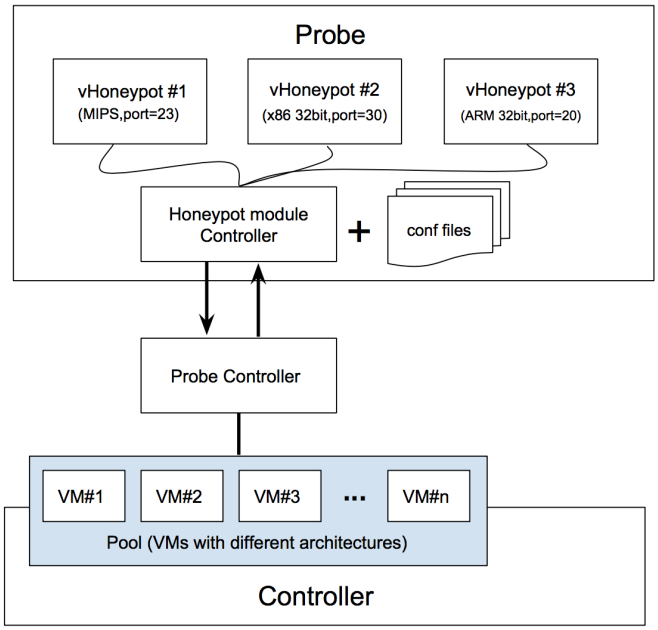

図 3.3: 本提案手法のプローブ上で動作するハニーポットの全体アーキテクチャ

IoT ボットネットの活動を観測するためのハニーポットモジュールを構築する際に、現状多くの IoT ボットネットには以下の 4 つの構造上の共通点があることに着目した上で感染デバイスの情報とマルウェアの動作と本体を収集するハニーポットを設計する。IoT 機器は組み込み機器として開発されている機器が多く、DDoS ボットネットの開発には幅広い種類の CPU アーキテクチャに対応したソフトウェアの開発が要求され、観測プローブにとってもコマンドの取得やマルウェアのバイナリの取得に関して幅広いアーキテクチャに柔軟に対応できる設計が求められる。

- インターネット上のアドレス空間を探索
- 脆弱性のあるサービスまたは非常に単純な認証情報が設定されている IoT 機器に外部より侵入を試みる
- 標的デバイスのアーキテクチャを判別後、マルウェアのファイルを外部よりダウンロードし、実行
- C&C サーバからの命令を待機し、命令に応じて攻撃を実行

ハニーポットモジュールの構成図を以下に示す。プローブ上でハニーポットは異なる CPU アーキテクチャのハニーポットを複数台動作させる事が可能であり、ハニーポットに対する入力やログイン試行はすべて記録され、感染デバイスからプローブへの入出力はサンドボックス化された中央サーバ内の VM プールへとサニタイズ等を行なった上でフォワードされる。その際、プローブ中のハニーポットモジュールは TCP 接続のハンドリングを行う。この VM プールには異なる CPU アーキテクチャを持つ OS イメージが管理されており、実際の応答を返すようになっている。ハニーポットの感染デバイスへの応答には、VM プールからの応答を返すものと特定の反応対象のコマンドとそのコマンドに対する応答を組に持った定義ファイルを中央サーバから取得し、それを参照する方法の二種類があり、この二つを組み合わせる事も可能になっている。しかし、一つのプローブの中に複数のハニーポットがあるという探索プローブの特徴を外部からなるべく気づかれないようにするため、一度に応答できるハニーポットは一個の仮想ハニーポットのみにするとする。

また、telnet を用いたコミュニケーションにおいて横浜国立大学の Yin Minn Pa Pa らは、TCP 3-way handshake 後にバナー表示フェーズ、認証フェーズ、コマンド対話フェーズの三段階があると言及している。[27] 具体的には、telnet クライアントにて telnet デーモンが動作しているサーバへ接続を行うと、認証を行ったり、ウェルカムメッセージ等の表示を行う画面が表示される。これがバナー表示フェースである。認証フェーズにおいて、telnet が動作しているサーバでパスワードが一致するかの検証を行い、認証が成功するとコマンド対話フェーズに移行する。認証フェーズが終わると、コマンド対話フェーズにおいてコマンドとそのコマンドに対応する処理が行われるという全体の流れがある。

今回、実際の IoT 機器のバナーを研究向けにポートスキャンした結果を公開している Censys[4] を利用し、実際の telnet サービスのバナーを収集した。収集したバナーはプローブコントローラーから送信された定義ファイルによってプローブ中のハニーポットに配布される。また、入力されたコマンドに対する応答を定義ファイルへとまとめる。定義ファイルはプローブコントローラーに蓄積され、各プローブに配布されていく。この定義ファイルは

VM プール内の VM に対するアクセスを最小限にとどめ、中央サーバの負荷軽減とセキュリティリスクに対して効果がある。

3.4.2 中央のコントロールサーバ

中央システムのアーキテクチャとしては、基本的に RIPE Atlas のサーバサイドアーキテクチャを基にしたアーキテクチャを構築する。RIPE Atlas の現在の接続プローブ数は 2017 年 12 月 21 日現在で、24000 台。データセンター向けの定点観測プローブである Anchor は 299 台が稼働しており、5,622 件/秒の速度でデータの処理が行われている。中央システムのアーキテクチャを以下に示す。

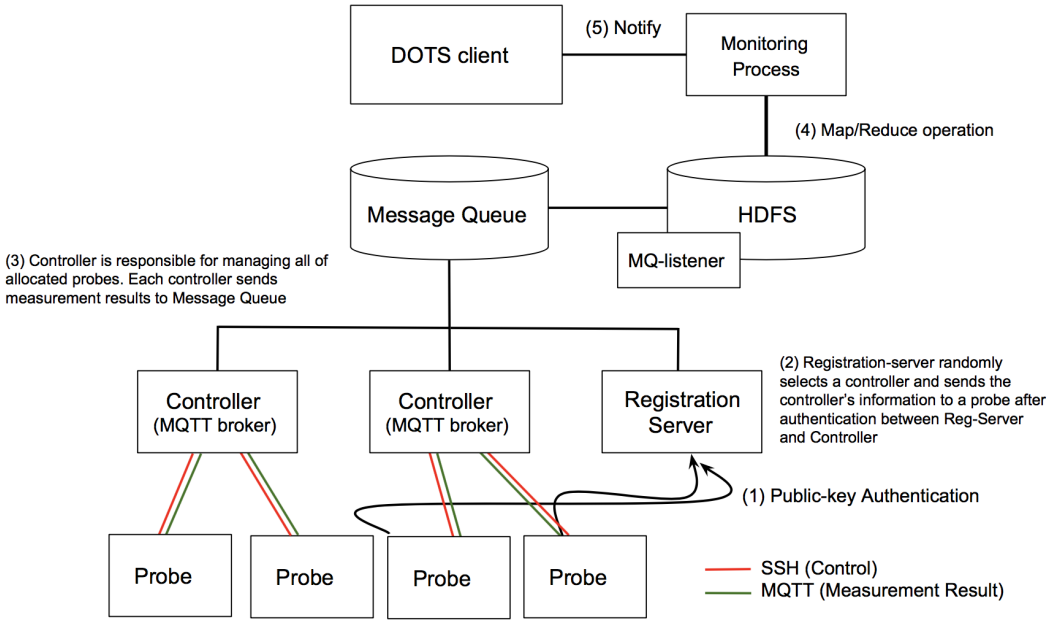

図 3.4: 提案アーキテクチャ

中央サーバは大きくわけて、登録サーバと管理サーバ、メッセージキューと追記型のデータベースで構成されている。大量のプローブを一つのサーバで管理するアーキテクチャの場合、大規模な観測ネットワークを構築する上で不適当なので鍵認証と複数ある管理サーバへの割り当てを担当する登録サーバのみを単一での設置を行う。それ以降、観測データや管理コマンドのやりとりはすべて割り当てられた管理サーバとプローブの間で行われる。

プローブから管理サーバへ観測データを送信するときに用いられるプロトコルには MQTT(MQ Telemetry Transport) プロトコルを用いる。TCP ベースの Pub-Sub 型のメッセージをやりとりするための軽量プロトコルであり、HTTP 等の他の汎用のプロトコルと比較してもオーバーヘッドが少ない。プローブと管理サーバとの間の通信は原則としてユーザー保護の観点から通信路を暗号化する必要があるが、MQTT は SSL を用いた暗号化通信に対応しているためこの条件を満たした今回のケースに適切なプロトコルであると考えられる。

観測データを受け取った各管理サーバはすべてのデータを一旦格納するためにメッセージキューに対してデータをエンキューする。メッセージキューはクラスタリング可能なものを利用し、各コンポーネントはメッセージキューからデータを取り出す（デキュー）を行うことによって処理を行う。データベースには、大規模なデータを効率的かつ高速に扱えるように追記型のデータベースを利用する。

3.5 セキュリティ

観測型プローブを利用することにおいて、セキュリティの確保は重要であり本論文の提案手法に関するセキュリティは多岐にわたるが本論文では、関連研究の RIPE Atlas プロジェクトで実際に利用されているセキュリティ機構を参考にした上で以下のようなセキュリティ対策を行う。

- 公開鍵認証を利用したプローブの認証
- ファイルシステムの暗号化
- デジタル署名を用いたファームウェアのアップデート
- DOTS プロトコルのシグナルチャンネルとデータチャンネルに対して証明書を利用した暗号化通信を利用
- 鍵などはすべてプローブごとに個別の鍵を利用することによってすぐに悪性ノードを遮断できるようにする

第4章　実装

この章では、3章の本論文における提案手法の評価を行うために提案するアーキテクチャに基づいた実装を行う。

4.1　実装の概要

本研究では、第1章の研究の目的で述べた本提案手法を構成する3つの要素のうち、プローブと中央サーバアーキテクチャの二項目に関する実装を行う。より具体的には、プローブと中央サーバ間での認証・管理手法、中央サーバのアーキテクチャとデータの処理に関して次項で言及する本論文で定義を行った機能要件を満たした実装を行うこととする。

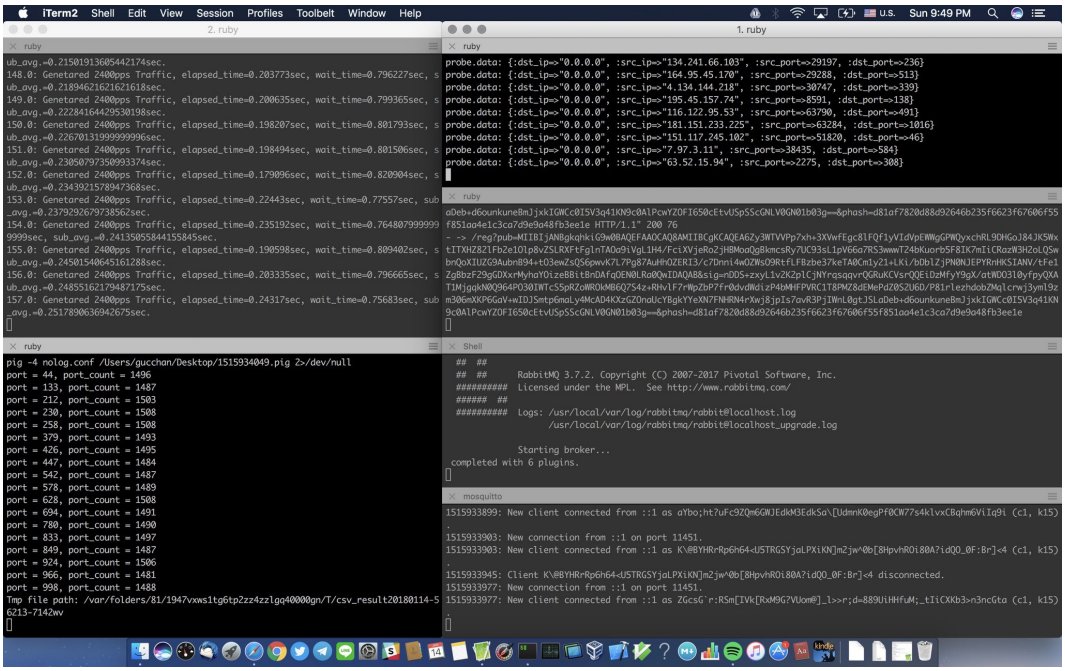

図 4.1: ローカルで動作をしているサーバアーキテクチャ

4.2　機能要件

本論文の評価を取るにあたってまず最初に機能要件を定義した上でその基準を満たす実装を行う。

表 4.1: 検証に利用したコンピュータ

MacBook ProCore i5 2.9GHz, 16GB RAM, 250GB SSD	macOS Sierra 10.12.6
VPS1x86_64, 2GB RAM, 200GB HDD	Ubuntu 4.4.0-97-generic
VPS2x86_64, 512MB RAM, 20GB SSD	Ubuntu 4.4.0-97-generic

表 4.2: 検証に利用したライブラリ群

MQTT Broker	Mosquito version 1.4.14
Message Queue	RabbitMQ version 3.7.2
Hadoop	Hadoop version 2.8.2
Apache Pig	Apache Pig 0.16.0

- 今回は、実際に観測プローブをデプロイした実験は行わず中央サーバアーキテクチャに対して擬似パケットを送信するシミュレーションの形をとる
- データを処理するためのデータベースは基本的には追記ベースのものを利用し、カウンティングを主に高速で行えるものを利用する

4.3 中央システムのアーキテクチャ

中央システムに用いられている各コンポーネントには以下の性能とバージョンの構成のコンピュータとソフトウェアを利用した。また、各コンポーネントのサーバやプログラムは Ruby で実装を行なった。

4.3.1 公開鍵認証とデジタル署名によるプローブの認証とネットワークへの参加

観測プローブが観測ネットワークに接続するために、AUTH ADD_PROBE CTRL_KEY CTRL_JOIN の四つのメッセージを定義し、これを利用する。観測ネットワークへの接続の際にはプローブソフトウェアに内蔵されている鍵を使って登録サーバと通信を行い、認証を行う必要がある。これは、AUTH メッセージを使ってデジタル署名を検証することによって構成されている。登録サーバは、配布済みのプローブの公開鍵情報をテーブルで保持しており登録されていない公開鍵が遅れられてきた場合には認証は行われない。内部で保持している公開鍵の形式は RFC4716 の Base64 エンコード部を保持しており、必要に応じてサーバの内部でデコードを行い鍵データを使ってデジタル署名を検証する。

署名が正しいものであると確認できた後は、登録サーバはランダムにコントロールサーバを選択し、選択されたコントロールサーバに対して ADD_PROBE メッセージを送信することによってプローブの鍵情報を登録する。鍵登録がコントロールサーバによって許可された場合はコントロールサーバとプローブの間で管理用に確立される SSH 接続用の公開鍵がレスポンスとして返却する。ADD_PROBE メッセージがコントロールサーバによって受け入れられなかった場合は、登録サーバはまた新しい接続先をランダムに決定する。

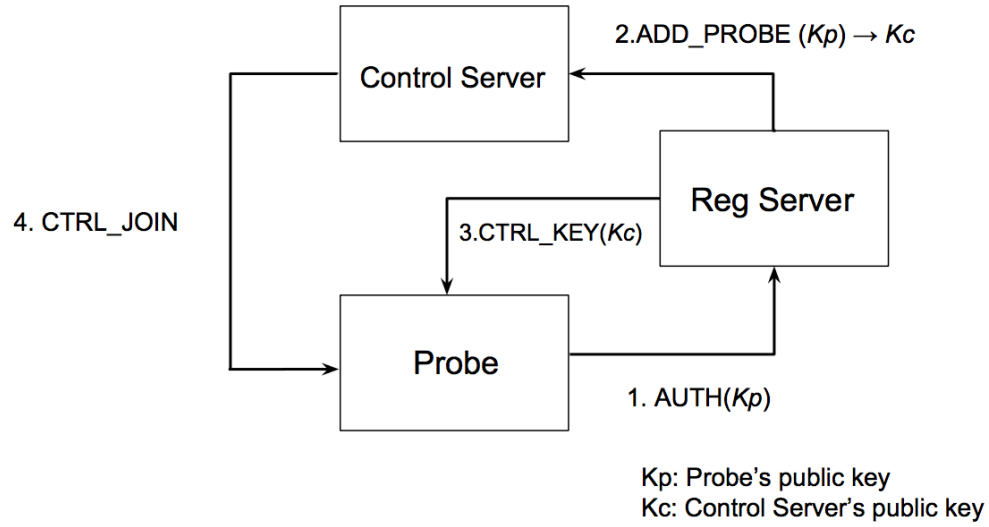

図 4.2: プローブ認証と計測の開始までのメッセージのやりとり

受け入れ先のコントロールサーバの鍵情報を得た登録サーバはこの情報を CTRL_KEY メッセージとしてデジタル署名を用いてプローブに伝え、プローブはコントロールサーバからの SSH 接続を受け入れる処理を行う。今回は簡単化のため authorized_keys ファイルに書き込みを行うことで実装を行った。Syslog からコントロールサーバからの SSH セッションの確立が行われたことを確認するとプローブとコントロールサーバは CTRL_JOIN メッセージを送信し、コントロールサーバの MQTT ブローカーに対して報告を行う準備ができたことを伝える。

4.3.2 プローブ

上記の鍵交換を行った後、MQTT セッションを通じて SSL を用いた暗号化通信の上でプローブは観測データをコントロールサーバに向けて送信する。今回は、実際のインターネット上からの探索パケットを実際に取得する方式ではないため予約、ループバックアドレスと特定の IP アドレスブロックを除外したアドレスレンジから IP アドレスと well-known(11024) ポート番号の組をランダムに一定の pps(packets per second) で生成して MQTT ブローカーであるコントロールサーバに送信する実装を行った。

4.3.3 コントロールサーバ

コントロールサーバの役割は、(1)SSH コネクションをプローブとの間で確立し、プローブを操作可能な状態にしておく．(2) プローブからの結果を受け取る MQTT ブローカーの

二つがある。今回の論文では、コントロールサーバの実装として MQTT 接続を受け入れ、メッセージキューに対して送信されたデータをキューに入れるという処理を実装している。

4.3.4 メッセージの取り出しと MapReduce 処理の実行

メッセージキューからのデータは 1 分おきに取り出され、部分ファイルが CSV 形式に整形されローカルファイルシステムから HDFS 上に定期的にコピーが行われる。部分ファイルは一時ファイルとして作成され、コピーが完了した後は自動的にローカルファイルシステムから削除される。HDFS 上にコピーされた部分データは機能要件で定義を行った三つの目的のためのデータ処理が並列で行われる。HDFS への操作は WebHDFS[19] を用いて、HDFS が提供している REST 形式での API を呼び出すことによってファイルシステムの操作を行った。今回、部分データに対する MapReduce 処理を行うために Apache Pig の DSL である Pig Latin を用いて実際のデータ処理を記述した。Pig は UDF(ユーザー定義関数) を用いて独自の処理を DSL に追加する事ができる。今回は、一定以上の閾値を超えてある特定の well-known ポートへの通信をカウントするための処理と IP プレフィクスマッチングを行うための構文を JPython で記述可能な UDF[18] を用いて実装している。Pig を用いた処理が行われた結果は、HDFS 上に保存されておりこれをパースして DOTS クライアントプロセスへ受けわたす事によってシグナリングが行われる。

第5章　評価

5.1　評価方針

前提条件として現状展開されている RIPE Atlas と同等の規模でデプロイすされていることを事を想定した上で評価を行う。RIPE Atlas は 2018 年 1 月 17 日現在、10,330 台のプローブ、データセンター向けの Anchor と呼ばれるコンピュータが 304 台稼働している。今回は、本提案手法とその手法を構成する中央サーバアーキテクチャ部分を中心とした評価を行い、提案した中央サーバアーキテクチャが問題なく動作すうかを検証する。

5.2　評価手法

評価手法としては、以下の 3 点を本論文の評価手法とする。

1. 従来のメールやウェブサイト等にかわることのでき、かつマルウェアの活動を迅速に把握し、対策に役立てるためのマルウェアのサーバのダウンロード情報や、ポートごとの観測状況などを報告することができるか
2. プローブによって送信されたデータが HDFS までに到達する間までのパケットロス
3. 遅延を発生させずにデータを送信した場合と、各パケットごとに擬似的な遅延を発生させ一定の pps(packet per second) でデータを送信した場合の比較
4. メッセージキューから HDFS への移動を行う際にバッファリングされているデータの推移

今回検証を行うにあたって、パケットサイズは x バイト固定、送信されるパケットの pps(packet per second) は RIPE Atlas と同規模の 10330 台のプローブが存在することを前提とした上で 10,330 台の各プローブが 10 秒に一回 SYN スキャンを受ける仮定の元、0.1pps * 10,330 = 1,033pps を 1 秒あたりのパケットの送信数として固定する。なお、異常パケットとして送信する擬似的なパケットデータはループバックアドレス等の一部のアドレスブロックを除外した IP アドレスと、ポート番号は Well-known ポート (65,535 通り) の範囲からランダムに算出した値を用いる。

5.2.1 到達までのパケットロス

1033pps で MQTT による通信でコントロールサーバへ報告するソフトウェアを作成し、検証を行った。1,033pps でパケットを安定して生成し、計測データに対してなるべく影響がないようにするため以下の方法で生成間隔を決定した。送信パケットサイズは

(1) 以下の計測アルゴリズムを使って 1,033 パケットを MQTT ブローカーに送信するために要した時間を調べる。(2) その時間 t が 1 秒以下だった場合は 1-t 秒待ち、この値を計測する。これによって 1,033pps を維持し、これを 1500 ラウンド (1,033 * 1500 = 1,549,500 パケット) 繰り返して平均時間の推移を計測する。(3) 得た平均値 v を 1,033 で割りその値を 1 パケット送信した間隔時間として差し込む。

平均値測定プログラム

```
$seq_idx = 0
$i = 0.0
$prev_sum = 0.0
def generate_traffic_at_1033pps(handle)
  loop do
    y = Time.new
    1033.times {|n|
       handle.publish('probe.data', make_tcp_syn_packet)
    }
    x = Time.new
    t = x - y
    sleep (1 - t) if t > 0 && t < 1
    $prev_sum += (1-t)
    $i += 1
    puts "#{$i}: Genetared 2400pps Traffic, elapsed_time=#{t}sec, w=#{(1+t)}sec, av
  end
end
```

5.2.2 データ転送にかかる所要時間の計測

遅延を発生させない状態で 1033pps * 1000 回、計 x 回の MQTT パケット送信を行った時と、1033pps を維持するように擬似的な遅延を発生した状態で同数の MQTT パケットを送信した所要時間を比較した。

5.2.3　バッファリングデータの推移

MQTT を通じて送信された収集パケットのデータは、メッセージキューを通じて HDFS へと移動される。この時、一定時間 (現時点では 60 秒に 1 回の頻度）でデータがローカルファイルシステムから HDFS へとコピーされ MapReduce 処理が行われるが、コピーできなかったデータは次の 60 秒までメモリ上にバッファリングされる。メモリ上にバッファされたデータの推移を 60 分間の間で計測を行い、バッファの増減が著しく増加しているかを検証する。

5.3　評価結果

上記の一連の評価を行った結果をまとめる。

5.3.1　定性評価

本論文では、参加者同士がプローブを設置することによって情報共有を行い、従来のメールやウェブサイト等で告知が行われていたセキュリティアラートを、より自動化との親和性の高い手法を用いることによって実現するアーキテクチャができた。また、擬似データによるシミュレーションでの実装により実際にデータの集計を行いポート番号が一定の閾値に達した場合に DOTS によるアラートを動作させることができた。

5.3.2　到達までの収集パケットのロス

プローブからデータを MQTT を用いたデータを実際に HDFS 上にコピーするまでに発生するパケットロスは、すべての経路で TCP を用いた通信を行なっているため自動的に再送等の処理が行われており、アプリケーション層レベルでのバッファリングもあわせて行なっているため収集データのロスは確認されなかった。

上図は収集したデータを転送する時に要した時間を比較したものであり、青、赤いずれも同数のパケットを送信している。青の線グラフは、擬似的な遅延を実装せずに MQTT ブローカーに向けてパケットを送信した時にかかった所用時間となっている。一方の赤の線グラフは 1,033pps を達成するために 1 パケットあたりに擬似的な遅延を発生させた場合の所要時間となっている。赤のグラフでは、一時的なデータ転送時間の上昇が確認できたが影響は一時的なものであると考えられる。どちらのグラフにおいてもおおよそ一定の時間でパケットの転送が処理できていることが確認できた。

続いて、メッセージキューから受け取ったデータを HDFS に定期的にコピーするためのバッファの状態の変化の遷移のグラフを示す。計測を開始してから間もない 5 分の幅では処理パケット数が比較的少ない結果が出ているが、残りの時間では 1 分あたり 30 万パケットを比較的安定して処理が行えており、バッファの詰まりなどの問題は確認されなかった。

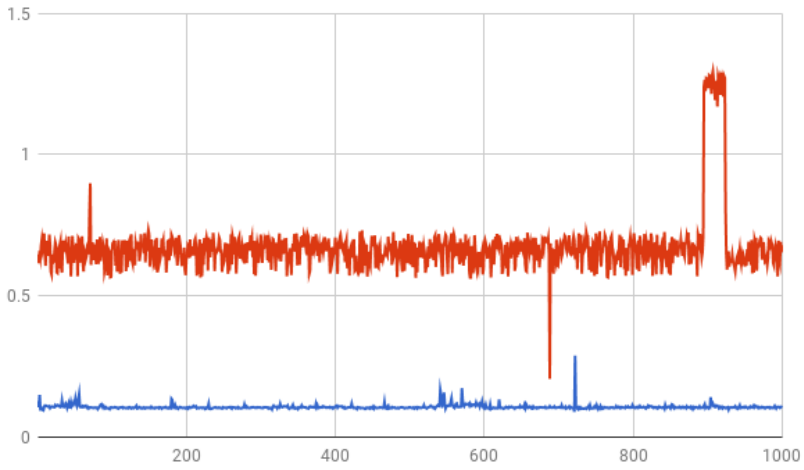

図 5.1: データ転送にかかる所要時間の比較

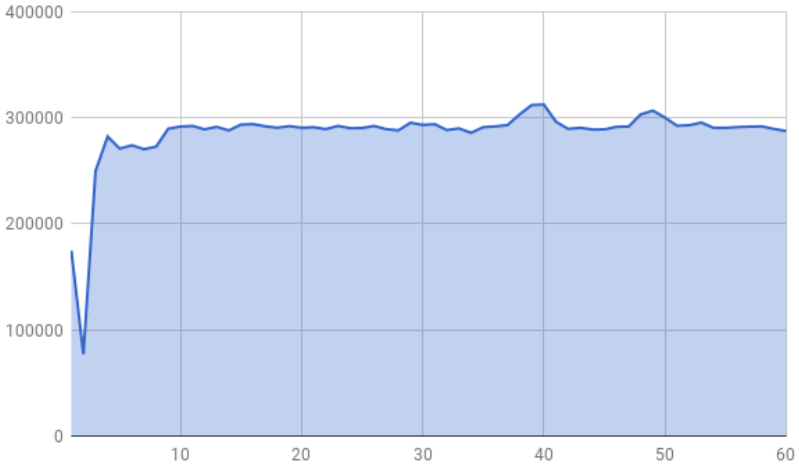

図 5.2: 収集パケット処理中のバッファの変化状態

第6章　結論

本章では、本論文の提案手法と評価の結果を交えた上で提案手法の有意性について議論する。

6.1　本研究のまとめ

本研究では、脆弱な IoT 機器を標的にした自己増殖型のボットネットに対して地理的な感染分布の偏りに着目を行い、IoT 機器を標的とした今後の初期感染活動はネットワーク環境とすでに設置された IoT 機器のセキュリティ対策の難しさを考慮しこれらの偏った分布の国や地域に対して行われる可能性が高いという仮説の元、以下の二つの点を研究の目的とした。

- DDoS 攻撃の監視と対策の連携においてメールやウェブサイトの注意喚起に代わる自動化との親和性の高い手法
- IoT 機器を標的とした DDoS ボットネットの感染活動を初期段階で把握し、上記の仕組みを用いて迅速に共有

具体的な仕組みとして、インターネット定点観測プローブと IETF で標準化が勧められている DOTS(DDoS Open Threat Signaling) プロトコルを用いた参加者共有型のコミュニティベースの IoT DDoS ボットネットアラートシステムを提案した。IoT ボットネットの感染の仕組みの共通点を洗い出し、定点観測プローブ内の IoT 機器をターゲットとしたハニーポットによる感染の際に必要とされるサーバ情報の抽出、プローブによるスキャン活動の動きを DOTS を用いて迅速に共有し、マルウェアが感染に利用しているサーバへの外向き通信をブロックする等の IoT DDoS ボットネットの素早い対策に生かす事が可能になる。

評価の結果として十分に実地検証においてもこの仕組みが動作可能かをシミュレーションを用いて確認を行った。

6.2　今後の課題と展開

本提案手法では、外部のプローブから相互に監視しあうことによってスキャン活動を行なっている攻撃ノードが属しているネットワーク管理者に対して DOTS を用いた迅速な DDoS シグナリングが可能になっているが、IP-spoofing によるスキャン活動に対する対策を行っ

ておらず、IP traceback[3] 等を用いたアーキテクチャに対する機能の追加が必要となっている。

また、設置されたプローブからの虚偽の情報の対応についても現状プローブ上に搭載されている OS とファイルシステム自体の暗号化と、固有の暗号鍵を各プローブに割り当てる事によって不正なプローブからの情報を即時停止できる仕組みを実装しているが、引き続きセキュリティ強度を高める手法を模索する必要がある。

Mirai や、多くの IoT DDoS ボットネットの出現とその攻撃性能の高さから IoT 機器のセキュリティに対するガイドラインの策定などが行われはじめ、IoT 機器のセキュリティに対する関心が高まっている。しかしすでに設置されている脆弱な IoT 機器への対処は容易ではない。今後の展開としては最終的に現地にプローブを実際に設置した上で、IoT 機器を取り巻くセキュリティの進歩への貢献を行う事が可能になる事を望んでいる。

謝辞

本論文の執筆にあたり、ご指導いただきました慶應義塾大学 環境情報学部教授 村井純 博士、同学部教授 中村修 博士、同学部准教授 Rodney D. Van Meter 博士、同学部准 教授 三次仁博士、同学部准教授 楠本博之博士、同学部准教授 植原啓介博士、同学部准教 授 中澤仁博士、政策・メディア研究科特任准教授 鈴木茂哉博士、SFC 研究所 上席所員 (訪問) 斉藤賢爾博士に感謝致します。

研究について日頃からご指導頂きました政策・メディア研究科博士課程 松谷健史氏、政策・メディア研究科特任助教 空閑洋平氏に感謝致します。研究室に所属したばかりの頃から本研究に至るまで、ご多忙の中、研究者として多くの面で未熟であった自分を見捨てることなく、技術面やストーリー面で絶えず多くのご指導をいただきました。両氏のご指導なくして本研究を卒業論文としてまとめることはできませんでした。重ねて感謝申し上げます。

参考文献

[1] Bashlite malware uses shellshock to hijack devices running busybox. http://www.securityweek.com/bashlite-malware-uses-shellshock-hijack-devices-running-busybox.

[2] Bittorrent.org bep5: Dht protocol. http://www.bittorrent.org/beps/bep_0005.html.

[3] A brief survey of ip traceback methodologies. https://www.uni-obuda.hu/journal/Murugesan_Shalinie_Neethimani_55.pdf.

[4] Censys. https://censys.io/.

[5] Cve-2014-6271 - shellshock. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271.

[6] Elfbashlite.a - trend micro. https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/ELF_BASHLITE.SMB.

[7] Geniosa/qbot. https://github.com/geniosa/qbot.

[8] Github: jgamblim/mirai-source-code. https://github.com/jgamblin/mirai-source-code/.

[9] Github: nttdots/go-dots. https://github.com/nttdots/go-dots.

[10] Hajime, the mysterious evolving botnet. https://securelist.com/hajime-the-mysterious-evolving-botnet/78160/.

[11] Internet census 2012. http://internetcensus2012.bitbucket.org/paper.html.

[12] An introduction to hardware hacking ripe atlas probe. https://www.mdsec.co.uk/2015/09/an-introduction-to-hardware-hacking-the-ripe-atlas-probe/.

[13] An introduction to hardware hacking ripe atlas probe. https://www.mdsec.co.uk/2015/09/an-introduction-to-hardware-hacking-the-ripe-atlas-probe/.

[14] Nicter 観測レポート 2016. http://www.nict.go.jp/cyber/report/NICTER_report_2016.pdf.

[15] Now mirai has dga future built in. http://blog.netlab.360.com/new-mirai-variant-with-dga/.

[16] Ripe atlas. https://atlas.ripe.net/.

[17] Ripe atlas: A global internet measurement by ripe ncc staff the internet protocol journal september 2015. http://ipj.dreamhosters.com/wp-content/uploads/2015/10/ipj18.3.pdf.

[18] User defined function - apache pig. https://pig.apache.org/docs/r0.9.1/udf.html.

[19] Webhdfs rest api - apache hadoop. https://hadoop.apache.org/docs/r1.0.4/webhdfs.html.

[20] 雄宣布召回在美国售的部分品. http://mp.weixin.qq.com/s?__biz=MzA4MDQ4NjMwOA==&mid=2651450911&idx=1&sn=f4d41b6fae77ece8493fdec1197d97f0.

[21] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. Understanding the mirai botnet. In *26th USENIX Security Symposium (USENIX Security 17)*, pp. 1093–1110, Vancouver, BC, 2017. USENIX Association.

[22] Vaibhav Bajpai, Steffie Jacob Eravuchira, and Jürgen Schönwälder. Lessons learned from using the ripe atlas platform for measurement research. *SIGCOMM Comput. Commun. Rev.*, Vol. 45, No. 3, pp. 35–42, July 2015.

[23] Daisuke Inoue, Masashi Eto, Koei Suzuki, Mio Suzuki, and Koji Nakao. Daedalus-viz: Novel real-time 3d visualization for darknet monitoring-based alert system. In *Proceedings of the Ninth International Symposium on Visualization for Cyber Security*, VizSec '12, pp. 72–79, New York, NY, USA, 2012. ACM.

[24] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pp. 53–65, London, UK, UK, 2002. Springer-Verlag.

[25] Andrew Mortensen, Flemming Andreasen, Tirumaleswar Reddy, Christopher Gray, Rich Compton, and Nik Teague. Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture. Internet-Draft draft-ietf-dots-architecture-05, Internet Engineering Task Force, October 2017. Work in Progress.

[26] Andrew Mortensen, Robert Moskowitz, and Tirumaleswar Reddy. Distributed Denial of Service (DDoS) Open Threat Signaling Requirements. Internet-Draft draft-ietf-dots-requirements-09, Internet Engineering Task Force, December 2017. Work in Progress.

[27] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. Iotpot: Analysing the rise of iot compromises. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*, Washington, D.C., 2015. USENIX Association.

[28] Tirumaleswar Reddy, Mohamed Boucadair, Kaname Nishizuka, Liang Xia, Prashanth Patil, Andrew Mortensen, and Nik Teague. Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel. Internet-Draft draft-ietf-dots-data-channel-11, Internet Engineering Task Force, December 2017. Work in Progress.

[29] Tirumaleswar Reddy, Mohamed Boucadair, Prashanth Patil, Andrew Mortensen, and Nik Teague. Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel. Internet-Draft draft-ietf-dots-signal-channel-14, Internet Engineering Task Force, December 2017. Work in Progress.

[30] Zach Shelby, Klaus Hartke, and Carsten Bormann. The constrained application protocol (CoAP). Technical Report 7252, RFC Editor, Fremont, CA, USA, June 2014.

[31] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '01, pp. 149–160, New York, NY, USA, 2001. ACM.

[32] Michel Veillette, Alexander Pelov, Abhinav Somaraju, Randy Turner, and Ana Minaburo. CBOR Encoding of Data Modeled with YANG. Internet-Draft draft-ietf-core-yang-cbor-05, Internet Engineering Task Force, August 2017. Work in Progress.