

Mirai-Botnet による分散型サービス不能攻撃に対する評価環境の開発

井上 博貴¹⁾, 三河 賢治²⁾, 田中 賢³⁾, 瀧野 敬²⁾

1) 前橋工科大学大学院 工学研究科

2) 前橋工科大学 工学部

3) 神奈川大学 情報学部

mikawa@maebashi-it.ac.jp

Development of Evaluation Environment against Distributed Denial of Service Attacks by Mirai-Botnet Families

Hiroki Inoue¹⁾, Kenji Mikawa²⁾, Ken Tanaka³⁾, Takashi Fuchino²⁾

1) Graduate School of Engineering, Maebashi Institute of Technology

2) Faculty of Engineering, Maebashi Institute of Technology

3) Faculty of Informatics, Kanagawa University

概要

近年インターネットに接続する IoT 機器の活用が進められる一方で、IoT 機器を標的とするマルウェアの活動も活発になっている。2016 年に出現した Mirai は、不十分な管理の IoT 機器を悪用して、大規模な分散型サービス不能攻撃 (DDoS Attack) を行った。その後、Mirai のソースコードが公開され、Mirai の亜種が多数出現した。Bot による感染活動や DDoS 攻撃の対策は急務である。本研究は、これらの対策を迅速に進めることができるよう、実際の Bot による感染活動や DDoS 攻撃を模倣できる試験環境を開発し、試験環境の構築に必要な機器の性能及び実現可能な被害の状況を明らかにする。実証実験の結果、小規模なネットワーク上に構築された標準的な機器を使用した試験環境において、十分にネットワークを過負荷状態に陥れることが可能であることが示された。

1 はじめに

スマートフォンと連携したネットワークカメラや家電製品、防犯システム等、便利さが故に近年インターネットに接続する IoT 機器が活用されている。その一方で、IoT 機器を標的としたサイバー攻撃が増加し、マルウェアの感染活動が活発になっている。このような IoT 機器は簡単に導入できる反面、それらを管理するユーザの知識不足もあって、初期状態のままで利用されていたり、不完全な設定のままで利用されていることも多い。

マルウェアは、例えばユーザの不注意や不十分な管理により機器にインストールされたり、機器の脆弱性を悪用して攻撃者によりインストールされる。2016 年に出現したマルウェアの Mirai は、ネットワークに接続された管理の不十分な IoT 機器に感染して最大 38 万個の機器からなる巨大なボットネットを構築し、2016 年 9 月の同時期にアメリカのセキュリティサイト「Krebs on Security」に対して最大 665Gbps、フランスのインターネットサービスプロバイダ「OVH」対

して最大 1Tbps という大規模な DDoS 攻撃を行った。2016 年 10 月に突如 Mirai のソースコードが GitHub 上に公開 [5] され、以降、Mirai を改造した亜種が多数出現することとなった。その一方で、Mirai のソースコードを詳しく分析した研究報告 [3, 1], Mirai 及び亜種によるボットネットの監視システムの構築 [2], DDoS 攻撃の対策を実習できる演習システムの開発 [4] 等、Mirai のソースコードの公開に伴い、DoS/DDoS 攻撃に関連する研究が活発に行われている。

DoS 攻撃として、SYN Flood 攻撃のように細工したパケットを大量に送信する攻撃、F5 アタックのように正常なパケットを大量に送信する攻撃等が知られる。前者の細工したパケットによる攻撃の場合、SYN Cookie 等のセキュリティ対策 [6] により Web サーバ側の対策は進んでいるが、前者の攻撃も含めて後者の大量のパケットによる攻撃の場合、ネットワーク帯域を圧迫し通信障害を引き起こすため、このような DoS 攻撃に対する対策は難しいとされる。DoS/DDoS 攻撃は現在もインターネット上で猛威を振るっており、それらの対策は急務である。

DoS/DDoS 攻撃に対する実用的なセキュリティ対策を研究する上で、実際の攻撃を模倣できる試験環境でその対策の有効性を確認したい。しかしながら、試験環境としてどのような機器やソフトウェアを選定するか、試験環境内の DoS/DDoS 攻撃でどのような被害を実現できるか、解決しなければいけない多数の課題が存在する。そこで本研究は、Mirai のソースコードの分析の結果、安全に運用可能な DoS/DDoS 攻撃を模倣する試験環境を構築し、試験環境の構築に必要な機器の性能及び実現可能な被害の状況を明らかにする。

2 Mirai-Botnet の構成

Mirai-Botnet のシステム構成及び各機器の連携の概要を図 1 に示す。Mirai-Botnet は、CNC サーバ、Scan Receiver, Loader, マルウェア配布サーバ (HTTP/TFTP サーバ) のサーバ群、IoT 機器 (Bot) で構成される。Mirai-Botnet のシステム構成や各機器の機能等は多くの文献で詳しく述べられており [1, 3], 本節では評価環境の構築に必要な特徴的な性質を中心に述べる。

Mirai-Botnet では、CNC サーバや Scan Receiver のドメイン名やポート番号、IoT 機器への Telnet スキャンで使用する認証情報等を難読化し、攻撃者の特定を困難にしている。Mirai による感染活動は、次のとおりである。Mirai に感染した IoT 機器 (Bot) は、ランダムに生成された IP アドレスに対して Telnet スキャンを試行し、ログインに成功した IoT 機器に Mirai のバイナリファイルを感染させることで、次々に新たな Bot を作りだす。したがって、最初の Bot となる IoT 機器については、Mirai のバイナリファイルを手動でインストールし、ネットワークに放出する必要がある。

2.1 CNC サーバ

CNC サーバは、Mirai-Botnet を管理するためのサーバである。Mirai-Botnet に関するデータをデータベース管理システムの MySQL で管理しており、Mirai-Botnet を使用するユーザ情報を管理する “users”, DoS 攻撃の履歴を管理する “history”, DoS 攻撃の対象としないホストを管理する “whitelist” の 3 つのテーブルをもつ。

Mirai-Botnet を使用するユーザは、CNC サーバに Telnet (23/TCP) または API (101/TCP) 経由でアクセスするが、Telnet でアクセスする場合、MySQL の “users” にあらかじめ登録されているユーザ ID と

表 1: CNC サーバで使用可能なコマンド

コマンド	説明
adduser	ユーザを追加する
botcount	現在登録されているボット数を表示する
-	使用するボット数を指定する
?	攻撃コマンドの説明を表示する
exit	CNC サーバとの通信を切断する

パスワードの組を使用してログインする。ユーザ ID とパスワードの入力には制限時間が設定されており、両方ともに 60 秒以内に入力を完了しない場合にはタイムアウトとなる。CNC サーバへのログインが成功すると、図 2 のような処理結果を想起させる文字列が出力されるが、何かしらの処理が実装されているわけではなくダミー表示である。API でアクセスする場合もあらかじめ MySQL に登録されている API キーを使用してログインする。API では、ログイン画面も Telnet とは異なり、入力可能なコマンドも攻撃コマンドのみといった簡易的な構成となっている。

Telnet 経由で使用可能なコマンドは表 1 のとおりである。攻撃を実行するためのコマンドは、“使用する Bot 数”, “攻撃名”, “攻撃先 IP アドレス”, “攻撃持続時間”, “攻撃フラグ” を空白で区切り、順に入力する。一部の項目は入力しなくてもよいオプションである。攻撃コマンドを入力後、“攻撃名”, “攻撃先 IP アドレス”, “攻撃持続時間”, “攻撃を実行した時間”, “ユーザ ID” の各情報が MySQL の “history” に保存される。また、攻撃に成功した場合、あらかじめ設定された時間 (クールタイム) が経過するまで次の攻撃を実行できない仕様となっている。CNC サーバは、Bot への攻撃命令の他、Bot の生存確認、Bot の登録等の通信を行う。

2.2 Bot

Mirai に感染した IoT 機器 (Bot) は、次の標的を求めて、ランダムに生成された IP アドレスの 23/TCP と 2323/TCP に対して 9 対 1 の比で Telnet ログインを試行する。このとき、Bot にハードコーディングされたユーザ ID とパスワードを用いて辞書攻撃を行う。BusyBox は、Linux のコマンド群をパッケージ化したもので、実装のサイズが小さいため IoT 機器にインストールされることが多い。Mirai は、BusyBox がインストールされた IoT 機器に感染する。実際、標準の Telnet サーバを起動した IoT 機器では Bot によるログイン試行は成功せず、BusyBox の Telnet サーバで

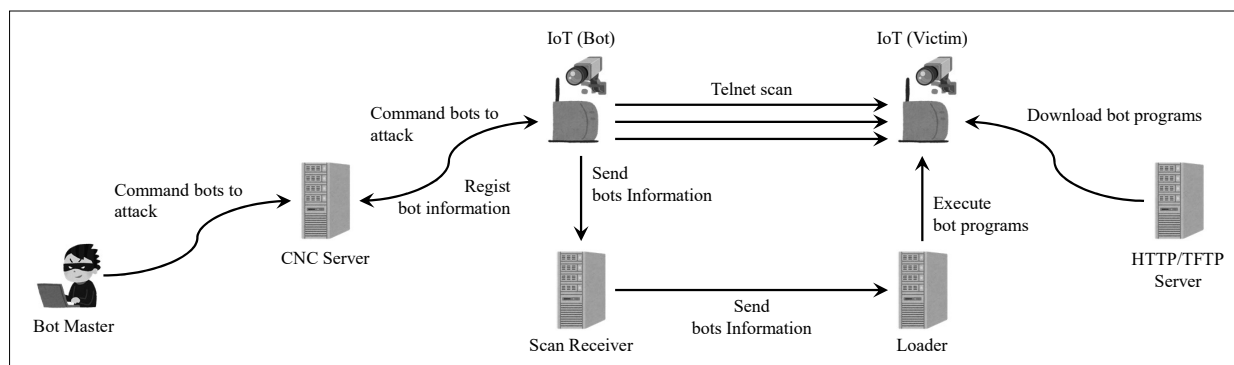


図 1: Mirai-Botnet の概観

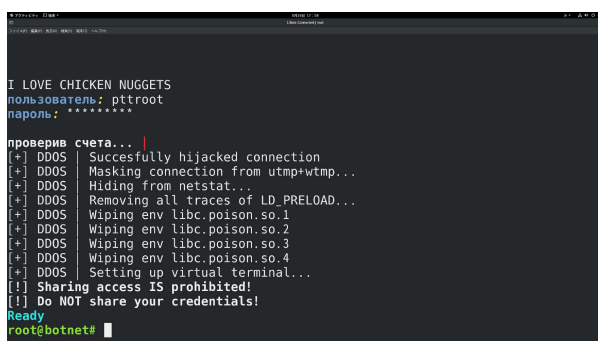


図 2: CNC サーバのログイン画面

のみ成功する。また、Bot 内で実行される多くのコマンドが BusyBox のコマンドを使用している。

Mirai に感染した IoT 機器は、CNC サーバに次のバイナリコード “\0x00\0x00\0x00\0x01” を送信し、CNC サーバに正式に Bot として登録される。Bot では、CNC サーバからの攻撃命令を待機するプロセスと他の IoT 機器に対する Telnet ログインを試行するプロセスが稼働している。Bot は、CNC サーバから攻撃命令を受信すると、攻撃命令の字句解析を行い、指定された攻撃先に対して指定された攻撃を実行する。全 11 種類の攻撃が用意されているが、実際には 10 種類の攻撃のみ実装されていて、未実装の攻撃はソースコード上で白紙となっている。これらのプロセスは起動時にランダムなプロセス名に変換されており、IoT 機器の管理者に感染を悟られないための偽装工作と思われる。なお、Bot のソースコードではデバッグモードが用意されているが、デバッグモードでは感染活動を行うコードを実行できない仕様になっている。

2.3 Loader と Scan Receiver

Scan Receiver は、Bot からの辞書攻撃により Telnet ログインに成功した IoT 機器の認証情報や IP アドレス等の機器情報を Bot から受信し、それらの情報を標準出力に出力するだけのサーバである。これらの機器

情報を Loader の標準入力で読み取り、Loader は IoT 機器に対して実際の感染活動を開始する。しかしながら、公開されているソースコード [5] には、これらの出力結果を Scan Receiver から Loader に自動で転送する仕組みは実装されておらず、本研究では評価環境の構築にあたり、自動で転送する仕組みを実装した。

Loader は、Scan Receiver から得られた認証情報等を用いて、IoT 機器に対して Telnet ログインを試みる。このとき、IoT 機器からの Telnet 応答の通信に対して、 $1/2^{16}$ の確率でランダムに生成された数値に一致しない宛先ポート番号の通信を遮断する。すなわち、IoT 機器は極稀な確率でのみ Mirai に感染することになり、どのような意図で感染活動を疎外する本コードを作成したのか理解に苦しむ。ログインに成功後、Loader は IoT 機器のアーキテクチャを調べ、アーキテクチャに対応した Mirai のバイナリファイルをダウンロードする。

これら一連の感染活動において、Loader が実行する特徴的な動作を述べる。IoT 機器へのログインに成功後、IoT 機器のシェルを POSIX 標準のシェルに切り替えている。これは、IoT 機器の管理者に history コマンド等で感染活動を悟られないための偽装工作と推測される。また、IoT 機器で使用している全マウントを取得し、“`.nippon`” というファイルに保存しているが、IoT 機器によって（次節の実証実験では、IoT 機器に搭載する OS によって）タイムアウトとなり、次の処理に進まない状況が確認された。マウントを取得する作業は感染活動に直接関係のないコードのようであり、評価環境の構築では、この部分のコードをコメントアウトしている。ダミーコマンドである “IHCCCE” の実行を記述している箇所があり、このようなコードもコメントアウトしている。

3 実証実験

本節では、はじめに Mirai-Botnet 評価環境の構築について、次に、Mirai-Botnet からの各種攻撃手法に対する被攻撃サーバ（Web サーバ）の挙動について述べる。最新の OS では標準設定でセキュリティ対策が施されており、DoS 攻撃に対する耐性が高い。被攻撃サーバの挙動に関して、ネットワークの応答反応で判断する。Web サーバは Bot からの攻撃を受けるので、あえて古い OS をインストールし多様な DoS 攻撃を成功させる、という方針も考えられるが、本実証実験では最新の OS をインストールし、このような環境下においても DoS 攻撃が十分に機能する状況を確認する。

3.1 Mirai-Botnet 評価環境の構築

第 2 節で述べたとおり、Mirai-Botnet は攻撃者側の機器として CNC サーバ、Scan Receiver、Loader、実行ファイル配信サーバ、DNS サーバ、被攻撃者側の機器のうち、遠隔操作される Bot として BusyBox を搭載する IoT 機器、被攻撃対象機器として Web サーバを用意する。Mirai-Botnet では、CNC サーバ、Scan Receiver、Loader 等に FQDN を与えており、これらの名前解決に DNS サーバを用意する。攻撃者側の 5 台の機器は、1 台の仮想サーバ上に構成されたゲスト OS として稼働する。仮想サーバの具体的な性能は、CPU として Intel Xeon E3-1240v5 (3.5GHz)、メモリとして 16GB、ネットワーク I/F として 1000Base-T (有線) である。

被攻撃者側の機器のうち、被攻撃対象機器の Web サーバは単体の物理サーバとして構成し、その性能は、CPU として Intel Core i7-6500 (2.5GHz)、メモリとして 8GB、ネットワーク I/F として 1000Base-T (有線) である。Web サーバの OS は、最新の Linux OS である Rocky Linux 9.4 (5.14.0-427.31.1) である。被攻撃者側の機器のうち、遠隔操作される IoT 機器 (Bot) は、性能の異なる 2 種類の Raspberry Pi をそれぞれ 2 台用意し、各機器に異なる OS をインストールしている。機種及び OS の組み合わせによる攻撃能力に相違が見られるか、実証実験で確認する。IoT 機器の具体的な性能を表 2 に示す。Bot からの DoS 攻撃による通信の挙動を計測するため、L2 スイッチにミラーポートを設定し、キャプチャツールとして Wireshark をインストールした計測用 PC を接続する。Mirai は、辞書攻撃の後に Telnet 通信を利用して感染活動を行うため、事前準備として、IoT 機器側の

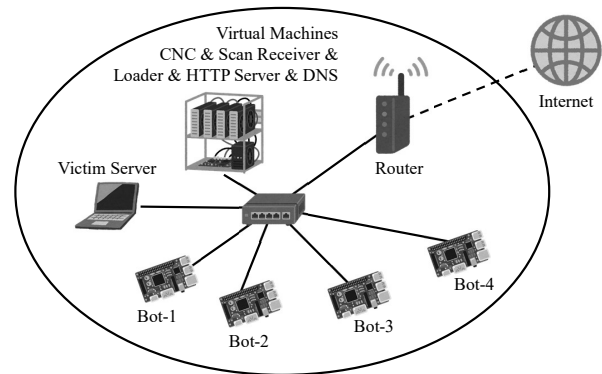


図 3: Mirai-Botnet 実証実験ネットワーク

管理者アカウント名とパスワード文字列を Mirai に搭載される辞書と同一の文字列に変更し、busybox による Telnet サーバ (23 番ポートまたは 2323 番ポート) を稼働させておく。また、攻撃者側のサーバ機器との通信を妨害されないように IoT 機器側に搭載されるセキュリティ機能、SELinux 及び Firewalld 等を停止しておく。

次に、これらの機器のネットワーク構成を述べる。Mirai の感染活動によって被害が外部のネットワークに及ばないようにブロードバンドルータを設置し、閉じたネットワーク環境を構成している (図 3 参照)。実際、実証実験を実施するときはブロードバンドルータの WAN 側のネットワークケーブルを外し、外部と完全に遮断された環境を実現している。仮想サーバ、Web サーバ、IoT 機器は、ブロードバンドルータ配下のスイッチングハブを中心とするスター型に接続し、仮想サーバ上の各ゲスト OS はブリッジ接続として論理的にブロードバンドルータ配下の機器となるように接続する。

3.2 DoS 攻撃に対する耐性評価

最新の OS は標準設定でセキュリティ対策が施されており、DoS 攻撃に対する耐性が高い。被攻撃対象機器の Web サーバに搭載される Rocky Linux 9.4 も例外なく SYN Cookie 等のセキュリティ対策 [6] が施されており、4 台の IoT 機器から同時に SYN Flood 攻撃を実施しても Web サーバのメモリを枯渇させることも CPU を上限まで使用させることも不可能であった。本実証実験では、ネットワーク帯域の枯渇に注目して DoS 攻撃に対する耐性を評価する。Bot 化された各 IoT 機器から Web サーバに対して、Mirai に搭載される DoS 攻撃から次の 6 種類、UDP Flood (DoS-1)、Valve Source Engine Specific Flood (DoS-2)、SYN Flood (DoS-3)、ACK Flood (DoS-4)、TCP STOMP

表 2: IoT 機器 (Bot) の性能

ID	Bot-1	Bot-2	Bot-3	Bot-4
機種	Raspberry Pi Zero W	Raspberry Pi Zero W	Raspberry Pi 3 B+	Raspberry Pi 3 B+
CPU	ARMv6	ARMv6	ARMv8	ARMv8
メモリ	512MB	512MB	1,024MB	1,024MB
NIC	1000Mbps (有線)	1000Mbps (有線)	1000Mbps (有線)	1000Mbps (有線)
OS	Raspbian GNU/Linux12	Raspbian GNU/Linux10	Ubuntu Server23.10	Raspbian GNU/Linux11

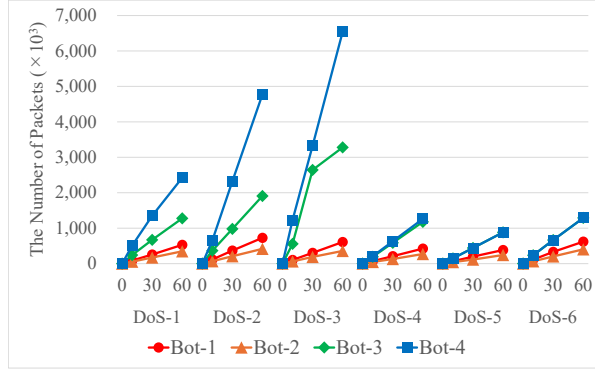


図 4: Bot からの累積受信パケット数

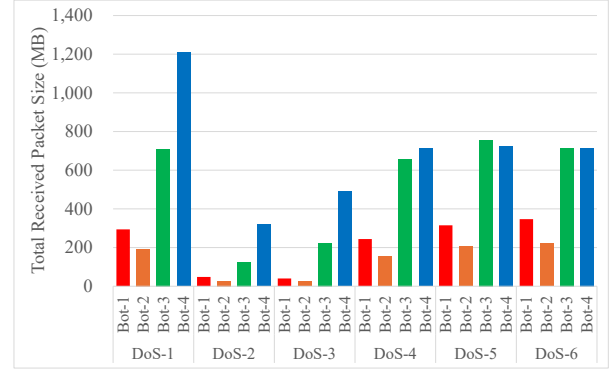


図 5: Bot からの累積受信パケットサイズ

Flood (DoS-5), UDP Plain Flood (DoS-6) を実行する。各攻撃の詳細を表 3 に示す。DoS-1, DoS-2 及び DoS-6 は UDP パケットを大量送信する DoS 攻撃であり, DoS-3, DoS-4 及び DoS-5 は TCP パケットを大量送信する DoS 攻撃である。

3.2.1 IoT 機器の送信能力

各 IoT 機器から Web サーバに対して攻撃通信を送信したときの計測用 PC で計測した累積の受信パケット数を図 4 に示す。攻撃別に各 IoT 機器から送信されたパケットを計測用 PC で 10 秒, 30 秒, 60 秒の時点で計測した累積の受信パケット数を表示している。図中, 縦軸 (受信パケット数の単位) は 10^3 個, 横軸は秒である。累積受信パケット数を攻撃別に比較すると, いずれの攻撃においても時間に比例してパケットを送信しており, 単位時間あたりの送信パケット数は一定を保っていると考えられる。一方, 機種別に比較すると, 同一の攻撃について, Raspberry Pi Zero W と Raspberry Pi 3 B+ では最大で 18 倍の送信能力の相違がみられるが, 同一機種であっても OS により 1.5 倍から 2 倍程度の送信能力に相違が見られる。また, Raspberry Pi 3 Model B+ において, DoS-4 から DoS-6 までの DoS 攻撃時の累積受信パケット数が著しく減少する状況が確認された。DoS-4 と DoS-5 は TCP コネクション型の攻撃なので同型の DoS-3 の累積受信パケット数, DoS-6 は UDP コネクションレス型の攻撃なので同型の DoS-1 の累積受信パケット数

に類似の傾向となるはずである。これは, 大量の攻撃的通信を短時間に送信したことによりサーマルスロットリング機能が作動したものと考えられる。

次に, 各 IoT 機器から Web サーバに対して攻撃通信を送信したときの Web サーバ側で 60 秒間計測した累積のパケットサイズを図 5 に示す。図中, 縦軸 (受信パケットサイズの単位) は MB である。前述のとおり, DoS-4 から DoS-6 までの攻撃では熱暴走に対する制御が作動していると考えられるため正確性に欠くが, DoS-3 のグラフから分かるように, TCP コネクション型の攻撃では大量の受信パケット数 (Bot からの大量の送信パケット数) に対して 1 つあたりのパケットサイズは 64 バイトと小さいため, 累積受信パケットサイズは大きくはない。UDP Flood 攻撃 (DoS-1) のように UDP コネクションレス型の攻撃がネットワーク帯域の占有に有効な手段となり得ることを示している。IoT 機器の攻撃通信の送信能力に関して, 機器のハードウェアによる相違だけでなく搭載する OS にも大きく依存している。

3.2.2 ネットワークの過負荷状況の可視化

次に, ネットワークの過負荷状況を評価するため, 各 IoT 機器から Web サーバに対して攻撃通信を送信したときの Web サーバの Ping 応答時間を図 6 から図 9 に示す。IoT 機器から Web サーバへの攻撃別に, 正常な機器から Web サーバに対して 1 秒間隔で Ping を 600 秒間送信したときの応答時間をグラフ化したも

表 3: 実証実験ネットワークで利用可能な DoS 攻撃

ID	攻撃	説明
DoS-1	UDP Flood	UDP パケットを大量に送信する。
DoS-2	VSE Specific Flood	ゲームエンジンに対して UDP Flood 攻撃を実行する。
DoS-3	SYN Flood	接続要求 (SYN) パケットを大量に送信する。
DoS-4	ACK Flood	確認応答 (ACK) パケットを大量に送信する。
DoS-5	TCP STOMP Flood	TCP セッション確立後に ACK Flood 攻撃を実行する。
DoS-6	UDP Plain Flood	情報を削減してサイズを小さくした UDP パケットを大量に送信する。

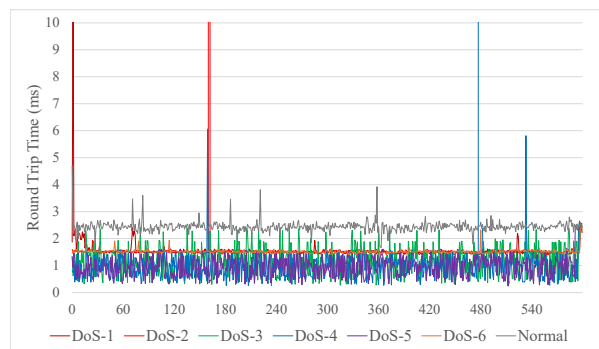


図 6: Bot-1 攻撃時の Ping 応答時間

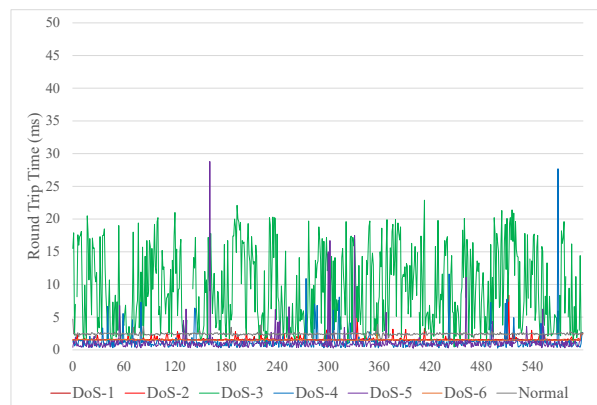


図 8: Bot-3 攻撃時の Ping 応答時間

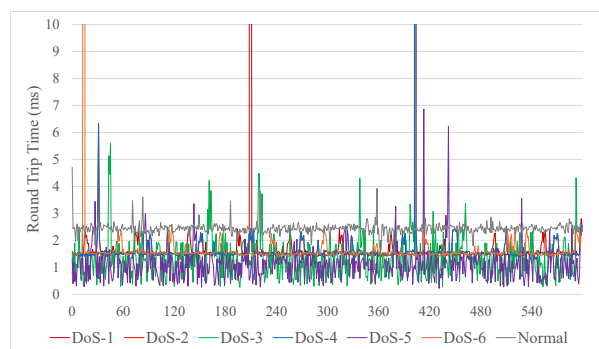


図 7: Bot-2 攻撃時の Ping 応答時間

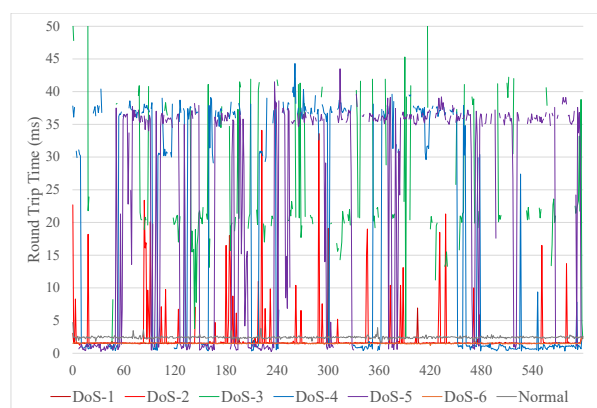


図 9: Bot-4 攻撃時の Ping 応答時間

のである。図中，“Normal”は、平常時（IoT 機器から DoS 攻撃を実施していないとき）の応答時間であり、被攻撃時の応答時間との比較のため掲載した。また、これらの応答時間の統計解析を表 4 に示す。実証実験ネットワークは、図 1 に示すとおり、ブロードバンドルータ配下の閉じたネットワークで構成される。Ping 応答時間はネットワークを構成する機器や環境に大きく依存するが、一般に、インターネットでは 50 ミリ秒以下、ローカルネットワーク内では 5 ミリ秒以下が標準的な応答時間であると言われている。実際、図 6 から図 9 において、平常時の Ping 応答時間（Normal）の平均は 2.44 ミリ秒（標準偏差 0.21）であった。

次に、DoS 攻撃別に Ping の応答時間を検証する。Bot-1 (Raspberry Pi Zero W) では、散発的に 5 ミリ秒を超える応答時間を観測しているが、おおむね 5 ミリ

秒以下で安定している（図 6 参照）。Bot-2 (Raspberry Pi Zero W) は、Bot-1 と同一機種であるが搭載する OS の世代が古い。Bot-2 でも Bot-1 と同様に散発的に 5 ミリ秒を超える応答時間を観測しているが、おおむね 5 ミリ秒以下で安定している（図 7 参照）。したがって、比較的性能の低い機種による単独の DoS 攻撃は成功していないと言える。

Bot-3 (Raspberry Pi 3 Model B+) では、UDP パケットによる DoS 攻撃（DoS-1, DoS-2 及び DoS-6）は、Bot-1, Bot-2 と同様におおむね 5 ミリ秒以下で安定しており、DoS 攻撃は成功していないと考えられるが、TCP パケットによる DoS 攻撃（DoS-3, DoS-4

表 4: Bot 攻撃時の Ping 応答時間の統計解析

(a) Bot-1 攻撃時						
	DoS-1	DoS-2	DoS-3	DoS-4	DoS-5	DoS-6
平均 (ミリ秒)	1.55	2.88	1.17	1.74	0.96	1.52
標準偏差	0.43	33.15	0.52	18.78	0.39	0.07

(b) Bot-2 攻撃時						
	DoS-1	DoS-2	DoS-3	DoS-4	DoS-5	DoS-6
平均 (ミリ秒)	1.96	1.85	1.32	1.85	1.12	1.78
標準偏差	7.95	6.71	0.71	6.71	0.60	4.33

(c) Bot-3 攻撃時						
	DoS-1	DoS-2	DoS-3	DoS-4	DoS-5	DoS-6
平均 (ミリ秒)	1.53	1.65	9.23	1.26	1.19	1.53
標準偏差	0.09	0.41	5.94	1.59	1.77	0.09

(d) Bot-4 攻撃時						
	DoS-1	DoS-2	DoS-3	DoS-4	DoS-5	DoS-6
平均 (ミリ秒)	1.56	2.39	40.11	19.11	23.95	1.53
標準偏差	0.24	3.53	114.88	17.64	16.19	0.08

(e) DDoS 攻撃時						
	DoS-1	DoS-2	DoS-3	DoS-4	DoS-5	DoS-6
平均 (ミリ秒)	1.55	1.57	90.38	40.89	63.37	1.56
標準偏差	0.05	0.08	279.09	35.81	210.41	0.21



図 10: 全 Bot からの DDoS 攻撃時の Ping 応答時間

及び DoS-5) では応答時間に相当のばらつきが観測され、散発的にパケットロス（計測不能）が発生しており、DoS 攻撃は成功していると言える（図 8 参照）。Bot-4 (Raspberry Pi 3 Model B+) は、Bot-3 と同一機種であるが機種専用の OS を搭載しており、機種の性能を最大限に引き出すことが可能である。Bot-4 では、UDP パケットによる DoS 攻撃においても散発的にパケットロスが発生しており、攻撃は成功していると考えられる。また、TCP パケットによる DoS 攻撃

では、頻繁にパケットロスが発生（図 9 中、パケットが計測不能となったとき、すなわちパケットロスが発生したときに不連続となる）しており、十分に攻撃が成功していると言える。したがって、Raspberry Pi 3 Model B+ 程度の比較的性能の高い機種であれば、単独の DoS 攻撃であっても十分にネットワークを過負荷状態に陥れることが可能である。

最後に 4 台すべての Bot から同時に DDoS 攻撃を行い、Ping の応答時間を検証する（図 10 参照）。UDP パケットによる DDoS 攻撃ではパケットロスが発生しておらず、前述の Bot-4 単体による UDP パケット攻撃で散発的にパケットロスが発生していたことを考慮すれば、DDoS 攻撃時に Bot-4 のサーマルスロットリング機能が作動したと考えるのが妥当であろう。しかしながら、このような状況であっても TCP パケットによる DDoS 攻撃では、パケットロスが大量に発生しており、攻撃が十分に成功している。

4 まとめ

本研究では、Mirai のソースコードを分析し、安全に運用可能な DoS/DDoS 攻撃を模倣する試験環境を開発した。実証実験の結果から、被攻撃者側の環境と

して小規模なネットワークと標準的な性能のサーバ機器、攻撃者側の環境として標準的な性能の IoT 機器を準備すればよいことが分かった。当初の実証実験では、被攻撃対象機器の Web サーバの搭載メモリを最小値の 384MB に設定した仮想マシンを構成して DoS 攻撃によるメモリの枯渇を狙ったが、Web サーバはほぼ問題なく稼働してしまった。OS のセキュリティ機能を停止する方法も考えられるが実際の運用面で現実的な選択ではないので、本研究ではネットワークの過負荷状況に絞って実証実験を行った。

また、一部の実験の結果から、高性能な IoT 機器において、急激な機器の温度上昇を防止するための安全機能が作動したと思われる性能低下が確認されて、IoT 機器の性能を最大限活用した攻撃とはならなかった。この点に関しても、実際のサイバー攻撃では同じような状況下で DDoS 攻撃が行われていると考え、あえて実験結果を掲載した。

本試験環境を利用すれば、マルウェアによる Bot の感染活動や Bot からの DoS 攻撃を小規模なネットワーク上で実現できるので、DoS 攻撃対策のパケット制御やファイアウォールの研究開発にとどまらず、Bot の活動を監視及び解析するようなハニーポットやサンドボックス等の研究開発を加速的に進めることができる。

参考文献

- [1] Kambourakias, G., Kolias, C. and Stavrou, A.: The Mirai Botnet and the IoT Zombie Armies, Proc. IEEE Military Communications Conference (MILCOM), pp.267–272 (2017).
- [2] 河口綾摩, 空閑洋平, 中村修: Mirai 型 DDoS ボットネットワークの監視環境の構築, マルチメディア, 分散協調とモバイルシンポジウム論文集, pp.1420–1425, 2017.
- [3] 小林稔: Mirai Botnet の検知と対策, Internet Infrastructure Review, vol.33, pp.18–24, 2016.
- [4] 真鍋督, 井口信和: クラウド環境を標的とする DDoS 攻撃の対策演習システムの開発と評価, 情報処理学会インターネットと運用技術シンポジウム論文集, pp.63–70, 2022.
- [5] Mirai-Source-Code,
<https://github.com/jgamblin/Mirai-Source-Code> (2024.04.01)
- [6] RFC4987: TCP SYN Flooding Attacks and Common Mitigations,

<https://datatracker.ietf.org/doc/html/rfc4987> (2024.04.01)