Vol.2019-IOT-44 No.5
2019/3/7

悪性 Botnet 包囲網の Bot による WannaCry のようなマルウェアの活動検知の試み

山之上卓[†1], 長副誉司[†1]

概要：悪性 Botnet 包囲網により, WannaCry のような LAN のスキャンを行うマルウェアの動作検知を試みたことについて述べる．WannaCry はその感染拡大を行うため，組織内外のホストの TCP/445 ポートに接続を試みる．様々なホストの TCP/445 ポートへの接続試行（scan）を行っているホストを見つけることにより, WannaCry に感染しているホストをある程度発見することができる．しかしながら，もし，同様のマルウェアが組織外のホストに対する scan を行わない場合は，組織の出入り口の監視のみではこのマルウェアを発見することは困難である．本論文では，このような，組織外との通信をあまり行わないマルウェアも含めた，WannaCry のような マルウェアの検知方法について述べる．この検知方法を安全に検証するため，本物のマルウェアの代わりに，スキャンを行うマルウェアの動作を真似たプログラムを作成し，利用した．

キーワード：Bonet, Bot, Wiki, P2P, IDS，分散協調,

An Attempt to Detect Malware such like WannaCry by Bots of the Malicious Botnet Capturing Network

TAKASHI YAMANOUE[†1], TAKASHI NAGASOE

Abstract: An attempt to detect malware, such like WannaCry, by bots of the malicious botnet capturing network, is discussed. WannaCry attempts to connect the compromised host to the TCP/445 port of hosts which are connected to networks of inside and outside of an organization, in order to reproduce itself. The host with WannaCry can be detected by finding out the host which is scanning TCP/445 ports of many hosts. However, if a malware, which is similar to WannaCry, does not scan outside of the LAN with the compromised host, it is hard to find out the compromised host by monitoring the traffic at the network doorway of the organization. We discuss the way to detect the host with the such malware using the beneficial botnet. We also made the pseudo WannaCry in order to evaluate the way.

Keywords: Botnet, Bot, Wiki, Ransom Ware, IDS, Distributed, collaboration

1. はじめに

ランサムウェアはインターネット上における脅威の一つである．2017 年，同時多発のランサムウェアでは最大規模の流行となった「WannaCry」は，米ホワイトハウスの発表で，全世界で 150 カ国，30 万件以上の被害を出したとされている[1].

WannaCry は, 感染すると端末内に入っている jpg, mp3, pptx，docx，xlsx などの主要な拡張子をもつファイルを暗号化し，暗号化されたファイルの拡張子は「WNCRY」に書き換わる．暗号化されたファイルの解除には，金銭を要求し，脅迫文を画面上に表示させる．金銭の支払いは，指定されたアドレスにビットコインでの支払うよう指示される．欧州圏では病院内のネットワークが感染被害に遭い，コンピュータシステムがロックされ手術や治療ができなくなった．日本でも様々な被害が発生した[2].

このような WannaCry による被害の拡大を防ぐため，端末内に侵入する前の WannaCry の挙動を検知することによってネットワーク内への侵入や被害拡大を防ぐ必要がある．

我々は, 従来から開発を続けてきた良性 Bot を使って，悪性 Botnet 包囲網を開発している[3][4][5]．良性 Botnet は LAN の Nat の内側に設置する Agent Bot と Agent Bot によって選択収集されたデータを解析する Analyzing Bot により構成された良性 Bot のグループである．

本論文では，WannaCry の機能の一部をもった，偽 WannaCry の開発と偽 WannaCry を使った検知の試みについて述べる. WannaCry は端末内に侵入するために行うポートスキャンを行う．Agent Bot はこのデータを収集し，Analyzing bot はそのデータを解析することにより，WannaCry がネットワーク内に侵入していることやどの端末がポートスキャンを行っているかの検知を行う．偽 WannaCry を WannaCry の代わりに実験 LAN の Host で動かし，Agent Bot でその活動データを収集した所,偽 WannaCry の活動の可能性を識別するデータを得ることが出来た．

その取得したデータを使って WannaCry の活動を検出するシステムを開発中である．

†1 福山大学
Fukuyama University

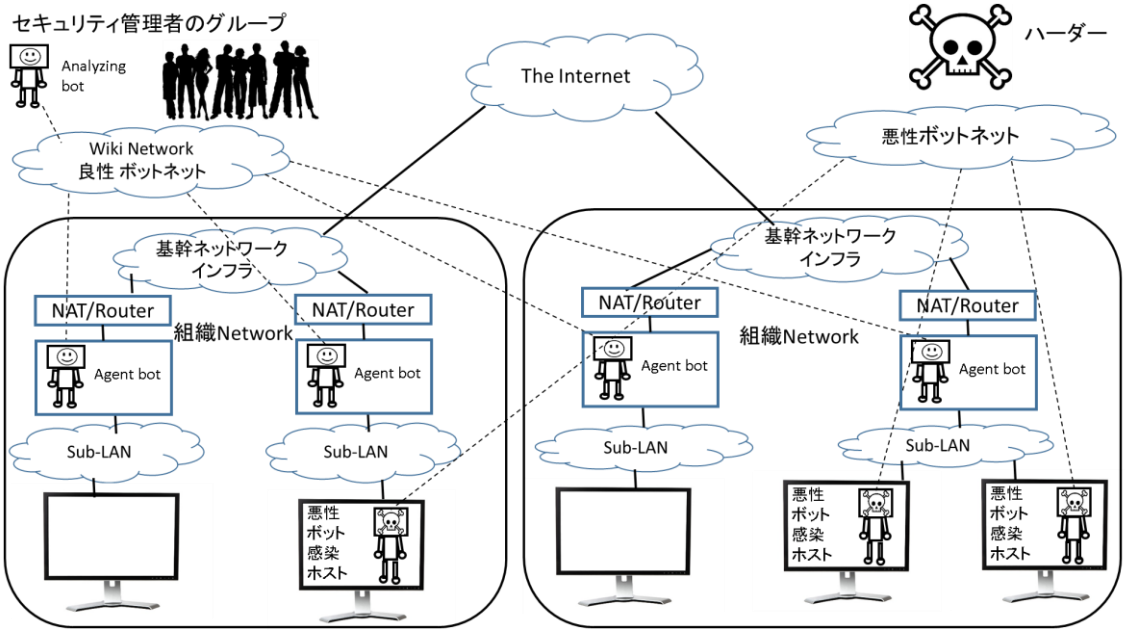

図 1　悪性 Botnet 包囲網(beneficial botnet)の概要

Figure 1　Outline of the beneficial botnet

2. 悪性 Botnet 包囲網

組織のネットワークの出入り口に IDS や IPS を設置して利用することが良く行われている．Communication and Control (C2)サーバとすべての bot の間の通信がそれによって検知可能であるので，組織の出入り口に設置された IDS や IPS は中央集中的な Communication and Control (C2)サーバを持つ悪性 Botnet には有効である．

しかしながら，IDS や IPS の内側の，組織内で行われる P2P 通信をこのような IDS や IPS で検知することは困難である．我々は，Gameover ZeuS のような P2P 通信機能を持つ botnet に対処する悪性 botnet 包囲網(beneficial botnet)を設計している．

2.1 悪性 bonet 包囲網の概要

この論文で述べる悪性 botnet 包囲網(beneficial botnet) は，従来の IDS や IPS の欠点を克服しようとするものである．図 1 に悪性 botnet 包囲網の概要を示す．Agent bot は Sub-LAN と NAT または Router の間に設置される．Agent bot は Sub-LAN と Sub-LAN の外部との通信データを収集する．Analyzing bot は Agent bot が収集したデータを集めて解析し，sub-LAN 内に潜む bot を検出する．すべての Agent bot と Analyzing bot はインターネット上の Wiki のページに書かれた script によって制御される．

2.2 悪性 bonet 包囲網の bot

Beneficial botnet で利用する beneficial bot は wiki ソフトウェアの wiki ページ上に書かれたスクリプトのインタープリタである[6][7][8]．図 2 に beneficial bot の振る舞いを示す．これは以下を繰り返す．

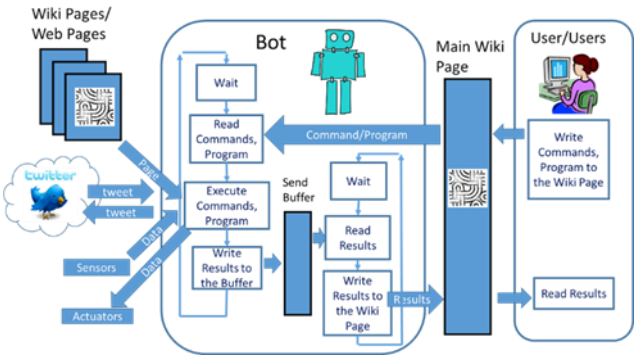

図 2　Bot の振る舞い

Figure 2　Behavior of a bot

1) 一定時間停止する．
2) コマンドとプログラムで構成されたスクリプトを，事前に bot に指示されていた wiki ページ(main wiki)から読み込む．
3) コマンドとプログラムを実行する．プログラムは main wiki の他に，他の wiki ページや web ページを読み込むことができる．もし，bot が agent bot であるなら，コマンドによって，bot が動いているコンピュータのネットワークインターフェース間の通信データを収集することができる．Agent bot は，インターフェース間の通信をコマンドによって制御することもできる．もし，bot が analyzing bot であるなら，R 統計計算システムを用いて，収集されたデータを解析することができる．
4) Bot の実行結果は送信バッファに書きこまれる．送信バッファに書きこまれたデータは，スクリプトが書

かれた wiki ページに書きこまれる.

```
objectPage http://www.[redacted]
device yamaRasPiDp9_1 or yamaRasPiDp9_2 start after no w
command: set readInterval=60000
command: set execInterval=0
command: clear sendBuffer;
command: program ex1
program: s=0;
program: for i=0 to 10
program:   s=s+i
program:   ex("service","putSendBuffer "+s)
program: next i
command: end ex1
command: run ex1
command: ex("service","sendResults.")
result:
0
1
3
6
10
15
21
28
36
45
55
currentDevice="yamaRasPiDp9_1",Date=2018/5/15/ 12:54:17
```

図 3　Bot が実行する Script の例

Figure 3　Example of the Script which is interpreted by a Bot

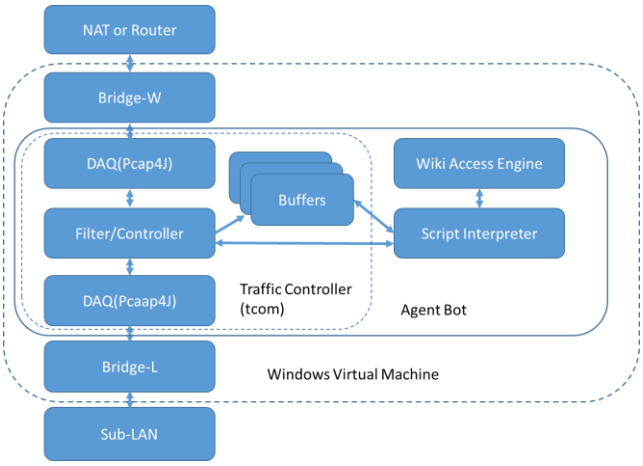

図 4　Agent Bot の構成

Figure 4　Structure of the Agent Bot

図 3 に script を書いた wiki ページの例を示す. result: の行より前の部分が script のコマンドとプログラムであり, result: より後の部分が実行結果である.

コマンドは command: で始まる行に書かれていて, プログラムは program: から始まる行に書かれている. この Script は, 1 から 10 までの和を計算し, その途中経過も含めて, result: より後に書きこんでいる.

コマンドとして, “set pageName <page-name>” と “include <url>” も使うことができる. “set pageName <page-name>” が bot のインタープリタで解釈されたとき, bot は次に読み込む main wiki ページを, 元の main wiki ページと同じ wiki の, <page-name> で示されたページで置き換える. <page-name>の一部としてその時の時間や日付を使うこともできる. これにより, 日時に応じた別のページに実行結果を書きこむことが可能になる. “include <url>” が bot のインタープリタで解釈されたとき, bot はその部分に <url> で示された wiki に書かれている script を埋め込む. これにより, オブジェクト指向プログラミングにおいて複数のオブジェクトが 1 つの class で書かれたプログラムを実行するのと同様に, 複数の bot が同じ script を実行することが可能になる. Include される script を含んだページを class ページと呼ぶことにする. それに対して, main wiki のことを object ページとも呼ぶことにする.

Bot は http を使って wiki ページと通信しているため, Bot が NAT で守られた LAN の中にいた場合でも, main wiki のページをインターネットや Bot から見える場所にある Wiki のサーバに置くことにより, Bot の管理者は Bot を NAT の外部から制御することが可能になる.

2.3 Agent Bot

Agent bot は”Traffic controller” (tcon) を備えた良性 bot である.

Tcon は 2 つの network interfaces とデータ取得ライブラリ(DAQ)と Filter/Controller と複数の Buffer を持っている. DAQ として Pcap4j [9]を使っている.

この Bot の script interpreter は tcon の操作も行う. Bot は 2 つの interface の間の通信データを収集し, main wiki の scriptによってそのデータを main wiki のページに書きこむ. 2 つの interfaces 間の通信の制御することもできる. Network interface の一つを NAT または router に接続し, もう一つを sub-LAN に接続することにより, sub-LAN と sub-LAN の外との間の通信のすべてを収集し, 制御することができる.図 4 に Agent bot の構造を示す.

tcom は以下の buffer を持つ. 通信データはその種類によって分類され, これらの buffer に格納される.

- Packet-history

この buffer は 2 つの interface 間で転送されるパケットの情報を格納する.この buffer は同じ source IP address と destination IP address のペア同士を格納するための sub buffer を持っている. Packet の payload の sha1 hash とその packet が転送された時間と packet の情報もペアと一緒に格納される. Sha1 の値は 2 つの異なる sub-LAN 間で行われる P2P 通信を検知するために使われる.

Sub buffer があふれた時, 古い情報が削除される.

- Arp-list

この buffer は Arp 通信をそれが行われた時間と共に

格納する. Arp spoofing の検知に使うことができる.

Agent bot はこれらの buffer から情報を得たり, 操作したりするコマンドを実行できる. 2 つのネットワーク間の通信を制御するコマンドも持っていて, このコマンドを使うことで悪性 bot の通信を見つけた時, その通信を遮断することもできる.

- MAC-list, Domain-list, Dhcp-list

その他, MAC-list, Domain-list, Dhcp-list などのリストを得る機能を持っている.

2.4 Analyzing Bot

Analyzing bot は各 agent bot で収集された情報を集め, それを解析する.

すべての良性 bot の言語プロセッサは, comma separated value (CSV) parser と表操作/表計算関数を備えている. Analyzing bot はこれに加えて R 統計計算システム[10]も備えている. Analyzing bot の script の中に R のプログラムを埋め込むことができる. 良性 bot が元々備えている言語プロセッサと R 統計計算システムの間で値を交換する関数も使うことができる.

3. 偽 WannaCry

WannaCry は,「SMBv1」の脆弱性を利用して自己増殖を行おうとする. その際, 組織内外のホストの TCP445 番ポートに接続を試みる. WannaCry のようなマルウェアが beneficial botnet によって検知できるかどうかを検証するため, WannaCry のように他ホストの TCP/445 番ポートへの接続の試行を繰り返す偽 WannaCry を作成した.

WannaCry が組織内外のホストの TCP/445 ポートに接続を試みるのに対して, 偽 WannaCry はそれが直接接続された LAN 内のホストについてのみ TCP/445 ポートに接続を試みる. WannaCry の場合は組織の出入り口で TCP/445 宛のパケットを監視することにより WannaCry が組織内で活動しているかどうかを推定できるが, もし同様のマルウェアが組織外との通信をあまり行わない場合は組織の出入り口における, そのマルウェアの活動を知ることが難しくなる.

このようなマルウェアが存在する可能性があるので, 我々は, そのようなマルウェアも検知できるような検査システムを開発中である. また, この検知システムの動作検証を行うため, ただたんに, LAN 内のホストの TCP/445 番ポートに対して繰り返し接続を試みる, 危害を加えないソフトウェア, 偽 WannaCry を作成した.

偽 WannaCry はホストのネットワークインターフェースの, IP 接続のための IP アドレスと netmask の情報を入手し, その情報から LAN 内に存在する可能性のあるホストの IP アドレスの範囲を計算し, その範囲にあるホストの TCP/445 番ポートに対して繰り返し接続を試みるものである.

4. 検知手法

偽 WannaCry は LAN 外のホストの TCP/445 番ポートへの接続を行わない. 偽 WannaCry は LAN 内に存在する可能性のあるホストの IP アドレスの範囲の IP アドレスのホストの TCP/445 番ポートに接続しようとするが, LAN 内では, 相手先ホストが存在しない場合, TCP 接続そのものが行われない. したがって, たとえミラーポートを持ったスイッチを使って監視していても, LAN 内の存在しない IP アドレス宛の TCP 通信は監視できない.

しかしながら, ルータや NAT などを使って, LAN 外との通信が可能になっている場合, 偽 WannaCry は少なくともルータや NAT の LAN 側の IP アドレスの TCP/445 番ポートへの接続の試行は行う. この接続を検知することにより, 偽 WannaCry が動いているホストの候補を挙げることは可能になる.

LAN 内でユニキャストの IP 通信が行われる場合, 送信側ホストは, 受信側ホストの MAC アドレスを入手するため, ARP パケットの送受信を行う. ARP パケットを監視することにより, どのホストが, 別のどのホストに IP 接続を行おうとしているか, 知ることができる.

LAN 内における IP パケットを監視していても, 偽 WannaCry が行うようなスキャンが行われているかどうかを知ることはむずかしいが, ARP パケットを監視することにより, 偽 WannaCry が scan 活動を行っていることを識別することが可能になる.

以上のことを踏まえて, 我々は, 以下の方法で偽 WannaCry の活動を識別することにする.

Step 1. Agent Bot において TCP/445 番ポート宛のパケットのリストと, Arp-list の取得を行う.

Step 2. Analyzing Bot において, それぞれの監視対象 LAN の Agent ボットから得られたデータを使って以下を行う.

Step 2.1 TCP/445 番ポートにパケットを送信しているホストの重複しない IP アドレスのリスト(候補リスト)を作成する.

Step 2.2 候補リストのそれぞれの IP アドレスについて. 以下を行う

Step 2.2.1 Arp-list の中で上の IP アドレスを送信元に持ち, Arp request を行うパケットの, 相手先 IP アドレスのリストを作成する.

Step 2.2.2 相手先の IP アドレスのリストについて, 重複していないアドレスのリスト(接続先リスト)を作成する.

Step 2.2.3 候補リストの中で, 接続先リストの要素の数が一定の数を超えたものを, 偽

WannaCry が活動している可能性が高いホストとして出力する．

5. 実験

我々は beneficial botnet を使って実験的な bot 検知基盤を実装し，検知手法によって偽 WannaCry が検知できるかどうかを検証するため，図 5 で示す beneficial botnet を使った bot 検知基盤に偽 WannaCry を配置し，動作させた．なお，Analyzing Bot についてはまだ開発中である．

5.1 Agent Bots の script と実行結果

図 6 に WannaCry を検知するための，agent bot の script を示す．この script は，1 分以上繰り返されている同じ (source IP, destination IP)のペアの unicast 通信の中で，TCP プロトコルで，destination Port が 445 の情報を集めることと Arp-list を収集することを表している．図 7 に，Agent bot が収集したデータが書きこまれた object page の中で，Agent bot の object page の，接続先が TCP/445 番ポートの通信を表している部分，図 8 に，Arp-list の中で Scan を行っている部分を示す．図 6 の script は object page の include コマンドにより読み込まれている．

以下に，図 7 の，接続先が TCP/445 番ポートの通信を表している 1 行を表す．

cmd=get repeating, date="2019/01/20 21:15:01 +0900", no=1081, if=1, smac="b8:27:eb:3a:6b:fa", dmac="bc:5c:4c:5d:1c:cd", prtcl=tcp, sip="192.168.2.100", dip="192.168.2.1", sp=53468, dp=445, sha1payload="no-payload", payloadLength=0, payload=-SYN- .

ここで，*cmd=get repeating* はこの行の通信が *get repeating* コマンドによって採取されたことを示す．*date=...* は採取された日付，*no=1081* は Agent Bot がこの通信に付けたシリアル番号，*if=1* はこのパケットを取得した Agent Bot が動いているホストのインターフェース番号，*smac="..."* はこの通信の送信元 MAC アドレス，*dmac="..."*はこの通信の受信先 MAC アドレス，prtcl=tcp は TCP プロトコル，*sip="..."* は送信元 IP アドレス，*dip="..."*は受信先 IP アドレス，*sp=...* は送信元ポート番号，*dp=445* は受信先ポート番号が 445，*sha1payload="…"* は payload の sha1 ハッシュ値，*paloadLength=0* は payload の長さ，*payload=-SYN-* はこのパケットが TCP の SYN パケットであることを表す．

以下に，図 8 の，Arp-list の 1 行を表す．

cmd=get arp-list, date="2019/01/20 21:24:19 +0900", nif=2, smac="b8:27:eb:3a:6b:fa", sip="192.168.2.100", dmac="ff:ff:ff:ff:ff:ff", dip="192.168.2.247", arp="oparation=1_(REQUEST),ptype=0x0800_(IPv4),htype=1_(Ethernet_(10Mb)),smac=b8:27:eb:3a:6b:fa,sipa=192.168.2.100,dmac=00:00:00:00:00:00,dipa=192.168.2.247".

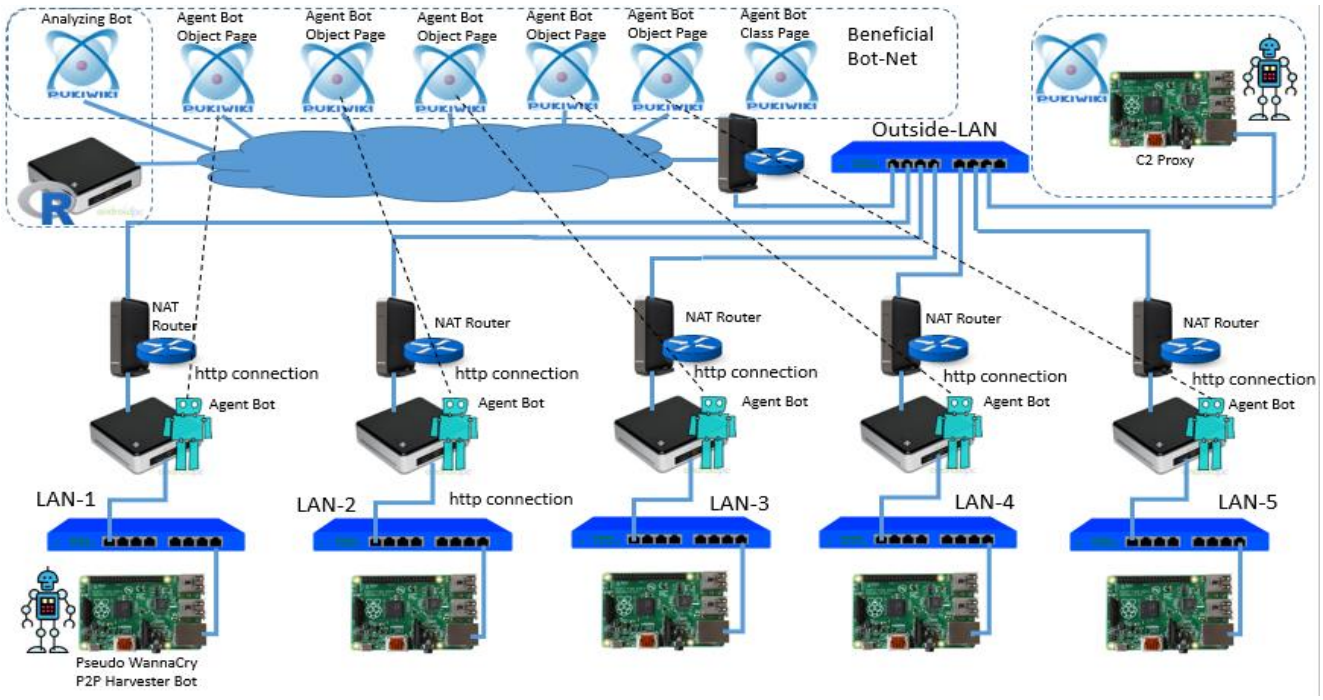

図 5　Bot 検知基盤に偽 WannaCry を配置した実験ネットワーク

Figure 5　Experimental network consists of the beneficial botnet and the Pseudo WannaCry

ここで，*cmd=get arp-list* はこの行の通信が *get arp-list* コマンドによって採取されたことを示す．*date=...* は採取された日付，*if=2* はこのパケットを取得した Agent Bot が動いているホストのインターフェース番号，*smac="..."* はこの通信の送信元 MAC アドレス，*sip="192.168.2.100"*は送信元 IP アドレス，*dmac="ff:ff:ff:ff:ff:ff"* はこの通信の受信先 MAC アドレスがブロードキャストであることを表す．

```
command: set readInterval=180000
command: set sendInterval=180000
command: set execInterval=0
command: tcon clear all.
command: set reportLength=1000
command: clear sendBuffer.
command: program ex1
program:  ex("tcon","set repeating number=20.")
program:  output10=ex("tcon","get repeating unicast over 60000.")
program:  output11=grep(output10,"prtcl=tcp")
program:  output11=grep(output11,"dp=445,")
program:  ex("service","println "+output11)
program:  ex("service","putSendBuffer "+output11)
command: end ex1
command: clear sendBuffer
command: run ex1
command: tcon get arp-list.
command: sendResults.
```

図 6　Agent bot に対する指示を行う Class page の Script

Figure 6　Script of the Class page to direct all of agent bots

```
prtcl=tcp, sip="192.168.2.100", dip="192.168.2.1", sp=51836, dp=445, shalpay
prtcl=tcp, sip="192.168.2.100", dip="192.168.2.1", sp=52308, dp=445, shalpay
prtcl=tcp, sip="192.168.2.100", dip="192.168.2.1", sp=52802, dp=445, shalpay
prtcl=tcp, sip="192.168.2.100", dip="192.168.2.1", sp=53310, dp=445, shalpay
prtcl=tcp, sip="192.168.2.100", dip="192.168.2.1", sp=53816, dp=445, shalpay
prtcl=tcp, sip="192.168.2.100", dip="192.168.2.1", sp=54302, dp=445, shalpay
prtcl=tcp, sip="192.168.2.100", dip="192.168.2.1", sp=54794, dp=445, shalpay
prtcl=tcp, sip="192.168.2.100", dip="192.168.2.1", sp=55240, dp=445, shalpay
prtcl=tcp, sip="192.168.2.100", dip="192.168.2.1", sp=55748, dp=445, shalpay
prtcl=tcp, sip="192.168.2.100", dip="192.168.2.1", sp=55748, dp=445, shalpay
prtcl=tcp, sip="192.168.2.100", dip="192.168.2.1", sp=56204, dp=445, shalpay
rtcl=tcp, sip="192.168.2.100", dip="192.168.2.104", sp=40552, dp=445, shalpay
prtcl=tcp, sip="192.168.2.100", dip="192.168.2.104", sp=46924, dp=445, shalp
```

図 7　Agent bot の Object page の TCP/445 宛のパケットの一部

Figure 7　A part of the object page for an agent bot

```
sip="192.168.2.101", dmac="bc:5c:4c:5d:1c:cd", dip="192.168.2.1", arp="oparation=1_(REQUEST),pty
sip="192.168.2.101", dmac="bc:5c:4c:5d:1c:cd", dip="192.168.2.1", arp="oparation=2_(REPLY),ptype
sip="192.168.2.100", dmac="ff:ff:ff:ff:ff:ff", dip="192.168.2.187", arp="oparation=1_(REQUEST),p
sip="192.168.2.100", dmac="ff:ff:ff:ff:ff:ff", dip="192.168.2.189", arp="oparation=1_(REQUEST),p
sip="192.168.2.100", dmac="ff:ff:ff:ff:ff:ff", dip="192.168.2.191", arp="oparation=1_(REQUEST),p
sip="192.168.2.100", dmac="ff:ff:ff:ff:ff:ff", dip="192.168.2.193", arp="oparation=1_(REQUEST),p
sip="192.168.2.100", dmac="ff:ff:ff:ff:ff:ff", dip="192.168.2.195", arp="oparation=1_(REQUEST),p
sip="192.168.2.100", dmac="ff:ff:ff:ff:ff:ff", dip="192.168.2.196", arp="oparation=1_(REQUEST),p
sip="192.168.2.100", dmac="ff:ff:ff:ff:ff:ff", dip="192.168.2.197", arp="oparation=1_(REQUEST),p
sip="192.168.2.100", dmac="ff:ff:ff:ff:ff:ff", dip="192.168.2.198", arp="oparation=1_(REQUEST),p
sip="192.168.2.100", dmac="ff:ff:ff:ff:ff:ff", dip="192.168.2.207", arp="oparation=1_(REQUEST),p
sip="192.168.2.100", dmac="ff:ff:ff:ff:ff:ff", dip="192.168.2.188", arp="oparation=1_(REQUEST),p
sip="192.168.2.100", dmac="ff:ff:ff:ff:ff:ff", dip="192.168.2.212", arp="oparation=1_(REQUEST),p
sip="192.168.2.100", dmac="ff:ff:ff:ff:ff:ff", dip="192.168.2.214", arp="oparation=1_(REQUEST),p
sip="192.168.2.100", dmac="ff:ff:ff:ff:ff:ff", dip="192.168.2.218", arp="oparation=1_(REQUEST),p
sip="192.168.2.100", dmac="ff:ff:ff:ff:ff:ff", dip="192.168.2.220", arp="oparation=1_(REQUEST),p
sip="192.168.2.100", dmac="ff:ff:ff:ff:ff:ff", dip="192.168.2.222", arp="oparation=1_(REQUEST),p
sip="192.168.2.100", dmac="ff:ff:ff:ff:ff:ff", dip="192.168.2.225", arp="oparation=1_(REQUEST),p
sip="192.168.2.100", dmac="ff:ff:ff:ff:ff:ff", dip="192.168.2.227", arp="oparation=1_(REQUEST),p
sip="192.168.2.100", dmac="00:0c:29:2c:95:05", dip="192.168.2.102", arp="oparation=2_(REPLY),pty
sip="192.168.2.100", dmac="00:0c:29:2c:95:05", dip="192.168.2.102", arp="oparation=2_(REPLY),pty
sip="192.168.2.100", dmac="00:0c:29:2c:95:05", dip="192.168.2.102", arp="oparation=2_(REPLY),pty
```

図 8　Agent bot の Object page の一部の arp-list の一部

Figure 8　A part of the object page for an agent bot

*dip="192.168.2.247"*は arp request により MAC アドレスを要求する ip アドレス，*arp="operation=1_(REQUEST), ..."* は，このパケットが arp request パケットであることを表す．

この arp-list には，上の行の他に，同じ sip=192.168.2.100 から様々なホストに対する arp request パケットが送信されている通信が記録されている．

5.2　Analyzing Bot の script

Analyzing bot は異なる sub-LAN で動作している Agent bot が集めたパケットのデータを読み込んで，それぞれの LAN で WannaCry (偽 WannaCry)が感染している可能性が高いホストを列挙する．

図 9 に Analyzing bot の，開発中の script の一部を示す．ここで，

```
'
'step 2.1 Make the candidate host list
'         which is sending syn to hosts of tcp 445
page=ex("connector", "getPage "+urls(i))
…
outx="eval {v1 = "+vals1+"; v2=unique(v1)}"
hostLists=ex("ex1","getVector   \"#\" v2")
hostListLen=tokenize(hostLists,"#",hostList)
```

は，「4. 検知手法」の Step.2.1 の「TCP/445 番ポートにパケットを送信しているホストの重複しない IP アドレスのリスト(候補リスト)を作成する.」を行っている．重複しないリストを作成するため，R の unique 関数を利用している[10].

```
'
' step 2.2.1 Get arp request infromation list,
'
pageR2=grep(pageR,"cmd=arp-list")
pageR3=grep(pageR2,"sip="+hostList(j))
pageR4=grep(pageR3,"oparation=1_(REQUEST)")
parseCsv(pageR4, Table, RowLabel, ColumnLabel)
```

は，「4. 検知手法」の Step 2.2.1　の「Step 2.2.1 Arp-list の中で上の IP アドレスを送信元に持ち，Arp request を行うパケットの，相手先 IP アドレスのリストを作成する.」を行っている．

```
'
' step 2.2.2 Get the number of arp request destination ip-s.
'
getColVector(Table,
RowLabel,ColumnLabel("dip"),dipVector)
vals1="c("+vec2csv(dipVector)+")"
outx="eval { vscan="c("+vals1+"); vlx=unique(vscan)}"
suspLists=ex("ex1","getVector \"#"\" vlx")
suspListLen=tokenize(suspLists, "#", suspList)
```

は、「4. 検知手法」の「Step 2.2.2 相手先の IP アドレスのリストについて，重複していないアドレスのリスト(接続

先リスト)を作成する.」を行っている．ここでも重複しないリストを作成するため, R の unique 関数を利用している．

```
program: for i=0 to n-1
program:   'step 2.1 Make the candidate host list
program:   '      which is sending syn to hosts of tcp 445
program:   page=ex("connector", "getPage "+urls(i))
program:   pageR=getResultPart(page)
program:   pageR2=grep(pageR,"prtcl=tcp")
program:   pageR3=grep(pageR2,"dp=445")
program:   repeatingLines=grep(pageR3,"cmd=get repeating");
program:   parseCsv(repeatingLines, Table, RowLabel, ColumnLabel)
program:   getColVector(Table, RowLabel,ColumnLabel("sip"),sipVector)
program:   vals1="c("+vec2csv(sipVector)+")"
program:   outx="eval {v1 = "+vals1+"; v2=unique(v1)}"
program:   hostLists=ex("ex1","getVector  \"#\" v2")
program:   hostListLen=tokenize(hostLists,"#",hostList)
program:   '
program:   'step 2.2 For each host.
program:   for j=0 to hostListLen-1
program:       '
program:       ' step 2.2.1 Get arp request infromation list,
program:       '
program:       pageR2=grep(pageR,"cmd=arp-list")
program:       pageR3=grep(pageR2,"sip="+hostList(j))
program:       pageR4=grep(pageR3,"oparation=1_(REQUEST)")
program:       parseCsv(pageR4, Table, RowLabel, ColumnLabel)
program:       '
program:       ' step 2.2.2 Get the number of arp request destination ip-s.
program:       '
program:       getColVector(Table, RowLabel,ColumnLabel("dip"),dipVector)
program:       vals1="c("+vec2csv(dipVector)+")"
program:       outx="eval { vscan="c("+vals1+"); vlx=unique(vscan)}"
program:       suspLists=ex("ex1","getVector \"#"\" vlx")
program:       suspListLen=tokenize(suspLists, "#", suspList)
program:       '
program:       ' step 2.2.3 If the number exceeded the threshhold value,
program:       '            output the host.
program:       '
program:       if suspListLen>5 then {
program:           output="In the LAN-"+i+", "
program:           output=output+"host_"+hostList(j)+"_may_have_WannaCry"
program:           ex("service","putSendBuffer "+output)
program:       }
program:     next j
program: next i
```

図 9　Analyzing Bot の Script の一部

Figure9　A part of the Script of the Analyzing Bot

```
'
' step 2.2.3 If the number exceeded the threshhold value,
'                   output the host.
'
if suspListLen>5 then {
    output="In the LAN-"+i+", "
output=output+"host_"+hostList(j)+"_may_have_WannaCry"
ex("service","putSendBuffer "+output)
}
```

は，「4. 検知手法」の「Step 2.2.3 候補リストの中で，接続先リストの要素の数が一定の数を超えたものを，偽 WannaCry が活動している可能性が高いホストとして出力する.」を行っている．ここで一定の数は 5 としている．

図 9 のスクリプトはまだデバッグ中でその実行結果は現時点で得られていないが，図 7 と図 8 の Object ページのデータを使った場合，Step. 2.1 で，候補リストとして，{192.168.2.100} が得られる．Step. 2.2.2 において，192.168.2.100 から送信される arp request の相手先 IP アドレスのリストとして{…, 192.168.2.197, 192.168.2. 189, 192.168.2.191, 192.168.2.195, 192.168.2.196, 192.168.2.197, …}が得られる．Step 2.2.3 において，相手先の IP アドレスのリストの大きさが 5 より大きいので，192.168.2.100 は偽 WannaCry が活動している可能性が高く，また、そのホストが接続されている LAN は 0 番目なので，

In the LAN-0, host_192.168.2.100_may_have_WannaCry

が Analyzing Bot の Object Page に出力される．

6. 4　関連研究

6.1 AAFID

Autonomous Agents for Intrusion Detection (AAFID)[11]は我々の良性 botnet と同様に分散 IDS の agent の集合である．このシステムは, agents と transceivers と monitors で構成されている．AAIF の agent と我々の agent bot は，どちらともコマンドによって制御され，通信データを収集する部分で類似している．Agent AAFID の agent は client host に install されているのに対して，我々の agent bot は LAN とそのルータ又は NAT ルータの間に設置される．我々の良性 botnet の管理者は agent bot を client host のそれぞれに install する必要はない．AAFID の monitor と transceiver は，双方とも agent などからデータを集めて，それを解析する部分で類似している．AAFID の monitor は wiki ページの script で制御されないのに対して，我々の agent bot や analyzing bot は wiki ページの script で制御される．Agent 間の通信方式については，AAFID については定義されていないのに対して，我々の botnet は wiki API を使っている．

6.2 Stealthwatch で WannaCry に感染した端末を検出する方法

Cisco Stealthwatch は，各ネットワーク機器からフルフローを収集することで組織内ネットワークの通信状況を可視化し，通信特徴から異常を検知するセキュリティ製品である．文献[12]で Cisco Stealthwatch を使うことで，現在猛威を振るっている WannaCry に感染した端末を検出する方法が紹介されている．実際の WannaCry は組織内外のホストの TCP/445 ポートへの接続を試みるため, Stealthwatch でその活動を検出することが可能である．しかしながら，偽 WannaCry のような，LAN 内に閉じた scan を行うマルウェアについては，その LAN を構成するネットワーク機器が Stealthwatch に対応したフルフローを収集できるものでない限り，その活動を検知することは難しい．

これに対して beneficial botnet の場合, Agent bot が組織ネットワーク管理外の, NAT で接続された LAN の，その LAN

と NAT の間に設置されていれば，偽 WannaCry のようなマルウェアを検出することが可能である．

6.3 Man in the Middle Attack

我々の agent bot は一種の man in the middle attack[13]を行っていると解釈することもできる．Agent bot によって，sub-LAN 内の多くの通信を制御可能である．我々は Agent bot が dark side に行かないように注意する必要がある．

6.4 KASEYA and UNIFAS

Kaseya のパソコン管理システム[14]や Furuno Systems の無線 LAN アクセスポイント管理システム(UNIFAS)[15]は beneficial botnet と同様に，定期的にエージェントプログラムが Web サーバにアクセスし，そこに書かれた指示をエージェントが実行し，結果を Web サーバ側に戻すことを行っている．エージェントプログラムと Web サーバは NAT を超えて相互に通信できる．しかしながら，これらのシステムはセキュリティ強化を目的としたものではない．また，これらのシステムに特化した Web サーバを必要とする．

5 おわりに

現在試作を行っている悪性 botnet 包囲網(beneficial botnet)によって，WannaCry のようなマルウェアの活動を検出できる可能性があることを示した。我々の beneficial botnet は script を格納するための wiki ページとその script の interpreter から構成されている．評価実験を行う為，悪性 botnet の通信をまねる偽 WannaCry も作成した．
悪性 botnet 包囲網は LAN 間通信の相関を取ることもできるが，今回の偽 WannaCry の検知については，相関は取っていない．

現時点で LAN-WAN 間通信が非常に遅いので実用的に使う為にはこの問題を改善しなければならない．セキュリティの強化とその検証も必要である．この他，利用しやすくするために，デバッグの方法や環境についても改善する必要がある．

謝辞

本研究の一部は JSPS 科研費 16K00197 の助成を受けて実施しました．良性 botnet およびその開発時に利用した PukiWiki, Java, Pcap4J, Eclipse, Eclipse Egit, M2Eclipse, Apache, Apache http client, twitter4j, Raspberry Pi, Raspbian の開発者，実験の実施を手伝ってくれた学生諸君に感謝します．

参考文献

[1] ランサムウェア「WannaCry」とは？感染対策・被害事例・復元方法, https://net-perfect.jp/ransomware-wannacry, as of 2019 2/4
[2] 「WannaCry」騒動とは何だったのか？ 感染理由とその対策 (1/2) , http://www.itmedia.co.jp/news/articles/1705/17/news106.html, as of 2019 2/4
[3]Yamanoue, T., Oda, K., Shimozono. K., “Capturing Malicious Bots using a Beneficial Bot and Wiki”, 2012, In Proceedings of the 40th annual ACM SIGUCCS conference on User services (Memphis, Tennessee, USA. 15-19 Oct. 2012). ACM, New York, NY, 91-96. DOI=https://doi.org/10.1145/2382456.2382477
[4] Takashi Yamanoue, Kentaro Oda, Koichi Shimozono. “A Malicious Bot Capturing System using a Beneficial Bot and Wiki,” 2013, Journal of Information Processing (JIP), vol.21, No.2, pp.237-245.
[5] 村上 順也，山之上 卓, "悪性 Botnet 包囲網における DGA 検知の試み", 情報処理学会 研究報告インターネットと運用技術（IOT）,2018-IOT-42(4),1-8 (2018-06-21) , 2188-8787
[6] Yamanoue, T., Oda, K., Shimozono. K. 2013. “An Inter-Wiki Page Data Processor for a M2M System, “ 2013, In Proceedings of the 4th International Conference on E-Service and Knowledge Management (ESKM 2013), Advanced Applied Informatics (IIAIAAI), 2013 IIAI International Conference on.(Matsue, Shimane, Japan, 31 Aug- 4 Sep. 2013) IEEE, Los Alamitos, CA. 45-50. DOI= https://doi.org/10.1109/IIAI-AAI.2013.48
[7] Yamanoue, T., Oda, K., Shimozono. K., “Experimental Implementation of a M2M System Controlled by a Wiki Network,” 2014, In Applied Computing and Information Technology, Studies in Computational Intelligence, Springer, Vol.553, 121-136.
[8] Yamanoue, T., “Monitoring Servers, With a Little Help from my Bots, “ 2017, In Proceedings of the 45th annual ACM SIGUCCS conference on User services (Seattle, Washington, USA. 01-04 Oct. 2017). ACM, New York, NY, 173-180. DOI=https://doi.org/10.1145/3123458.3123461
[9] Pcap4J, https://www.pcap4j.org/, as of 2019 2/4
[10] Ihaka, R., and R. Gentleman. “R: a language for data analysis and grapics”. 1996, J. Comp. Graph. Stat. 5:299-314. Available via http://www.R-project.org.
[11] Spafford, E. H. and Zamboni, D., “Intrusion detection using autonomous agents”, 2000, Elsevier, Computer Networks vol.34, pp.547-570.
[12] 吉池 克史, “Stealthwatch で WannaCry に感染した端末を検出する方法”, 2017-5, Cisco Japan Blog, https://gblogs.cisco.com/jp/2017/05/detect-wannacry-with-stealthwatch/
[13] Conti, M., Dragoni, N. and Lesyk, V. 2016 “A Survey of Man In The Middle Attacks”, 2016, IEEE Communications Surveys & Tutorials, Vol. 18, Issue 3, IEEE, 2027-2051. DOI=10.1109/COMST.2016.2548426
[14] KASEYA, http://www.kaseya.com/
[15] UNIFAS, http://www.furunosystems.co.jp/product/detail/unifas.html