動的解析による BOT コマンドの自動抽出

星澤 裕二† 岡田 晃市郎† 太刀川 剛†

†株式会社セキュアブレイン 〒102-0083 東京都千代田区麹町 2-6-7

E-mail: †{yuji_hoshizawa, kouichirou_okada, tsuyoshi_tachikawa}@securebrain.co.jp

あらまし 本稿では，文字列操作関連の API 呼び出しを監視および制御し，調査対象となる BOT のコマンドを取得する方法を提案する．BOT 対策を考える上で，BOT の感染時の挙動だけでなく，感染後，ハーダーからの指令によりどのような動作を行うかを知ることは重要である. 静的解析によりコマンド受信後の動作を含む BOT の完全な挙動を把握することは可能であるが，短時間ですべての情報を得ることは難しい．また，多くの解析エンジニアを確保しなければ大量の BOT を処理することができない．本稿で提案するシステムでは，これらの問題を解決し，自動的に短時間で BOT コマンドを入手できるようにする．初期実験においては，BOT を制御するコマンドを自動的に抽出して BOT に送信し，各コマンド受信後の BOT の振る舞いを解析することが確認できた．

キーワード ボット，コマンド，自動解析，動的解析，ハーダー

Extracting BOT commands automatically by dynamic analysis

Yuji HOSHIZAWA† Koichiro OKADA† and Tsuyoshi Tachikawa†

†SecureBrain Corporation 2-6-7 Kojimachi, Chiyoda-ku, Tokyo, 102-0083 Japan

E-mail: †{yuji_hoshizawa, kouichirou_okada, tsuyoshi_tachikawa}@securebrain.co.jp

Abstract In this research, we monitor and control the string manipulation functions of the Win32 API, and suggest a way of obtaining commands of target BOT. In the study of anti-BOT, it is important to know not only the behavior of BOT when infecting, but how it runs after infection by the commands from herder. Grasping all behavior of BOT including operation after commands reception is possible by static analysis, but getting all information in a short time is difficult. Also, it requires many analysis engineers to analyze and process the enormous amount of BOT. The system suggested in this paper solves these problems and enables us to get BOT commands quickly and automatically. In the early experiments, we confirmed the system extracts the commands and sends them to BOT automatically, and analyzes the behavior of BOT after the reception of each BOT command.

Keyword BOT，Command，Automated Analysis，Dynamic Analysis，Herder

1. はじめに

BOT の動作を短時間で知るためには，動的解析(ブラックボックス解析とも呼ばれる)手法による挙動解析が用いられる．しかし，BOT の処理の多くはハーダーから送信されるコマンドに応じたものであるため，通常，隔離環境で実施される動的解析では BOT の一部しか解析できないことになる．つまり，外部からのトリガ(ハーダーによる指令)によって発動する BOT に対して動的解析手法は有効なものではないということである．

一方，静的解析(ホワイトボックス解析とも呼ばれる)手法を用いた場合，コマンド処理部分も含めすべてのコードを解析することが可能である．しかし，BOT によっては 100 個以上のコマンドを持つものもあり，短時間で容易に解析結果を得るというわけにはいかない．

現時点においては，どちらの解析手法を利用しても問題があるため，大量に出現している BOT へは短時間で挙動を解析できる新しい仕組みが必要である．

本稿では，動的解析手法を用い自動的に短時間で BOT コマンドの抽出を行い，さらに，抽出したコマンドをハーダーになりすまして送信し，BOT にコマンドに応じた処理をさせる方法を検討する．

2. 提案手法

本稿では，BOT コマンドを自動的に抽出するために標準ライブラリなどで提供される文字列処理関数に戻り値の書き換えなどの処理を追加する方式を検討する．また，抽出した BOT コマンドを BOT へ自動的に順次送信して，BOT コマンド受信後の BOT の挙動を調査可能にする．

2.1. BOT コマンド

BOT をコントロールする BOT コマンドは表 1のようにコマンドとコマンドのパラメーターで構成され， 通常，"<bot command> [parameter] […]"といった書式になっている．コマンドによってはパラメーターが無い場合もある．

ハーダーは IRC サーバー(C&C サーバー)を介してこうした BOT コマンドを BOT へ送信し，ボットを自在にコントロールする．

表 1. SDBOT の BOT コマンド(一部抜粋)

コマンド書式	内容
killthread <thread#> [thread#]	指定したスレッドを終了させる
udp <host> <# of packets> <packet size> <delay> [port]	指定ホストへ UDP パケットを送信する. パケット数, パケットサイズ, 送信間隔, ポート番号が指定可能
download <url> <destination> <action>	指定 URL からファイルをダウンロードする. action が 1 の場合, ファイルを実行する
remove	自分自身を削除する
visit <url>	指定したサイトを訪問する
sysinfo	感染ホストの OS バージョン, プロセッサ速度, メモリサイズなどを通知する

過去に発見された BOT のコード(図 1)を調査したところ，ハーダーから送信された BOT コマンドの文字列を strcmp 関数などの文字列処理関数で比較し，コマンドに応じた処理へ分岐していることがわかった.

```
WORM_TOXBOT.B (W32.IRCBot)
push    offset Str2; ".bot.sleep"
push    ebx             ; Str1
call    strcmp
pop     ecx
pop     ecx
test    eax, eax
jnz     short loc_4043BF
```

```
WORM_CODBOT.U (W32.Toxbot)
mov     eax, [esi]
test    eax, eax
jz      short loc_40A68A
push    [esp+arg_0]     ; char *
push    eax             ; char *
call    _strcmpi
pop     ecx
test    eax, eax
pop     ecx
jz      short loc_40A680
```

```
WORM_RANDEX.AI (W32.Randex.GEL)
push    edi
push    offset aIrcraw  ; "ircraw"
call    l_strcmp        ; static link
pop     ecx
test    eax, eax
pop     ecx
jnz     short loc_408B02
```

図 1. BOT のコマンドチェックコード例

2.2. システム構成

提案方式のシステム構成を図 2に示す.

Victim Host は BOT を実行する環境で, 動作中の BOT の API 呼び出しを監視し，呼び出された API 名，パラメーター, 戻り値をログに記録できる. また, Victim Host の中には BOT コマンド抽出のために文字列処理関数などに処理を追加する BOT コマンド抽出モジュール(BOT Command Extractor)がある.

Internet Emulator は，Victim Host 上で動作する BOT に擬似インターネット環境を提供する仕組みである. DNS サーバーや Web サーバー，メールサーバーなどの各種サーバーがあり，BOT からのさまざまなリクエストに適切なレスポンスを返すようになっている．サーバーへのアクセスはログに記録できる．また，Victim Host 上で動作する BOT コマンド抽出モジュールと連携して BOT コマンドを自動的に抽出できるようにする機能を追加した IRC サーバーも有する．さらに，抽出した BOT コマンドを自動的に順次送信し，ハーダーをエミュレーションする仕組みにしている.

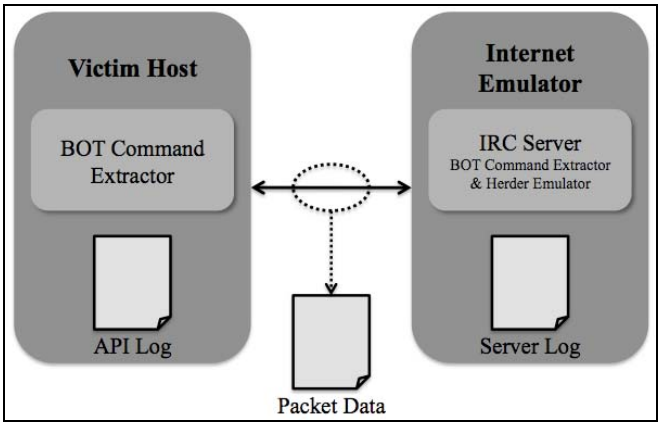

図 2. BOT コマンド自動解析システム

2.3. 実装

BOT コマンド自動解析処理の流れを図 3に示す. BOT が IRC サーバーへ接続を試みると，BOT コマンド抽出モードへ移行し，BOT コマンド抽出，BOT コマンドパラメーター調査，ハーダーエミュレーションによる BOT 制御と順次処理を行う.

BOT コマンド抽出モジュールの制御や BOT コマンドの抽出結果は, IRC の PRIVMSG コマンドを使って行われる．これにより，BOT を動作させながらリアルタイムでコマンドの解析を行うことが可能である.

2.3.1. BOT コマンド抽出

BOT が呼び出す文字列処理関数を操作し，BOT がハーダーからのコマンドをチェックする際の文字列比較において比較する文字列が完全に一致する場合も必ず偽(false)を返すようにする. BOT は受信したコマンドと一致する文字列を検索していくため，すべてのコマンド文字列を知ることができる.

次に BOT コマンド抽出処理の詳細を IRC サーバーと

BOT コマンド抽出モジュールのそれぞれについて説明する.

(1) IRC サーバー

(a) BOT が接続してきた場合，BOT コマンド抽出開始メッセージ("START-CMD-EXTRACTOR") を送信し，BOT コマンド抽出モードへ移行
(b) 図 4のコマンド抽出用メッセージを送信する
(c) BOT コマンド抽出モジュールから抽出結果通知(図 5)を受信する

(2) BOT コマンド抽出モジュール

(a) BOT が recv 関数を呼び出すと，コマンド抽出モジュールは，BOT のプロセスメモリー内を調査して静的にリンクされている_strcmpi，strcmp，strstr，strncmp，atol 関数を探し出し，BOT コマンド抽出用関数に置き換える(表 2). これらの関数以外にも動的にリンクされる下表の関数には，あらかじめ BOT コマンド抽出用の処理が追加されている。
(b) BOT が呼び出す文字列処理関数のパラメーターに含まれている文字列によって次のような処理を行う
  (ア) "START-EXTRACTOR"という文字列が含まれている場合，BOT コマンド抽出処理を開始する
  (イ) "%%check%%"という文字列が含まれている場合，前述図 5の書式で抽出結果を IRC サーバーへ送信する

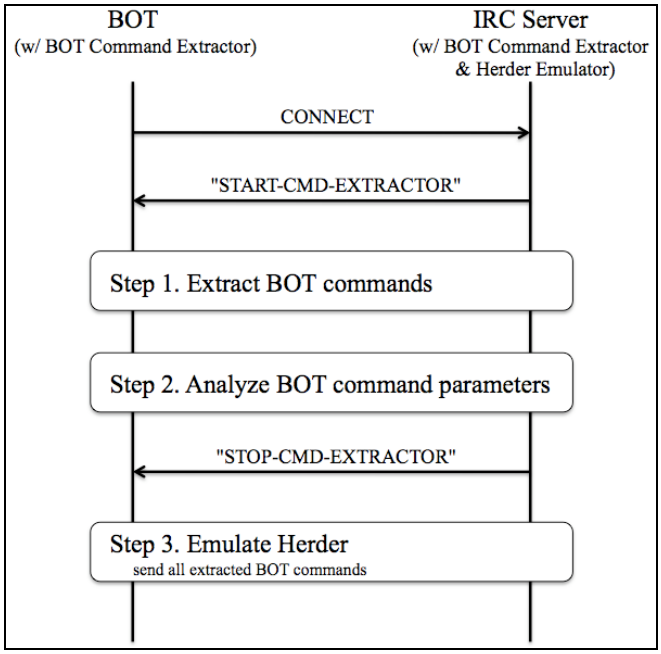

図 3. BOT コマンド自動解析処理の流れ

:zhr!zhr@irc.securebrain.co.jp PRIVMSG <channel name> :<nickname> %%check%%

<channel>にはチャンネル名，<nickname>にはニックネームがセットされる

(例)
:zhr!zhr@irc.securebrain.co.jp PRIVMSG #brap :Brap-cjltkhrw .%%check%%

下は PRIVMSG コマンドではなくコマンド応答(Command Response)を使用したメッセージ
:zhr!zhr@irc.securebrain.co.jp 332 #brap :Brap-cjltkhrw .%%check%%

図 4. BOT コマンド抽出メッセージ例

PRIVMSG #zhr :<function name> <parameter>

<function name>にはパラメーターの処理に使用した関数名, <parameter>には使用パラメーターがセットされる．パラメーターが複数存在する場合，スペース区切りで列挙される

(例)
PRIVMSG #zhr :strcmp bot.repeat %%check%%

この例では，strcmp 関数で"bot.repeat"と"%%check%%"という文字列の比較を行っていることがわかる

図 5 .抽出結果通知例

表 2. 処理追加対象関数

文字列処理
_stricmp(), stricmp(), lstrcmpA(), lstrcmpW(), lstrcmpiA(), lstrcmpiW(), StrToIntA(), StrStrA(), StrStrIA(), StrCmpNA(), StrCmpNIA(), atoi()
ファイル操作
DeleteFile(), DeleteFileA(), DeleteFileW(), FindFirstFileA(), FindFirstFileW(), fopen(), OpenFile()
Windows 終了
ExitWindowsEx()
プログラム起動
CreateProcess(), CreateProcessW(), ShellExecuteA(), ShellExecuteW()
ネットワーク関連
inet_addr(), gethostbyname(), InternetConnect(), InternetOpenUrlA(), InternetOpenUrlW(), DnsQuery_A(), DnsQuery_UTF8(), DnsQuery_W()

2.3.2. BOT コマンドパラメーター調査

BOT コマンドには一つまたは複数個のパラメーターを有するものがある．BOT コマンドパラメーター調査では，抽出したすべての BOT コマンドのパラメーターの有無と各パラメーターの型を割り出す.

前述の BOT コマンド抽出と同様，調査のための処理を追加した文字列処理関数など用いる．

あらかじめ決められた文字列 (<parameter #>%%check%%<bot command #>)をパラメーターにセットした BOT コマンドを送信し，細工された文字列処理関数などで当該文字列の使用を確認する．文字列が使用された場合，使用した関数名を通知する．

例えば, CreateProcess 関数の第一引数として使用した場合には，ファイル名と推測する．これは CreateProcess 関数の第一引数が実行ファイル名を指定するものだからである．

次に IRC サーバーと BOT コマンド抽出モジュールの処理の流れを示す．

(1) IRC サーバー

(a) BOT へ PING コマンドを送信する
(b) PONG コマンドの返信待ち．指定時間内に受信できなければ (f)へ
(c) すべての BOT コマンドパラメーター抽出処理が終了した場合 (g)へ
(d) BOT コマンドパラメーター抽出メッセージを送信する(図 6)
(e) BOT からの返信メッセージ(図 7)待ち．メッセージを受信した場合，メッセージを保存して (a)へ

```
:zhr!zhr@irc.securebrain.co.jp 332 <channel name> :<nickname> <bot command> 1%%check%%<index> 2%%check%%<index> 3%%check%%<index>

<channel>にはチャンネル名，<nickname>にはニックネーム，<bot command>にはパラメーター抽出対象 BOT コマンド，<index>には BOT コマンドインデックスがセットされる

(例)
:zhr!zhr@irc.securebrain.co.jp 332 #brap :Brap-cjltkhrw .ddos.synflood 1%%check%%52 2%%check%%52 3%%check%%52 4%%check%%52 5%%check%%52 6%%check%%52
```

図 6. BOT コマンドパラメーター抽出メッセージ例

```
PRIVMSG #zhr :<function name> <parameter>

<function name>にはパラメーターの処理に使用した関数名, <parameter>には使用パラメーターがセットされる

(例)
PRIVMSG #zhr :atoi 1%%check%%23
PRIVMSG #zhr :inet_addr 1%%check%%52
PRIVMSG #zhr :CreateProcess 2%%check%%13
```

図 7. パラメーター情報

(f) BOT 再起動メッセージ("RESTART-TARGET ")を送信し， (a)へ
(g) BOT コマンド抽出処理終了メッセージ("STOP-CMD-EXTRACTOR")を送信し，結果を BOT コマンドファイル(図 8)へ出力する

```
[cmd.bot.execute]
arglen=0
arg1="INT:atoi"
arg2="FILE:CreateProcess"
msg="'PRIVMSG :Brap-eihpcdg :Password accepted."PRIVMSG :Brap-eihpcdg :couldn't execute file.'"
info=""
index="13"
(中略)
[cmd.ddos.synflood]
arglen=0
arg1="NET:inet_addr NET:gethostbyname"
arg2="INT:atoi"
arg3="INT:atoi"
arg4="INT:atoi"
msg="'PRIVMSG :Brap-okznsxw :Password accepted."PRIVMSG :Brap-okznsxw :synflood: flooding 1-52 port 4 for 2 seconds, 3 ms delay.'"
info=""
index="52"
```

図 8. BOT コマンドファイル

(2) BOT コマンド抽出モジュール

BOT が呼び出す文字列処理関数のパラメーターに含まれている文字列によって次のような処理を行う

(a) "STOP-EXTRACTOR"という文字列が含まれている場合，BOT を強制終了する
(b) "RESTART-TARGET"という文字列が含まれている場合，BOT を再び実行する
(c) "%%check%%"という文字列が含まれている場合，前述図 7の書式でパラメーター情報を IRC サーバーへ送信する

2.3.3. ハーダーエミュレーションによる BOT 制御

ハーダーエミュレーションは，BOT コマンドファイルから送信用 BOT コマンドを生成し，Victim Host で動作している BOT へ順次送信する機能を有する．また，BOT が呼び出す API，DNS サーバーや Web サーバーなどの各種サーバーへのアクセス，ネットワーク通信を監視し，それぞれログに記録できる．

BOT コマンドファイルの BOT コマンドパラメーターは図 6で示したように arg1="INT:atoi" や arg2="FILE:CreateProcess"となっている．ハーダーエミュレーションでは，この値を表 3のルールで変換し，BOT コマンドを生成する．

図 9は，図 8の BOT コマンドファイルから生成した BOT コマンドである．

3. 実験

前述の実装方法によりプログラムを試作して，研究用データセット CCC DATAset 2008 のマルウェア検体(以下，「CCC2008 検体」)を用いて，BOT コマンドの自動抽出と抽出した BOT コマンドの実行に関しての実験を行った．

抽出できた BOT コマンドを表 4に示す．表中の N は数値，S は文字列，A は IP アドレスまたはホスト名，F はファイル名を表す．また，空欄はパラメーターが無いか，または未使用であることを表している．約 20 分間で 99 個の BOT コマンドを抽出することができた．

次に前述のハーダーエミュレーションの仕組みを用い,抽出した BOT コマンドを Victim Host 上で動作している BOT へ送信する実験を行った．この実験では，コマンド受信後の BOT の動作を調査するために, BOT とサーバー間の送受信データや BOT が呼び出す API をログに記録した．

図 10は ddos.phatwonk コマンド受信後の API 呼び出しログである．コマンド受信後，connect 関数でコマンドのパラメーターで指定されたホスト(この例では zhr.securebrain.co.jp)の TCP/1025，TCP/21，TCP/22 へ接続を試みていることがわかる．

表 3. コマンドパラメーター変換ルール

arg#の値	変換後の値
FILE:DeleteFile	C:¥temp¥test.exe
FILE:FindFirstFile	C:¥temp¥test.exe
FILE:CreateProcess	C:¥temp¥test.exe
FILE:ShellExecute	C:¥temp¥test.exe
NET:inet_addr	zhr.securebrain.co.jp
NET:gethostbyname	zhr.securebrain.co.jp
FILE:OpenFile	C:¥temp¥test.exe
URL:InternetConnect	zhr.securebrain.co.jp
URL:InternetOpenUrl	http://zhr.securebrain.co.jp /download/dummy.exe
文字列:strcmp	strcmp された文字列そのもの
INT:atoi	20 から 120 までのランダムな数字
(NONE)	8000 から 9000 までのランダムな数字

```
: 332 #brap :Brap-oyngxovq .bot.execute 72 C:¥temp¥test.exe 8258 8468

: 332 #brap :Brap-oyngxovq .ddos.synflood zhr.securebrain.co.jp 80 104 92 8185 8503
```

図 9. コマンド例

表 4. 自動抽出した BOT コマンド(1/2)

#	コマンド	パラメーター			
		1	2	3	4
1	.bot.repeat	N			
2	.commands.list				
3	.cvar.list				
4	.cvar.get				
5	.cvar.set				
6	.cvar.loadconfig				
7	.cvar.saveconfig				
8	.mac.logout				
9	.axx				
10	.bot.about				
11	.bot.die				
12	.bot.dns	N			
13	.bot.execute	N	F		
14	.bot.id				
15	.bot.nick				
16	.bot.open				
17	.bot.remove				
18	.bot.removeallbut	S			
19	.bot.rndnick				
20	.bot.status				
21	.bot.sysinfo				
22	.bot.longuptime	N			
23	.bot.highspeed	N			
24	.bot.quit				
25	.bot.flushdns				
26	.bot.secure				
27	.bot.unsecure				
28	.bot.command				
29	.irc.disconnect				
30	.irc.action				
31	.irc.dccsend				
32	.irc.getedu				
33	.irc.gethost				
34	.irc.join				
35	.irc.mode				
36	.irc.netinfo				
37	.irc.part				
38	.irc.privmsg				
39	.irc.quit				
40	.irc.raw				
41	.irc.reconnect				
42	.irc.server	N,S	N,S	N,S	
43	.http.download	S			
44	.http.execute	S			
45	.http.update	S			
46	.http.visit				
47	.ftp.download	S			
48	.ftp.execute	S			
49	.ftp.update	S			
50	.http.speedtest				
51	.ddos.udpflood	A	N	N	N
52	.ddos.synflood	A	N	N	N
53	.ddos.httpflood	S	N	N	S
54	.ddos.stop				
55	.ddos.phatsyn	A	N	N	N

表 4. 自動抽出した BOT コマンド(2/2)

#	コマンド	パラメーター			
		1	2	3	4
56	.ddos.phaticmp	A	N	N	
57	.ddos.phatwonk	A	N	N	
58	.ddos.targa3	A	N		
59	.redirect.tcp	N		N	
60	.redirect.gre	A			
61	.redirect.http	N	S		
62	.redirect.https	N	S		
63	.redirect.socks	N			
64	.redirect.socks5	N			
65	.redirect.stop				
66	.rsl.reboot				
67	.rsl.shutdown				
68	.rsl.logoff				
69	.pctrl.list				
70	.pctrl.kill				
71	.pctrl.listsvc				
72	.pctrl.killsvc				
73	.pctrl.killpid	N			
74	.inst.asadd				
75	.inst.asdel				
76	.inst.svcadd				
77	.inst.svcdel				
78	.harvest.cdkeys				
79	.logic.ifuptime	N			
80	.logic.ifspeed				
81	.harvest.emails				
82	.harvest.emailshttp				
83	.harvest.aol				
84	.harvest.registry				
85	.harvest.windowskeys				
86	.plugin.load				
87	.plugin.unload				
88	.scan.addnetrange		N		
89	.scan.delnetrange				
90	.scan.listnetranges				
91	.scan.clearnetranges				
92	.scan.resetnetranges				
93	.scan.enable	S			
94	.scan.disable	S			
95	.scan.startall				
96	.scan.stopall				
97	.scan.start				
98	.scan.stop				
99	.scan.stats				

また, ddos.phatwonk コマンド送信後の BOT からの応答メッセージから ddos.phatwonk [host] [time] [delay]という書式であることが推測できる(図 11).

図 12は redirect.tcp コマンド受信後の API 呼び出しログである．コマンド受信後，listen 関数でリクエストの受信待機を開始していることがわかる．

4. まとめ

提案手法を実装した試作プログラムの実験により，自動的にパラメーターを含む BOT コマンドを抽出でき，また，抽出した BOT コマンドを動作中の BOT へ送信し，BOT の振る舞いを解析できることが示された．

この手法を BOT の解析プロセスに盛り込むことによって短時間で挙動解析することができるようになる．

今後，この手法を応用し，他のタイプの BOT のコマンド調査もできるようにするとともに，提案手法の有効性を確認していく．

```
NVCOM.EXE, inet_addr, [zhr.securebrain.co.jp], -1
NVCOM.EXE, DnsQuery_W, [zhr.securebrain.co.jp][1][0][0][1f4e378][0], 0
NVCOM.EXE, gethostbyname, [zhr.securebrain.co.jp], 14185888
NVCOM.EXE, WSASocketA, [2][3][ff][0][0][1], 29424
NVCOM.EXE, setsockopt, [72f0][0][2][ ][4], 0
NVCOM.EXE, htons, [401], 260
NVCOM.EXE, socket, [2][1][0], 29420
NVCOM.EXE, ioctlsocket, [72ec][8004667e][1f4ed60], 0
NVCOM.EXE, connect, [72ec][85.217.202.135:1025][10], -1
NVCOM.EXE, closesocket, [72ec], 0
NVCOM.EXE, htons, [15], 5376
NVCOM.EXE, socket, [2][1][0], 29436
NVCOM.EXE, ioctlsocket, [72fc][8004667e][1f4ed60], 0
NVCOM.EXE, connect, [72fc][85.217.202.135:21][10], -1
NVCOM.EXE, closesocket, [72fc], 0
NVCOM.EXE, htons, [16], 5632
NVCOM.EXE, socket, [2][1][0], 29436
NVCOM.EXE, ioctlsocket, [72fc][8004667e][1f4ed60], 0
NVCOM.EXE, connect, [72fc][85.217.202.135:22][10], -1
```

図 10. API 呼び出しログ(ddos.phatwonk)

```
Herder Emulator >>> BOT
: 332 #brap :Brap-oyngxovq .ddos.phatwonk zhr.securebrain.co.jp 66 30 8595 8058

BOT >>> Herder Emulator
PRIVMSG :Brap-oyng xovq :Phatwonk: flooding zhr.securebrain.co.jp for 66 seconds, 30 ms delay.
```

図 11. IRC サーバーログ(ddos.phatwork)

```
NVCOM.EXE, socket, [2][1][6], 29452
NVCOM.EXE, htons, [58], 22528
NVCOM.EXE, ntohs, [5800], 88
NVCOM.EXE, bind, [730c][204f740][10], 0
NVCOM.EXE, listen, [730c][32], 0
```

図 12. API 呼び出しログ(redirect.tcp)

謝辞

本研究は平成 19 年度に情報通信研究機構から委託を受け実施した「ボット制御命令の模擬が可能なマルウェア解析システム」の成果の一部である．

本研究を進めるにあたって有益な助言と協力を頂いた関係者各位に深く感謝致します．