動的解析によるBOTコマンドの自動抽出

Malware Workshop 2008
2008年10月10日
株式会社セキュアブレイン
星澤裕二・岡田晃市郎・太刀川剛

背景と目的

- 背景
  - 大量発生しているBOTの感染を未然に防いだり、感染してしまった場合に被害を最小限に抑えたりするためにBOTの挙動を短時間で知ることが重要
- 目的
  - 短時間でBOTのすべての挙動を知りたい
    - 感染活動だけでなく、ハーダーからの指令を受信後の挙動も解析しなければならない
  - 挙動を知るには・・・
    - 動的解析(ブラックボックス解析)
      - 長所：短時間で容易にマルウェアの挙動を知ることができる
      - 短所：条件によって処理を分岐したり、何かをトリガに状態変化したりするものは解析が困難
    - 静的解析(ホワイトボックス解析)
      - 長所：すべてのコードを完全に解析することが可能
      - 短所：時間がかかる。OSやネットワーク、プログラミング言語など幅広い知識が不可欠

提案手法

- 隔離された動的解析環境で自動的に挙動を解析
- BOTコマンド抽出
  - 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
- ハーダーエミュレーション
  - 抽出したコマンドをハーダーになりすまして送信する

strcmp()変更前

```
char cmd[] = "logout";

if (strcmp("rndnick", cmd) == 0 || strcmp("rn", cmd) == 0) {
        ---snip---
}
else if (strcmp("die", cmd) == 0 || strcmp("d", cmd) == 0) {
        ---snip---
}
        mp("logout", cmd) == 0 || strcmp("lo", cmd) == 0) {
        snip---
}
else if (strcmp("reconnect", cmd) == 0 || strcmp("r", cmd) == 0) {
        ---snip---
}
```

変更前のstrcmp()では、比較対象文字列が"logout"の場合、"rndnick", "rn", "die", "d", "logout"のコマンド文字列が取得できる

API.LOGの例
[strcmp] [rndnick][logout], FALSE
[strcmp] [rn][logout], FALSE
[strcmp] [die][logout], FALSE
[strcmp] [d][logout], FALSE
[strcmp] [logout][logout], TRUE

"lo", "reconnect", "r"は取得できない

strcmp()変更後

strcmp()変更前のAPI.LOG

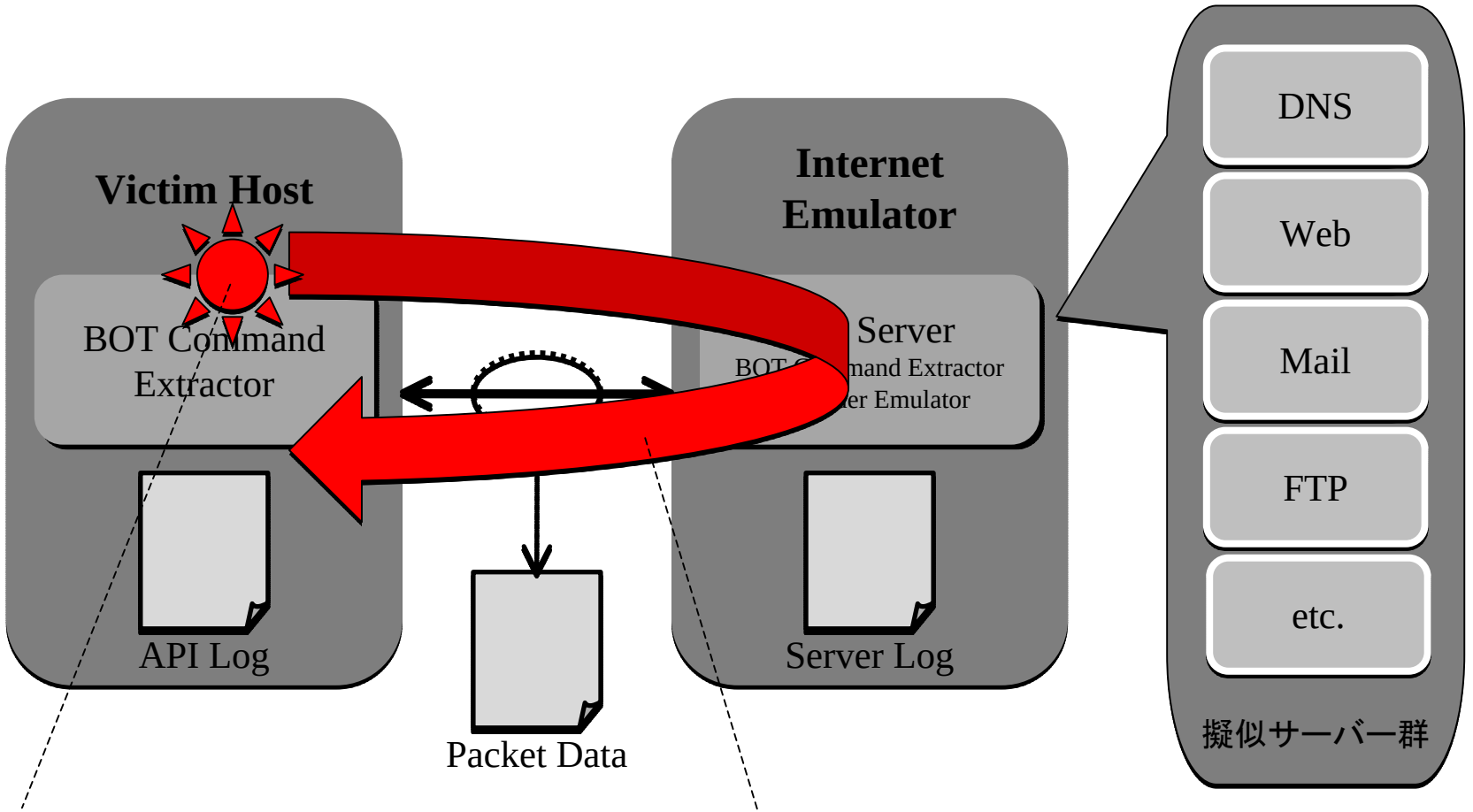
[strcmp] [rndnick][logout], FALSE
[strcmp] [rn][logout], FALSE
[strcmp] [die][logout], FALSE
[strcmp] [d][logout], FALSE
[strcmp] [logout][logout], TRUE

strcmp()変更後のAPI.LOG

[strcmp] [rndnick][logout], FALSE
[strcmp] [rn][logout], FALSE
[strcmp] [die][logout], FALSE
[strcmp] [d][logout], FALSE
[strcmp] [logout][logout], **FALSE**
[strcmp] [lo][logout], FALSE
[strcmp] [reconnect][logout], FALSE
[strcmp] [r][logout], FALSE

- ハーダーから送信されたコマンドをチェックする際の文字列比較で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
- 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取得できる

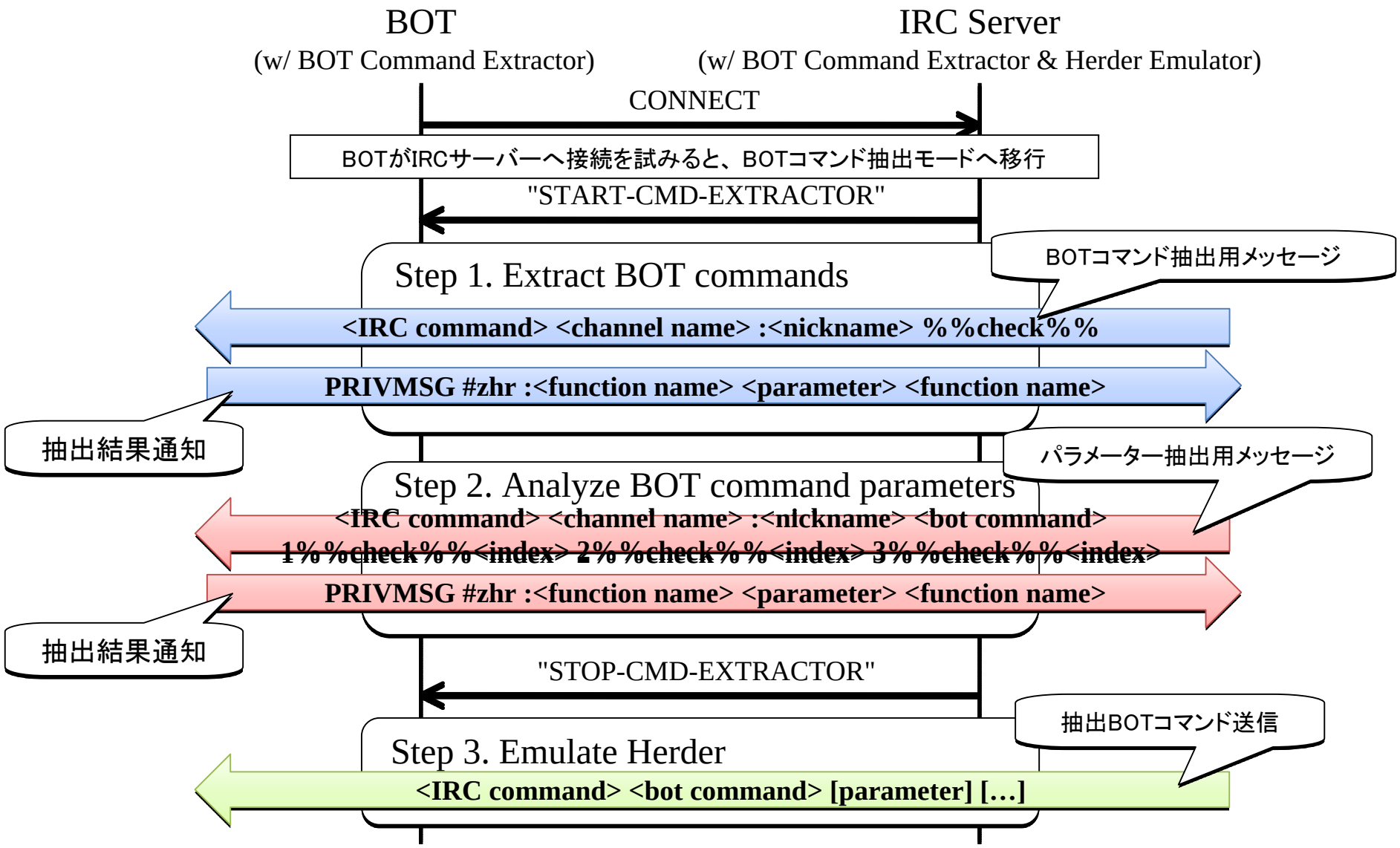
システム構成



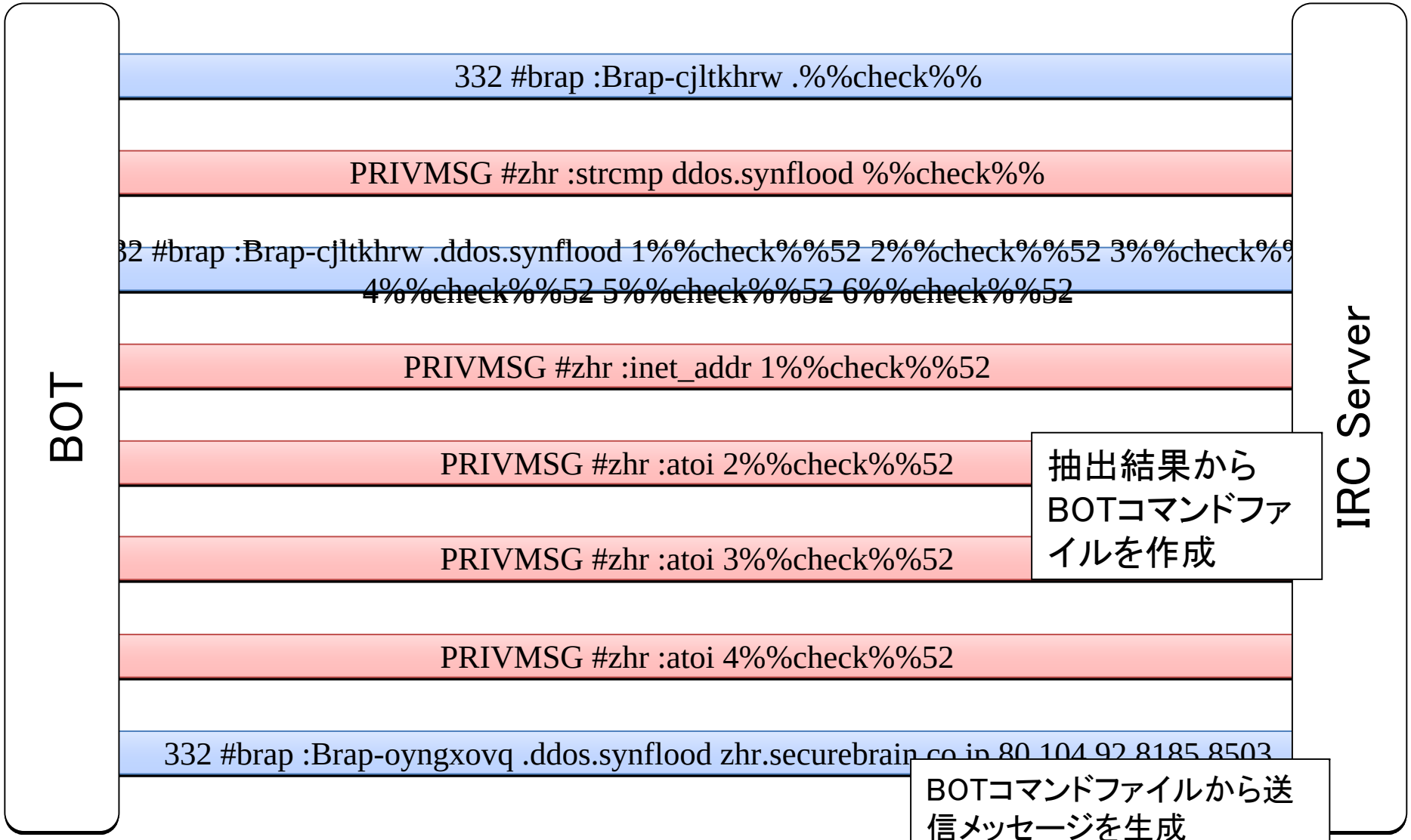
API呼び出しを監視できるようにした Victim Host上でBOTを実行する

Internet Emulator上のIRCサーバーと連携してBOTコマンドを抽出する

処理フロー


BOT
(w/ BOT Command Extractor)
IRC Server
(w/ BOT Command Extractor & Herder Emulator)
CONNECT
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
"START-CMD-EXTRACTOR"
Step 1. Extract BOT commands
BOTコマンド抽出用メッセージ
<IRC command> <channel name> :<nickname> %%check%%
PRIVMSG #zhr :<function name> <parameter> <function name>
抽出結果通知
パラメーター抽出用メッセージ
Step 2. Analyze BOT command parameters
<IRC command> <channel name> :<nickname> <bot command>
1%%check%%<index> 2%%check%%<index> 3%%check%%<index>
PRIVMSG #zhr :<function name> <parameter> <function name>
抽出結果通知
"STOP-CMD-EXTRACTOR"
抽出BOTコマンド送信
Step 3. Emulate Herder
<IRC command> <bot command> [parameter] […]

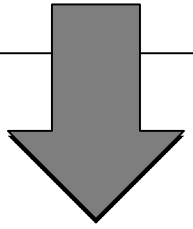
メッセージ送受信例


BOT
IRC Server
332 #brap :Brap-cjltkhrw .%%check%%
PRIVMSG #zhr :strcmp ddos.synflood %%check%%
32 #brap :Brap-cjltkhrw .ddos.synflood 1%%check%%52 2%%check%%52 3%%check%%
4%%check%%52 5%%check%%52 6%%check%%52
PRIVMSG #zhr :inet_addr 1%%check%%52
PRIVMSG #zhr :atoi 2%%check%%52
抽出結果から
BOTコマンドファ
イルを作成
PRIVMSG #zhr :atoi 3%%check%%52
PRIVMSG #zhr :atoi 4%%check%%52
332 #brap :Brap-oyngxovq .ddos.synflood zhr.securebrain.co.jp 80 104.92.8185.8503
BOTコマンドファイルから送
信メッセージを生成

BOTコマンドファイル

- 抽出結果からBOTコマンドファイルを作成
- パラメーター変換ルールを適用し送信メッセージを生成

```
[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."]
flooding 1-52 port 4 for 2 seconds, 3 ms delay.'"
info=""
index="52"
```



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 .ddos.synflood zhr.securebrain.co.jp 80 104 92 8185 8503
```

CCC2008検体での実験結果 1/2

- 99個のBOTコマンドを自動的に抽出
- 処理時間は約20分間

#	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				

数値

ファイル名

#	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				

IPアドレス

文字列

CCC2008検体での実験結果 2/2

- ddos.phatwonk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のTCP/1025、TCP/21、TCP/22へ接続を試みていることがわかる

```
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
```

まとめ

- 提案手法を実装した試作プログラムの実験で自動的にパラメーターを含むBOTコマンドを抽出できた
- 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た
- 提案手法を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

ご清聴ありがとうございました

Malware Workshop 2008
2008年10月10日
株式会社セキュアブレイン
星澤裕二・岡田晃市郎・太刀川剛