G-7

静的解析と動的解析を組み合わせた IoT デバイスに対するマルウェア解析システムの検討

A study on analysis system of malware for IoT devices combining static and dynamic analysis

○房安　良和[1]，泉　隆[2]，香取　照臣[2]（日本大学）
*Yoshikazu Fusayasu[1], Takashi Izumi[2], Teruomi Katori[2] (Nihon University)

The IoT (Internet of Things) is a concept that connecting various devices to the Internet. With the spread of IoT devices, various cyber attacks have been confirmed. In this paper, we examined the configuration of the malware analysis system.

1. まえがき

近年, IoT の普及によって様々なデバイスがインターネットに接続されるようになった．これに伴い，IoT デバイスに感染し DDoS 攻撃等を実行させる Mirai[1]や，ストレージを破壊し，デバイスを使用不能にする BrickerBot[2]をはじめとするマルウェアの出現が確認され, 多くの IoT デバイスへと感染被害が拡がった．多種多様なマルウェアの実行を阻止する必要があるが, IoT デバイスに対するセキュリティ対策は, 現状, 十分に行われていない. そこで, セキュリティ対策を講じるために，まず IoT デバイスに対するサイバー攻撃を分析する必要があると考え，先行研究[3]では不正アクセスやマルウェアを収集するハニーポットを構築した. 本研究では, そのハニーポットで収集したマルウェアを解析するシステムの構築を目的とする.

本稿では，マルウェア解析システムの構成及び解析手法についての検討を行った.

2. ハニーポット/マルウェア解析システム構成

先行研究で構築したハニーポット(Machine A)と本研究で構築するマルウェア解析システム(Machine B)の構成を Figure1 に示す.

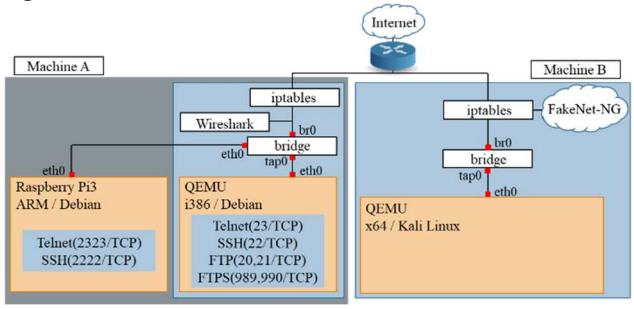

Figure 1. Diagram of Honeypot and analysis system

Machine B では，QEMU を用いて IoT デバイスを模した Linux デバイスをエミュレートし，Machine A で捕獲したマルウェアの解析を行う. なお, エミュレートするデバイスの CPU アーキテクチャについては，解析対象のマルウェアに合わせて選択する. Machine B を構築するために使用するソフトウェアを Table1 に示す.

Table 1. Configuration of Machine B

Categories	Programs to use	Purpose
Access controller	iptables	・To transfer communication generated from malware to Fakenet-NG ・To prohibit communication with the Internet
Internet service simulator	FakeNet-NG	・To emulate the Internet ・To collect communication packet(pcap file)

3. マルウェア解析システム構成

Machine B で使用する解析ソフトウェアとそのソフトウェアを用いた解析内容について, 表層解析, 動的解析, 静的解析に分け，Table2 に示す. 静的解析結果を用いたマルウェアの書換え及び, 動的解析を繰り返すことで, マルウェア内のコードを網羅的に実行する. なお, マルウェアの実行で発生する外部との通信に対応するため, FakeNet-NG を用いてインターネット及び C&C サーバ等との通信で発生する通信のエミュレートを行う.

Machine B での解析結果は, マルウェア情報として利用できるよう, ハッシュ値や保有機能等をまとめ, データベース化する.

Table 2. Analysis contents on Machine B

Analysis method	Analysis tools	Analysis contesnts
Surface analysis	PEframe	Information directly written in malware
Dynamic analysis	strace	System calls caused by execution of malware
	pcap file	・C&C server's location ・Communication caused by execution of malware
Static analysis	radare2	・Estimating the content of processing that occurs during malware execution ・Debugging and rewriting of malware

3.1. radare2[4]

radare2 は様々な CPU アーキテクチャの実行ファイルに対応した逆アセンブル等のバイナリ解析のためのフレームワークである．フローグラフの表示や実行ファイルの書換え，デバッガ等の機能を有する.

3.2. 静的解析と動的解析の組み合わせ

動的解析に使用する単純な strace の実行のみでは, C&C サーバからの命令で動作をするマルウェアや，解析妨害機能を持つマルウェア等については不十分である．そこで，静的解析として radare2 を用いた条件分岐(jmp 系命令)等の書換えを行い(Figure2)，再度動的解析を行う．これにより，マルウェアの網羅的な実行及び解析が可能となる．なお，全条件分岐に対して書換えを行うと，while

1：日大理工・院(前)・情報 2：日大理工・教員・情報

等のループ処理が存在する場合に変数の値が正しく取得されない等の問題が生じるため，書換えの必要/不要の判別を行う必要がある．

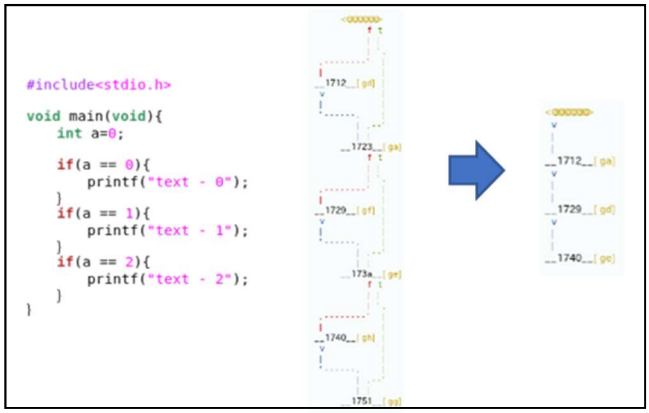

Figure 2. Flow graph created with radare2 before and after rewriting

4. radare2 を用いた書換えに向けた検討

3.2.節で示した解析を行うために, Figure3 に示すソースコードを用いて，実行命令等の可視化や条件分岐への書換えを行った．

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/ptrace.h>

void sub(void){
        printf("sub");
}

int main(void){
        char str[] = "hogehoge";
        int i = 0;
                        }
while(str[i] != NULL){
        printf("%c", str[i]);
        i++;
}
printf("\n");
if(ptrace(PTRACE_TRACEME, 0, 1, 0)==-1){
        printf("-anti debug-\n");
        exit(0);
}
if(i == 0){ sub(); }
return 0;
```

Figure 3. Source code used

4.1. 実行命令の可視化

実行命令等の情報を可視化し，Figure 3 を実行した際の実行箇所及び，各条件分岐情報についてそれぞれ Figure 4，5 に示す．

```
0x0048c27e      e8fdfdffff      call sym.imp.ptrace
                                ⋮
0x0048c289      751c            jne 0x48c2a7
---------- true: 0x0048c2a7  false: 0x0048c28b
                                ⋮
0x0048c295      e8a6fdffff      call sym.imp.puts               ; int puts(const char *s)
                                ⋮
0x0048c2a2      e8a9fdffff      call sym.imp.exit               ; void exit(int status)
                                ⋮
0x0048c2ab      7505            jne 0x48c2b2
---------- true: 0x0048c2b2  false: 0x0048c2ad
0x0048c2ad      e827ffffff      call sym.sub
```

Figure 4. Visualization of instruction execution location (Red:Executed / Green:Not executed)

```
1 0x0048f23a 1 0x48f25a 1 -----
1 0x0048f267 0 0x48f23c 1 0x0048f269
1 0x0048f289 0 0x48f2a7 1 0x0048f28b
0 0x0048f2ab 0 0x48f2b2 0 0x0048f2ad
```

Figure 5. Visualization of execution results at each branch (1:Executed / 0:Not executed，Red:Rewritten / Green:Not Rewriteen)

Figure 4，5 における各実行箇所の判定は，シングルステップ実行を用いた実行アドレスの取得及び逆アセンブル結果との比較で行った．また，Figure5 では，実行(1)/未実行(0)，条件分岐命令のアドレス，True 時の移動先アドレス，False 時の移動先アドレス，の順に表示している．

Figure4 では，各アドレス及び命令について，文字色を用いて実行(赤)/未実行(緑)の状態を示している．これにより，プログラム実行時のデバッガ使用が検知され，exit 関数によって，プログラムが終了したことがわかる．

Figure5 では，各条件分岐のアドレスについての実行/未実行に加え，文字色によって条件分岐に対する書換え有(赤)/無(緑)を表示している．これによって，実行毎の各実行箇所を視覚的に容易な判断を可能とし，マルウェアの手動解析を行う際の補助に繋がると考える．上記は，逆アセンブルした結果を静的に利用しており，間接的な関数の参照に対応することができない．プログラムの動作をより正確に把握するために今後，改善する必要がある．

4.2. 解析手法の実現に向けた検討

3.2 節で述べた本研究における解析手法の実現に向け検討した解析全体の流れを Figure6 に示す．

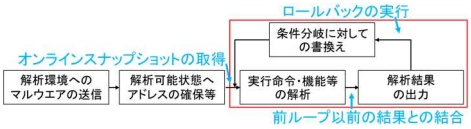

Figure 6. Examined the method of the malware analysis

Figure6 内赤枠部において，解析～出力・書換えを繰り返すことで複数回の実行によってマルウェア内のコードを網羅的に実行する．

4.3. 実行ファイルの書換え

4.1.節に示した書換え前の条件分岐情報(Figure5)を用いて，複数回に分けた条件分岐の書換えを行った．条件分岐の書換えは移動元のアドレス値が小さい順に，以下の条件に一致したアドレスに対して書き換えを行った．

・jmp 系の命令が実行されており(1)，かつ移動先アドレスのどちらかが未実行(0)

・書換え以前に当該アドレスを一度も実行していない

紙面の都合上，結果について省略するが，2 回の書換えによってソースコードの while ループ部分以外が網羅的に実行された．while ループ部分を書換えた場合に無限ループが発生したため，今後は自動的に処理を中断する等の対策を行う．また，条件式等から明らかに無駄と判断できる書換え処理を削減するため，静的解析から取得する情報を増やす等の対策を行う．

5. まとめ

本稿では，マルウェア解析システムの構成及び実行ファイルを用いた解析手法の検討を行い，手法の実現性を検証した．今後は，間接的な関数参照や無限ループへの対策を行い，実際のマルウェアへの適用を進める．

6. 参考文献

[1] WIRED:「The Reaper IoT Botnet Has Already Infected a Million」, https://www.wired.com/story/reaper-iot-botnet-infected-million-networks/,2019-09
[2] radware:「Bricker PDoS Attack: Back With A Vengence」, https://security.radware.com/ddos-threats-attacks/brickerbot-pdos-back-with-vengeance/, 2019-09
[3] 房安良和・小寺建輝・泉隆:「ハニーポットを用いた IoT デバイスに対するサイバー攻撃の分析」, 平成 30 年電気学会全国大会,3-089,2018-03
[4] github:「radare/radare2 : unix-like reverse engineering framework and commandline tools security」,https://github.com/radare/radare2, 2019-09