TOMOYO Linux を用いた Linux マルウェアの動的解析環境構築に向けた初期的検討

吉村 豪康[†1]　橋本 正樹[†1]　辻 秀典[†1]　田中 英彦[†1]

概要：近年，新種のマルウェアが急激に増えており，2015 年の 1 年間だけで 1 億 4000 万種を超える新種が発見されている．マルウェアへの対策を考える上では，その挙動を明らかにすることが重要であり，その方法の一つに動的解析があるが，これまでの動的解析システムは，主に Windows マルウェアを対象としており，Linux マルウェアを対象とした動的解析システムについては検討が不十分である．そこで本研究では，マルウェア検体を実行する犠牲ホストとしてセキュア OS である TOMOYO Linux を用いた Linux マルウェア動的解析システムを構築し，Linux マルウェアを動的解析した結果を考察する．本検証では，TOMOYO Linux の持つプロセスの振る舞い学習機能をマルウェアの挙動追跡に応用することで，マルウェアの挙動を的確に把握し，複数プロセスが連携するようなマルウェア（多段型マルウェア）についても抽象度を上げた観点から追跡できることを確認した．

キーワード：MWS，プロセス挙動，マルウェア，動的解析，セキュア OS，TOMOYO Linux，アクセスコントロール

A preliminary study of dynamic analysis systems for Linux malware using TOMOYO Linux

Katsuyasu Yoshimura[†1]　Masaki Hashimoto[†1]　Hidenori Tsuji[†1]　Hidehiko Tanaka[†1]

Abstract: In recent years, the number of new malware is rapidly increasing; over 140 million malwares have been found in 2015. For countermeasures against malwares, it is important to clarify the behavior of malwares by dynamic analysis. However, dynamic analysis systems so far are mainly for Windows malwares and those for Linux malwares have not been much studied yet. In this paper, we build a dynamic analysis system using TOMOYO Linux as a sacrificial host running on the virtual environment. Using the system, we found the process behavior analyzer of TOMOYO Linux can easily observe the behaviors of malwares and malware behaviors spreading over multiple processes can be easily tracker as well.

Keywords: MWS, Process Behavior, Malware, Dynamic Analysis, Secure OS, TOMOYO Linux, Access Control

1. はじめに

1.1 研究の背景と課題

近年, 新種のマルウェアが急激に増えており, AVTEST[1]によると 2015 年の 1 年間だけで 1 億 4000 万種を超える新種が発見されている．マルウェアがターゲットとする対象も拡大しており, 今までは Windows を対象としたマルウェアが主流であったが，サーバや組み込み機器に広く使用されている Linux を対象としたマルウェアも急増している．これは，Linux サーバやルータの中には固定 IP アドレスが振られ定常稼働している場合も多く，攻撃用のインフラとして利用価値が高いことが要因として挙げられる．また，Linux は Windows に比べてマルウェア感染が一般的でなかったことから, 相対的に Windows に比べてセキュリティ対策が低くなっていること，全てのモノがインターネットに繋がる IoT 時代となり，Linux ベースの組み込み機器が急増していることも要因として挙げられる．

マルウェアへの対策を考える上では，その挙動を明らかにすることが重要であり，その方法の一つにマルウェア動的解析がある．マルウェアの動的解析では，解析環境内でマルウェア検体を実際に実行し，通信内容や内部挙動を観測する．代表的なマルウェア動的システムとして，CWSandbox[2]，Anubis[3]，NORMAN Sandbox[4]などが挙げられる．しかし，これらのマルウェア動的解析システムでは, マルウェアを動作させる解析環境は主に Windows を狙うマルウェア向けであり，Linux を狙うマルウェアの動的解析システムについてはこれまでほとんど検討されていなかった．

1.2 関連研究

大月らの研究[5][6]は，OS よりも下位層で動作する仮想計算機モニタ BitVisor 内へ解析のための拡張機能 Alkanet を開発している．Alkanet は，ゲスト OS 上のプロセスやスレッドから発行されるシステムコールをフックし，システムコールの種類と引数に加え，その処理結果の取得を可能としている．これによって，マルウェアの挙動をより詳細

†1 情報セキュリティ大学院大学
Institute of Information Security

に解析可能になり,かつ取得したシステムコール履歴から,さらに具体的なマルウェアの挙動の抽出し，解析レポートの出力を試みた結果について報告している．しかし，動的解析の対象は Windows を狙うマルウェアのみで，Linux を狙うマルウェアは対象としていない.

原田らの研究[7][8]は, 情報セキュリティを担保するための基礎技術であるアクセス制御に関して，アプリケーションの実行状況を考慮可能な新しいアクセス制御方式について提案しており,そのシステムの概念と実現方法を紹介し，それを Linux 上で実装した TOMOYO Linux における評価結果を報告している．従来のアクセス制御は，主体であるアプリケーションとそれがアクセスしようとするファイルなどの客体の組み合わせによってアクセス可否を判断していた．そのためアプリケーションの処理内容及びアクセスを認めることにより情報システムに与える影響を考慮することができなかった．しかしながら，原田らの提案方式では，アプリケーションの実行状況を，システム起動時以降から当該アプリケーションの実行に至るまでの履歴と，アプリケーションのコマンドライン引数や要求発生時の環境変数等の情報から解釈し，これらの情報を条件に用いることによりアクセス可否を判定することができる．また，情報の取得／判定／強制をカーネルで行うので，安全に漏れなくアクセス制御の強制を行うことができる．しかし，マルウェア解析に応用可能かの検証はしていない.

都丸らの研究[9]は, TOMOYO Linux を用いてマルウェアの動的解析を行い，マルウェアのプロセス活動を収集し，リンク付けを絶対パスで行う手法について提案している．また, 提案した手法で TOMOYO Linux においてセキュリティポリシを策定すれば，メールで受信したマルウェアの活動を紐付けることができ，アクセス制御を行うことができる．さらに，マルウェアの活動をさかのぼることで感染原因を特定することができることを示している．しかし，提案のみで実装をしておらず動作の検証はしていない.

1.3 本研究の貢献

本研究では，マルウェア検体を実行する犠牲ホストとしてセキュア OS である TOMOYO Linux を用いた Linux マルウェア動的解析システムを構築し，Linux マルウェアを動的解析した結果を考察した．また，マルウェアの挙動の追跡の単位を，プログラムの最小単位である機械語レベルではなく,抽象度の高いシステムコールレベルとすることで,マルウェアの挙動を的確に把握し，複数プロセスが連携するようなマルウェア（多段型マルウェア）についても抽象度を上げた観点から追跡できることを検証する.

1.4 本稿の構成

本稿では，第１章はじめに以降，第２章で提案手法の概要として，多段型マルウェアを追跡するための要件と TOMOYO Linux の学習機能について説明する．さらに第３章では評価実験の内容と結果を述べる．最後に第４章でまとめと今後の課題について述べる.

2. 提案手法の概要

2.1 多段型マルウェアの追跡

本研究では，マルウェア挙動の追跡単位を，プログラムの最小単位である機械語レベルではなく，抽象度の高いシステムコールレベルとすることで，マルウェアの挙動を的確に把握し，かつ複数プロセスが連携するようなマルウェア（多段型マルウェア）も追跡できる手法を検討する.

プログラムは通常，システムに影響を与えるような処理を実行する際には，その処理を提供する OS 内のプログラムを呼び出す“システムコール”を使う．一般的には，これを追跡すればマルウェアの挙動を観測できる.

しかし近年では，実行中に新たなプロセスを起動したり,他のプロセスに対してコードを書き込むなど一つのプロセスを監視するだけでは挙動を把握しきれないプロセスを超えて拡散するマルウェア（多段型マルウェア）が増加している．本研究は多段型マルウェアであっても追跡を可能とする手法を提案するものである.

2.2 TOMOYO Linux とその学習機能

提案手法は，セキュア OS である TOMOYO Linux の自動学習機能を用いたマルウェアのシステムコールトレースを基礎とする．TOMOYO Linux の自動学習機能とは，システムの起動から終了まで“利用する機能やサービスを実行する”だけで，必要なアクセス許可内容を自動的に収集する機能である.

Linux などの UNIX の流れを汲む OS では, アプリケーションの実行について特徴的な手順により行う．アプリケーションを実行しようとするプロセスが fork()システムコールにより自身の複製となるプロセスを生成し，複製されたプロセスは execve()システムコールを呼び出すことによってその内容がアプリケーションのものに置き換わる．図 1 では，/bin/bash のプロセスは fork()システムコールを発行し，その複製されたプロセスが execve()システムコールを発行することにより，/bin/date が実行されている．各プロ

セスの右側に表示されているのは，それぞれのプロセスの実行履歴である．その内容はプロセスの生成時に親プロセスから引き継がれ，execve()を行う際に実行しようとするアプリケーションのプログラム名を追加することにより更新できることが分かる．

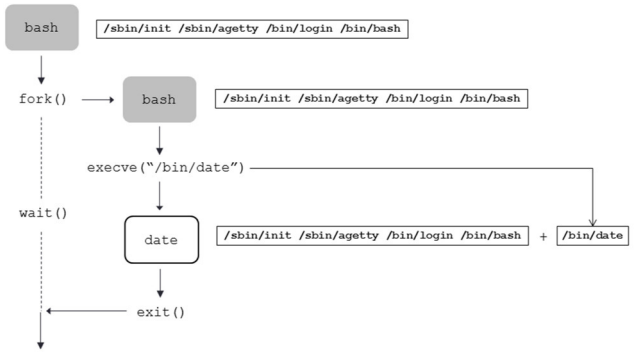


図 1　プログラム実行履歴の定義

Figure 1　Definition program execution history.

TOMOYO Linux では，すべてのプロセスについて，プログラム実行履歴を記憶しておき，その属するドメインごとにアクセス許可を定義し，それに基づきアクセス制御を行う．ドメイン遷移の例を図 2 に示す．

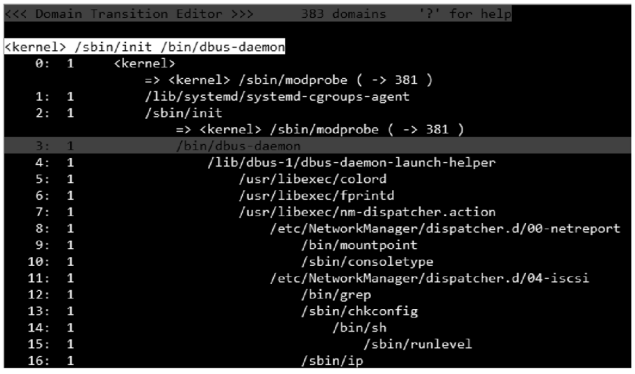


図 2　ドメイン遷移の例

Figure 2　Example of domain transition.

TOMOYO Linux のポリシは，ドメインごとの定義ブロックが複数連続する構造を持ち，各ドメインのブロックは，ドメイン名の宣言で始まり，そのドメインに許可するアクセス許可を必要なだけ列挙したものとなる．ドメインポリシの例を図 3 に示す．

```
1  <kernel> /sbin/init /sbin/agetty /bin/login /bin/bash /usr/bin/passwd
2
3  file read /etc/passwd
4  file read /etc/shadow
5  file write /etc/.pwd.lock
6  file read /dev/urandom
7  file create /etc/nshadow 0666
8  file write /etc/nshadow
9  file chown/chgrp /etc/nshadow 0
10 file chmod /etc/nshadow 00
11 file rename /etc/nshadow /etc/shadow
```

図 3　ドメインポリシの例

Figure 3　Example of domain policy.

TOMOYO Linux には制御モードという概念がある．制御モードは，アクセス制御の動作内容を指定するもので，disable，learning，permissive，enforcing の 4 種がある．各モードの意味について，表 1 に示す．

表 1　TOMOYO Linux の制御モード

Table 1　TOMOYO Linux mode.

モード	内容	動作
disabled	無効	通常のカーネルと同様に動作する．
learning	学習	ポリシ違反が発生しても要求を拒否しない． ポリシ違反が発生しないようにするのに必要なアクセス許可をポリシに追加する．
permissive	確認	ポリシ違反が発生しても要求を拒否しない．
enforcing	強制	ポリシ違反が発生したら要求を拒否する．

モードは，ドメインごとに指定することができ，ポリシファイルで定義するか，あるいはポリシエディタで変更することができる．

learning モードでは，プログラムの実行履歴（ドメイン一覧）と各ドメインのアクセス要求の情報を収集することができる．すべてのプログラムがそれぞれ独立したドメインとなり，基点となる<kernel>から現在に至るまで実行されたすべてのプログラムのパス名を結合したものを示すことができる．この環境を用い，検体を実行し，マルウェアの挙動を把握することを試みた．さらに挙動結果をベースに enforcing モードへ切り替えることでマルウェアのアクセス制御を試みた．

3. 評価実験

3.1 実験内容

VMware Workstation 12 上に CentOS 6.6 をインストールし，ローダブルカーネルモジュールの TOMOYO Linux 1.8.4 カーネルの適用と再コンパイルを行い，learning モードで構築した．また，構築した環境がマルウェアの挙動の追跡に有効かどうかを確認するために，すでに活動が記録されている実際のマルウェアを用いて解析を行った．

実際のマルウェア検体は VirusShare.com[10]より入手した Linux.OSF.8759 (ハッシュ値 SHA1: 1213e130aa4fdfcec29ffcefb4fbf53178a681c3)（以後，この検体を検体１と呼ぶ）及び，Linux.Trojan.Mumblehard.E (ハッシュ値 SHA1: 65a2dc362556b55cf2dbe3a10a2b337541eea4eb)（以後，この検体を検体２と呼ぶ）の２検体で実験を行った．

検体１は ELF 実行プログラムに感染する Linux マルウェアとして知られており[11]，ウイルス本体とバックドア型の全く異なる２つのコンポーネントで構成されている．また，アンチデバック機能を有し，デバッガの下で実行されている場合は，すべてのファイル感染ルーチンをスキップする．カレントディレクトリで 200 ファイルまで感染動作を行い，感染したファイルはサイズが 8759 バイト増加する．検体１に対して，file コマンド及び，readelf コマンドを実行した結果を図 4 に示す．

```
VirusShare_42dfedcbcea03c35587d873e42c67451: ELF 32-bit LSB executable, Intel 8
386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.0
0, not stripped
[root@livecd tmp]# readelf -hl VirusShare_42dfedcbcea03c35587d873e42c67451
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 02 00 e9 48 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Intel 80386
  Version:                           0x1
  Entry point address:               0x8048c49
  Start of program headers:          52 (bytes into file)
  Start of section headers:          15048 (bytes into file)
  Flags:                             0x0
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         6
  Size of section headers:           40 (bytes)
  Number of section headers:         30
  Section header string table index: 27

Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  PHDR           0x000034 0x08048034 0x08048034 0x000c0 0x000c0 R E 0x4
  INTERP         0x0000f4 0x080480f4 0x080480f4 0x00013 0x00013 R   0x1
      [Requesting program interpreter: /lib/ld-linux.so.2]
  LOAD           0x000000 0x08048000 0x08048000 0x01c49 0x01c49 R E 0x1000
  LOAD           0x001c60 0x08049c60 0x08049c60 0x00154 0x00170 RW  0x1000
  DYNAMIC        0x001d14 0x08049d14 0x08049d14 0x000a0 0x000a0 RW  0x4
  NOTE           0x000108 0x08048108 0x08048108 0x00020 0x00020 R   0x4

 Section to Segment mapping:
  Segment Sections...
   00
   01     .interp
   02     .interp .note.ABI-tag .hash .dynsym .dynstr .gnu.version .gnu.version
r .rel.got .rel.bss .rel.plt .init .plt .text .fini .rodata
   03     .data .eh_frame .ctors .dtors .got .dynamic .bss
   04     .dynamic
   05     .note.ABI-tag
```

図 4　検体１の内容

Figure 4　Sample 1 details.

検体２はバックドアとスパム送信機能を兼ね備えた Linux マルウェア“Mumblehard”として知られており[12][13]，Linux や FreeBSD で動作するトロイの木馬型のマルウェアである．リモートでバックドアとして機能するとともにスパムメールを送りつけるという，２つのコンポーネントから構成されている．また，Perl で記述されており，ソースコードを難読化させるために ELF バイナリに圧縮・暗号化されたプログラムが含まれ，アセンブリ言語が用いられている．検体２に対して，file コマンド及び，readelf コマンドを実行した結果を図 5 に示す．

```
VirusShare_b1338cd9b5a853d8920f5a868108135b: ELF 32-bit LSB executable, Intel 80
386, version 1 (FreeBSD), statically linked, corrupted section header size
[root@livecd tmp]# readelf -hl VirusShare_b1338cd9b5a853d8920f5a868108135b
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 09 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - FreeBSD
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Intel 80386
  Version:                           0x1
  Entry point address:               0x804804c
  Start of program headers:          44 (bytes into file)
  Start of section headers:          0 (bytes into file)
  Flags:                             0x0
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         1
  Size of section headers:           0 (bytes)
  Number of section headers:         0
  Section header string table index: 0

Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  LOAD           0x000000 0x08048000 0x08048000 0x07137 0x0754c RWE 0x1000
```

図 5　検体２の内容

Figure 5　Sample 2 details.

3.2 実験結果

検体を learning モードで自由に動作させることで，その活動のために必要なアクセス許可のためのポリシを得ることができる．

検体 1 は TOMOYO Linux の Domain Transition Editor 上より，/usr/bin/newgrp と/usr/bin/passwd を起動していることが分かる．また，検体 1 から起動された/usr/bin/newgrp は/bin/bash を起動，/usr/bin/passwd は/bin/bash を起動していることが分かる．さらに，/usr/bin/newgrp から起動された/bin/bash は検体 1 を起動していることが分かる．図 6 に検体 1 のドメイン遷移結果を示す．

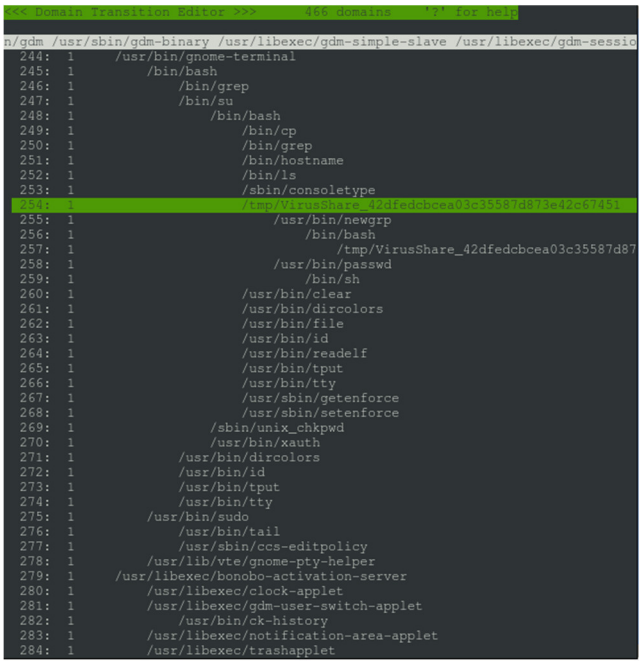
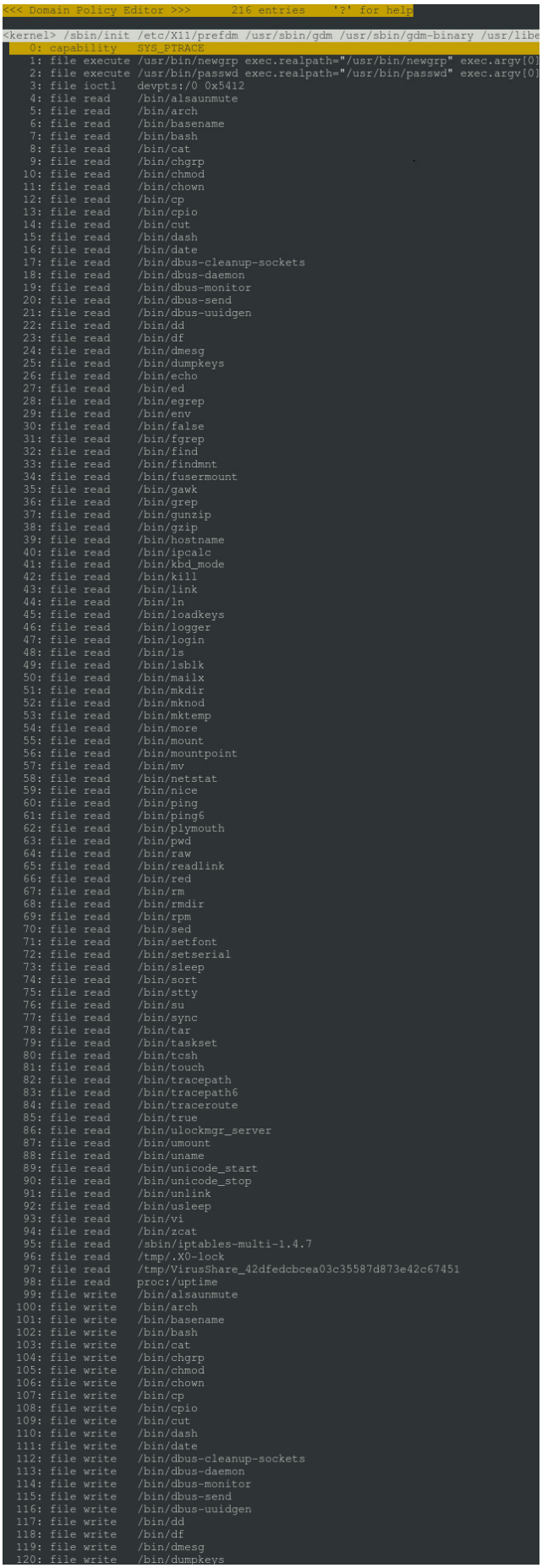


図 6　検体 1 のドメイン遷移

Figure 6　Domain Transition of Sample 1.

次に, TOMOYO Linux の Domain Policy Editor 上からも，検体 1 は/usr/bin/newgrp と/usr/bin/passwd を起動していることが分かる．また，devpts に対して 0x5412 番の ioctl 要求を行っていることが分かる．また，カレントディレクトリ内の 95 個のファイルに対して読み込みモードでオープンを行い, 94 個のファイルに対して書き込みモードでオープンを行っていることが分かる．さらに，21 個の環境変数を参照し，0.0.0.0 のポート 3049 番に対して UDP ソケットがバインドを行っていることが分かる．図 7 に検体 1 のドメインポリシを示す．



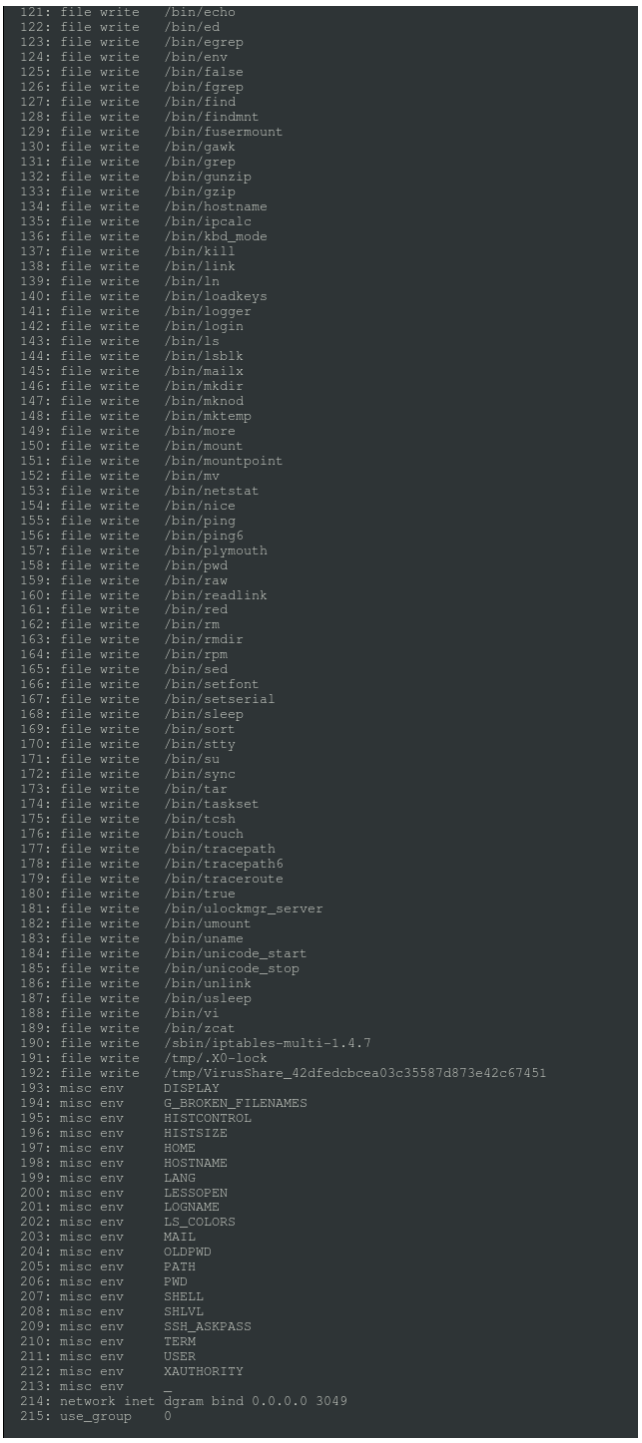


図 7　検体 1 のドメインポリシ

Figure 7　Domain Policy of Sample 1.

また，TOMOYO Linux のアクセス制御機能を learning モードから enforcing モードへ切り替えることで，Operation not permitted エラーとなり，検体 1 の実行を制限できたことを確認した．図 8 に検体 1 の実行を制限した結果を示す．

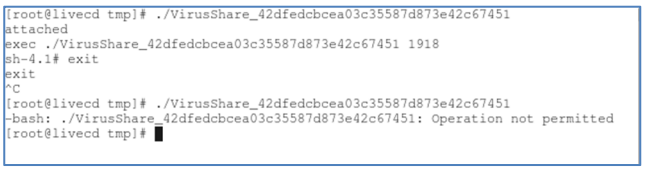


図 8　検体 1 のアクセス制御

Figure 8　Access control of Sample 1.

検体 2 は TOMOYO Linux の Domain Transition Editor より，/usr/bin/perl を起動していることが分かる．図 9 に検体 2 のドメイン遷移結果を示す．

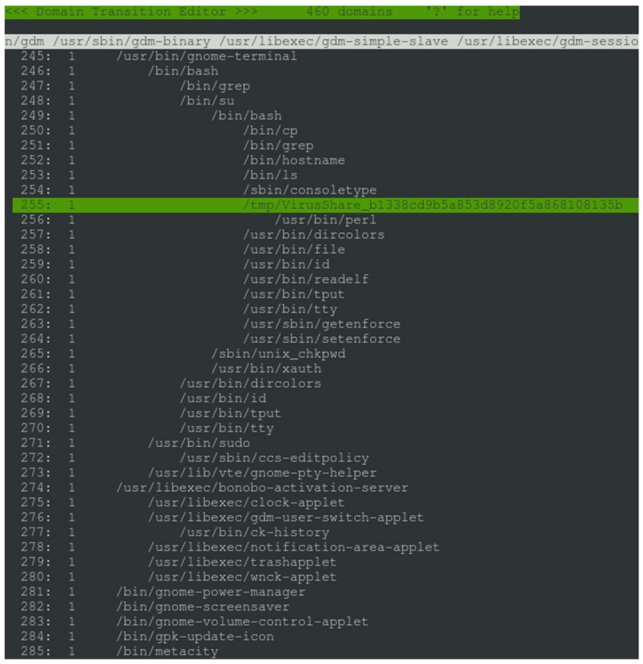


図 9　検体 2 のドメイン遷移

Figure 9　Domain Transition of Sample 2.

次に，TOMOYO Linux の Domain Policy Editor 上からも，検体 2 は/usr/bin/perl を起動していることが分かる．また，ファミリーが 2，タイプが 2，プロトコルが 1 7 のソケットに対して 0x541B 番の ioctl 要求を行っていることが分かる．また，37 個のファイルに対して読み込みモードでオープンを行っていることが分かり，21 個の環境変数を参照していることが分かる．また，192.168.146.2 および 216.239.32.10 のポート 53 番に対して UPD ソケットで送受信していることが分かり，74.125.25.27 のポート 25 番に対して TCP ソケットで接続していることが分かる．図 10 に検体 2 のドメインポリシを示す．

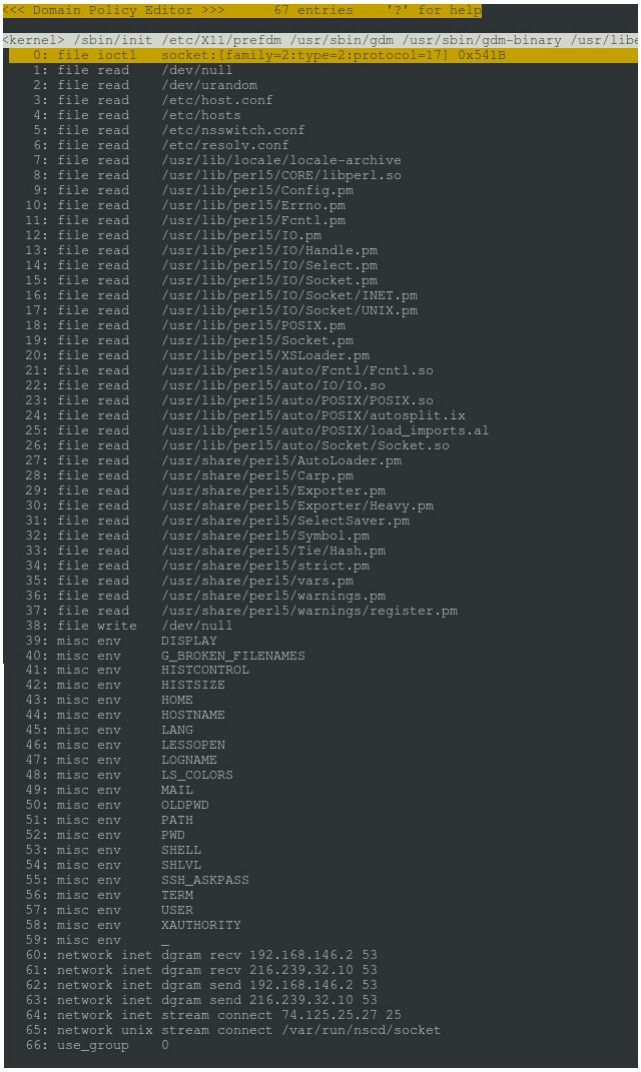


図 10　検体２のドメインポリシ

Figure 10　Domain Policy of Sample 2.

また，TOMOYO Linux のアクセス制御機能を learning モードから enforcing モードへ切り替えることで，Operation not permitted エラーとなり，検体２の実行を制限できたことも確認した．図 11 に検体２の実行を制限した結果を示す．

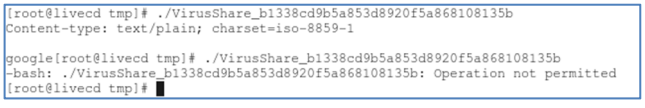


図 11　検体２のアクセス制御

Figure 11　Access control of Sample 2.

検体１，検体２の実験結果を整理し，表 2 に示す．

表 2　実験結果の整理

Table 2　Organizing of the experimental results.

検体名	Domain Transition	Domain Policy
Linux.OSF.8759	Linux.OSF.8759 /usr/bin/newgrp /bin/bash Linux.OSF.8759 /usr/bin/passwd /bin/sh	file execute /usr/bin/newgrp file execute /usr/bin/passwd file execute /bin/bash file execute /bin/sh file read /bin/bash file read /bin/su file read proc:/uptime file write /bin/bash file write /bin/su network inet dgram bind 0.0.0.0 3049 など
Linux.Trojan.Mumblehard.E	Linux.Trojan.Mumblehard.E /usr/bin/perl	file execute /usr/bin/perl file read /etc/host.conf file read /etc/hosts file read /etc/nsswitch.conf file read /etc/resolv.conf network inet dgram recv 216.239.32.10 53 network inet dgram send 216.239.32.10 53 network inet stream connect 74.125.25.26 25 network unix stream connect /var/run/nscd/socket など

上記結果より，マルウェアのシステムコールを正確にトレースできたことに加えて，プロセスを超えて拡散したスレッドも区別し，追跡できることを確認した．これは，各プロセスについて，それぞれの実行履歴の内容を参照し区別することができたためである．

また，TOMOYO Linux のアクセス制御機能を learning モードから enforcing モードへ切り替えることで，Operation not permitted エラーとなり，観測したマルウェアの実行を制限できたことも確認した．

4. まとめと今後の課題

4.1 まとめ

本実験より，セキュア OS である TOMOYO Linux の持つプロセスの振る舞い学習機能をマルウェアの挙動追跡に応用することで，マルウェアの挙動を的確に把握し，複数プロセスが連携するようなマルウェア（多段型マルウェア）についても抽象度を上げた視点から追跡できることを確認した．

また，TOMOYO Linux のアクセス制御機能を learning モードから enforcing モードへ切り替えることで，観測したマルウェアの実行を制限できたことも確認した．

4.2 今後の課題

マルウェアにはさまざまな種類があるため，マルウェアの種類によっては，本提案手法で挙動解析できるマルウェアと挙動解析できないマルウェアがあると考えられる．そのため，検体の数を増やして検証を行い，TOMOYO Linux によるマルウェアの動的解析では，マルウェアの挙動のうち，どのような情報が取得でき，どのような情報が取得できないかを明確に規定する必要がある．その上で，TOMOYO Linux では分析できないマルウェアの種類の明確化を行い，分析できないマルウェアに対しての検出判断アルゴリズムの提案を行う必要がある．

また，TOMOYO Linux のシステムコールトレースログをさらに分析し，マルウェアの特徴的な挙動だけを抽出するツールを作成することで，取得したシステムコールトレースログから複雑なマルウェアや注目するマルウェアを抽出して解析を行うことが可能かの検証が必要である．

また，今回使用した TOMOYO Linux 1.8.4 は標準の Linux カーネルに統合されている TOMOYO Linux 2 系とは異なり，SELinux，Smack，AppArmor などと同時に利用することも可能である．そのため，ラベルベースのアクセス制御とパス名ベースのアクセス制御の両方を有効にした場合に，マルウェアに対するアクセス制御にどのように影響するかの検証が必要である．

また，今回は TOMOYO Linux を用いて，Linux マルウェアのみを解析対象としたが，本手法が Windows マルウェアに対しても有効であるかを検証するためには，Windows 向けにツールを開発して検証することが必要である．

参考文献

[1] AVTEST, https://www.av-test.org/（2016.8.10 アクセス）
[2] CWSandbox, http://www.cwsandbox.org/（2016.8.10 アクセス）
[3] Anubis, https://anubis.iseclab.org/（2016.8.10 アクセス）
[4] NORMAN Sandbox, http://www.norman.com/（2016.8.10 アクセス）
[5] 大月勇人，瀧本栄二，樫山武浩，毛利公一「マルウェア挙動解析のためのシステムコール実行結果取得法」,Computer Security Symposium (CSS) 2011
[6] 大月勇人，瀧本栄二，樫山武浩，毛利公一「マルウェア観測のための仮想計算機モニタを用いたシステムコールトレース手法」, 情報処理学会論文誌,55(9),2034-2046 (2014-09-15) , 1882-7764
[7] 原田季栄，半田哲夫，橋本正樹，田中英彦,「アプリケーションの実行状況に基づく強制アクセス制御方式」， 情報処理学会論文誌，Vol.53 No.9 1-18 (2012)
[8] 原田季栄，半田哲夫,「Linux のセキュリティ機能 : 4. ラベルに基づくセキュリティの限界とその補完 TOMOYO Linux の設計思想と試み」,「情報処理」，Vol.51 No.10 pp.1276 - 1283, Oct. 2010.
[9] 都丸裕大，橋本正樹，田中英彦,「侵入防御のためのプロセス活動リンク付方式に向けた初期的検討」, 研究報告セキュリティ心理学とトラスト（SPT）,2016-SPT-17(15),1-6 (2016-02-25) , 2188-8671
[10] VirusShare.com, https://virusshare.com/（2016.8.10 アクセス）
[11] 日本 CA 株式会社 テクニカルレポート, http://www.caj.co.jp/virusinfo/2002/elf_osf_8759.htm/（2016.6.14 アクセス）
[12] ESET Unboxing Linux/Mumblehard, http://www.welivesecurity.com/wp-content/uploads/2015/04/mumblehard.pdf （2016.8.10 アクセス）
[13] キャノン IT ソリューションズ株式会社 ESET SPECIAL SITE, https://eset-info.canon-its.jp/malware_info/trend/detail/150709_3.html （2016.8.10 アクセス）