Docker を用いたリソース偽装型攻撃観測システムの提案

牛島 遼 [1,a)]　加藤 雅彦 [1]

概要：現在，高度化するサイバー攻撃において，企業や組織を狙った標的型攻撃は増加の一途を辿っている．標的型攻撃は，攻撃手法が巧妙であるため，その被害の全貌の把握が困難である．その攻撃手法を明らかにするためには，企業や組織を模倣したネットワークシステムを構築し，より多くの攻撃情報を観測することが必要である．しかし，そのような観測システムを構築することは容易ではない．そこで本研究では，標的型攻撃の攻撃手法を効率良く観測するために，コンテナを利用した低リソース攻撃観測システムを提案する．本システムは，環境構築を自動化し，かつ，攻撃者に観測環境であることを気づかせないためのリソース情報偽装を行い，より効果的に攻撃情報を観測することを目指す．

キーワード：標的型攻撃，Docker，リソース偽装，マルウェア，動的解析

A Proposal for Resource Spoofing Attack Observation System Using Docker

Ryo Ushijima[1,a)]　Masahiko Kato[1]

Abstract: In recent years, the number of targeted attacks on companies and organizations has been increasing. It is difficult to understand the full extent of the damage caused by targeted attacks because of the sophistication of the attack method. In order to clarify the attack method, it is necessary to construct a network system that mimics a company or organization and observe more attack information. However, it is not easy to construct such an observation system. In this paper, we propose a container-based low-resource attack observation system for the efficient observation of targeted attacks. Our system aims to automate the construction of the environment and more effectively disguise resource information so that attackers are not aware that they are in the observation environment.

Keywords: Targeted Attack, Docker, Resource Spoofing, Malware, Dynamic Analysis

1. はじめに

現在，高度化するサイバー攻撃において，企業や組織を狙った標的型攻撃は増加の一途を辿っている．警察庁のサイバーインテリジェンス情報共有ネットワークによると，標的型メール攻撃は年々増加しており，平成 30 年上半期は 2578 件であったのに対して，令和元年上半期では 2687 件と，前年同期と比べて増加している [1]．IPA によると，標的型攻撃には，以下の 5 段階が存在するといわれている [2].

- i　攻撃のターゲットとなる企業や組織を攻撃するための情報を収集する事前調査段階
- ii　標的型攻撃メールや, 特定の Web サイト閲覧によるマルウェア感染を狙った初期潜入段階
- iii　マルウェア感染した PC 内でバックドアを作成し，外部の C&C サーバと通信を行い，新たなマルウェアをダウンロードする攻撃基盤構築段階
- iv　機密情報の存在箇所の特定や情報の取得を行い，その情報を元に新たな攻撃を仕掛けるシステム調査段階
- v　攻撃専用のマルウェアをダウンロードして，攻撃を遂行する攻撃最終目的の遂行段階

しかしながら，標的型攻撃は，攻撃対象が企業や組織の内部システムであるため，攻撃の詳細な情報が明らかになることは稀である．2015 年に厚生労働省管轄の日本年金機構が標的型攻撃により，約 125 万件の個人情報が流出した事件の詳細が公開されている．この事例では，攻撃に使用されたと思われる不審なメールが 4 種類存在し，それぞれが業務に関連があるようなメールに偽装していた [3]．このように標的型攻撃は，攻撃手法が巧妙であり，かつ，特定組織のシステムが攻撃対象となるため，被害の全貌の把握や攻撃手法の解明が困難である．攻撃手法を明らかにするためには，企業や組織を模倣したネットワークシステムを構築し，攻撃者をおびき寄せ，攻撃の過程を観測する方法が考えられる．攻撃の観測環境として利用されるハニーポットやサンドボックスは, 単一の仮想マシン, もしくは, 物理マシンを用意して攻撃手法の観測を行っている．そのため，この観測手法では，単一ホストへの影響を図ることしかできない．ネットワークシステム全体を模倣すること

1　長崎県立大学 University of Nagasaki
a)　bs217005@sun.ac.jp

© 2020 Information Processing Society of Japan

で，単一ホストだけでなく，ネットワーク上での影響についても分析を行うことが可能となる．一方で，個人がそのような大規模なネットワークシステム全体を模倣した環境を構築するのは，環境構築の労力的に容易ではなく，コスト的な観点からも困難である．そのため，標的型攻撃の観測や解析を行う研究も限られる．

そこで，本研究では，より多くの研究者が標的型攻撃の観測や解析を行えるように，企業や組織のネットワークシステムを再現しつつ，コンテナを用い，低リソースで観測を行うことができる手法を提案する．

一方で，前述の IPA レポート iv にあるように，攻撃者がシステム内部を調査すると，偽装環境であることが容易に判明してしまい，観測環境であることを察知され，攻撃が中止される可能性が考えられる．より長い期間攻撃者を観測環境に滞在させるためには，偽装環境であることが察知されないようにすることが望ましい．そこで，偽装環境を使用した場合でも，できる限り実環境に近く見えるように CPU やメモリにおけるリソース情報の偽装を行い，攻撃の過程を観測しやすくする．2 章で関連研究を紹介し，3 章で提案手法についての説明を行う．4 章で実装方法についての詳細を述べ，5 章で評価を行う．6 章で本提案手法についての考察を行い，7 章でまとめる．

2. 関連研究

攻撃観測システムとして様々な手法が提案されている．代表的なものとして，前述したハニーポットがある．ハニーポットには，主に，低対話型・高対話型・仮想ハニーポットが存在し，それぞれに大きな特徴がある．

低対話型ハニーポットは，特定の OS やアプリケーションを模倣して，監視を行う．模倣している機能に限定した監視になるため，下記の高対話型ハニーポットと比較した際に，得られる情報量が少ない．代表的なものとして，Honeyd [4]，GHH ("Google Hack" Honeypot) [5]，Kippo [6]などがある．高対話型ハニーポットは，実際に脆弱性を残した OS やアプリケーションなどをハニーポットとして利用しているため，高度な観測ができるが，侵入された際に攻撃の踏み台として利用される等のリスクが存在する．仮想ハニーポットは，仮想環境を用いることで，ホストを侵入前の状態に戻すことが容易であるなど，高対話型ハニーポットにあるようなリスクを抑え，管理面で有効である．しかし，マルウェアによっては，仮想環境の特徴を検知するため，ハニーポットであることが侵入者に察知される可能性がある．また，サンドボックスを使った攻撃観測も行われている．サンドボックスはハニーポット同様，被攻撃環境として，実際にマルウェアを動作させて，監視を行う．代表的なものとして，Cuckoo Sandbox [7], Norman Sandbox [8], CW Sandbox [9]などがある．NICT が 2017 年に大規模攻撃観測システムとして STARDUST を提案している [10]．STARDUST は，実ネットワークに高精細な並行ネットワークを構築し，攻撃者に実・並行ネットワークの判別を困難にさせ，攻撃者の行動を観測している．

3. 提案手法

3.1 コンテナを使った攻撃観測システム

1 章で述べたように，低リソースで攻撃観測を行うために，コンテナを使用した攻撃観測システムを提案する（**図 1**）．提案システムでは，攻撃対象となるコンテナ群をネットワーク上に配置する．コンテナ群には，DNS, Web, Mail, Proxy サーバが配置される DMZ ネットワーク及び， File サーバと 100 台規模のクライアントが配置される社内ネットワークが存在する．それぞれのネットワークを Router コンテナで接続する．これらの通信は，Capture コンテナでパケットキャプチャを行い，Log Analysis System コンテナでログ分析を行う．

図 1 攻撃観測システム

Figure 1 Attack Observation System Using Container

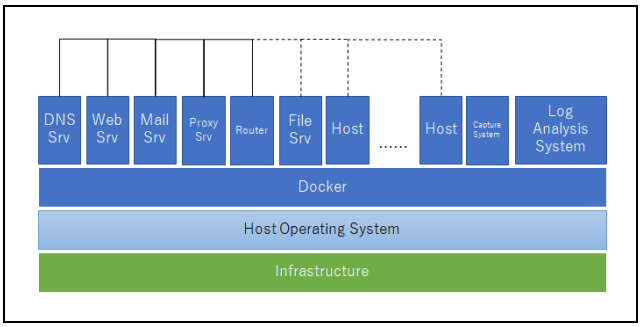


図 1 における実線と点線は，2 つのネットワークが存在する例を示しており，2 つのネットワークは Router コンテナを通して接続する．Capture コンテナと Log Analysis System コンテナは，ホスト OS からのみ接続可能とし，攻撃対象となるコンテナ群からは接続することができない．

また，提案システムは，起動に必要な構成ファイルを用意することで， 1 つのコマンドによって起動及び，停止を行う．これにより，手間をかけることなく，容易に環境が構築可能となる．

3.2 リソース情報の偽装

大規模なネットワークを構築するために，コンテナを複数用意する．各コンテナのリソースが少なく，攻撃者に観測システムであることが察知されてしまうため，システム上のコンテナに対して，リソース情報の偽装を行う．偽装を行う為に，偽装したいリソース情報を持つコンピュータから事前にリソース情報を入手する．リソース情報を入手後，攻撃観測システム上コンテナの /proc/cpuinfo,

/proc/meminfo を volumes mount で偽装する．

4. 実装

ホスト OS に Windows10 Home を使用し，VM 上に提案システムを実装する．

4.1 実装環境

Virtual Machine 上で Docker [11]，Docker-Compose [12]を用いて実装する．実装を行うにあたり，**表 1**，**表 2** の環境を用いる．

表 1 実装環境

Table 1 Implementation Environment

OS	Windows 10 Home
CPU	Intel® Core™ i9-9880H 2.30GHz ×2
RAM	32.0GB
DISK	1TB
仮想環境	VMware Workstation Player 15.56 build-16341506

表 2 コンテナ環境

Table 2 Docker Environment

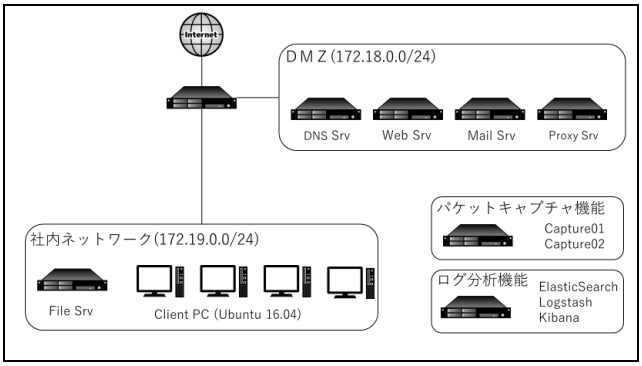
OS	Ubuntu 16.04.6 LTS (Xenial Xerus)
CPU	2 Core
RAM	20.0GB
DISK	256.0GB
Docker	v19.03.12
Docker-Compose	v1.26.2

4.2 コンテナ

攻撃観測システムにおけるネットワーク構成図を**図 2** に示す．

図 2 ネットワーク構成図

Figure 2 Network Configuration



攻撃観測システムにおけるマシン情報は，**表 3** の通りである．サーバ用に CentOS:7.8.2003 コンテナ，クライアント用に Ubuntu:16.04 コンテナを用意する．ログ分析システムとして，Elasticsearch [13] , Logstash [14]， Kibana [15]の ELK Stack を採用した．

DMZ のパケットをキャプチャする Captute01 コンテナ，社内ネットワークのパケットをキャプチャする Capture02 コンテナについては tcpdump で攻撃観測システム内のパケットをキャプチャしているが，Docker の Host Network [16] に接続されているため，攻撃観測システム内において IP Address は割り振られていない．

表 3 マシン情報

Table 3 Machines Information

OS	Application	IP Address
CentOS 7.8.2003	BIND 9.11.4	172.18.0.10
CentOS 7.8.2003	Apache 2.4.6	172.18.0.11
CentOS 7.8.2003	Postfix 2.10.1	172.18.0.12
CentOS 7.8.2003	Squid 3.5.20	172.18.0.13
CentOS 7.8.2003	Router	172.18.0.2 172.19.0.2
CentOS 7.8.2003	Samba	172.19.0.5
Ubuntu 16.04	Client 100 Units	172.19.0.10 ~ 172.19.0.110
CentOS 7	Elasticsearch7.0.0	172.20.0.10
CentOS 7	Logstash7.0.0	172.20.0.11
CentOS 7	Kibana7.0.0	172.20.0.12
CentOS 7.8.2003	Capture01	None
CentOS 7.8.2003	Capture02	None

クライアント及び，サーバ用のコンテナには，それぞれ**表 4** のユーザを追加する．

表 4 ユーザ・パスワード（Client & Server）

Table 4 User / Password (Client & Server)

ユーザ名	パスワード
root	toor
admin	password

表 5，**表 6** のソフトウェアをクライアント及び，サーバのコンテナで共通してインストールしている．ソフトウェアを共通にインストールしたコンテナを用意することで，BIND などのアプリケーションがインストールされたコンテナのビルド時間を短縮する．クライアント及び，サーバコンテナにおいて，router コンテナにポートフォワーディングを行う為に net-tools をインストールしている．

表 5 インストールソフト一覧（Client）

Table 5 Installed Software List (Client)

ソフトウェア名	
ssh	build-essential
evince	net-tools
vim	tcpdump
emacs	traceroute
libreoffice	dnsutils
default-jdk	

表 6 インストールソフト一覧（Server）

Table 6 Installed Software List (Server)

ソフトウェア名	
ssh	traceroute
net-tools	bind-utils
tcpdump	iproute

図 3，**図 4** が，クライアント及び，サーバのコンテナ構築ファイル（Dockerfile）である．

FROM は, 元のコンテナイメージを示しており, RUN は, 実行されるコマンドを示している．また，クライアントコンテナでは，ユーザディレクトリの作成も行っている．

図 3 コンテナ構築ファイル（Client）

Figure 3 Container Configuration File (Client)

```
FROM ubuntu:16.04
RUN apt update -y
RUN apt upgrade -y
RUN apt dist-upgrade -y
RUN apt autoremove -y
RUN apt autoclean -y
RUN apt install -y libreoffice* evince default-jdk build-essential vim emacs
RUN apt install -y net-tools tcpdump traceroute dnsutils ssh
RUN echo root:toor | chpasswd
RUN useradd -m admin
RUN echo admin:password | chpasswd
RUN mkdir /home/admin/Desktop
RUN mkdir /home/admin/Documents
RUN mkdir /home/admin/Downloads
RUN mkdir /home/admin/Music
RUN mkdir /home/admin/Pictures
RUN mkdir /home/admin/Public
RUN mkdir /home/admin/Templates
RUN mkdir /home/admin/Video
COPY ./conf/ssh_config /etc/ssh/ssh_config
```

図 4 コンテナ構築ファイル（Server）

Figure 4 Container Configuration File (Server)

```
FROM centos:7.8.2003
RUN echo "toor" | passwd --stdin root
RUN useradd -m admin
RUN echo "password" | passwd --stdin admin
RUN yum -y update
RUN yum -y install net-tools tcpdump traceroute bind-utils iproute openssh-server
COPY ./conf/sshd_config /etc/ssh/sshd_config
RUN systemctl enable sshd
```

4.3 リソース偽装

リソース偽装として，/proc 以下の cpuinfo, meminfo に Docker-Compose の volumes mount を用いて，偽装を行う．事前に偽装するための fakefile を作成する．fakefile を作成するために，別のコンピュータから事前に，/proc/cpuinfo, /proc/meminfo のリソース情報を取得する．**図 5**，**図 6** が取得した cpuinfo, meminfo の一例である．

図 5 偽装用 cpuinfo の一部（例）

Figure 5 A Part of cpuinfo for Spoofing (Example)

```
processor    : 0
vendor_id    : GenuineIntel
cpu family   : 6
model        : 79
model name   : Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz
……
```

図 6 偽装用 meminfo の一部（例）

Figure 6 A Part of meminfo for Spoofing (Example)

```
MemTotal:        16402220 kB
MemFree:          2324108 kB
MemAvailable:     8059596 kB
Buffers:           499580 kB
Cached:           5060708 kB
……
```

リソース偽装のための volumes mount はコンテナの設定を記述するファイルの volumes タブで行う．**図 7** は，コンテナ構成ファイルである docker-compose.yml におけるソースコードの一部である．ソースコードにあるように，volumes:タブで, 用意した fakefile をコンテナの/proc/cpuinfo, /proc/meminfo にそれぞれ mount している．

図 7 docker-compose.yml の一部

Figure 7 A Part of docker-compose.yml

```
host001:
    build: ./host
    image: ubuntu:host
    hostname: "host001"
    networks:
      eth1:
        ipv4_address: 172.19.0.10
    tty: true
    privileged: true
    command:
      - /sbin/init
    dns:
      - 172.18.0.10
    volumes:
      - cgroup:/sys/fs/cgroup
      - ./data/fake/cpuinfo_h:/proc/cpuinfo
      - ./data/fake/meminfo_h:/proc/meminfo
    environment:
      - HTTP_PROXY=http://proxy.sun-sec.home:3128
      - HTTPS_PROXY=http://proxy.sun-sec.home:3128
      - NO_PROXY=127.0.0.1,localhost
```

5. 評価

提案システムの有効性を評価するため，コンテナの構築時間，リソース消費量及び，リソース情報が偽装されていることを確認する．さらに，実際の攻撃を想定して，マルウェアを動作させ，パケットキャプチャ機能及び，ログ分析機能の評価を行う．

5.1 コンテナ環境評価

コンテナイメージ作成後の攻撃観測システムのプロビジョニングに要した時間を計測する．また，コンテナのホストに対する CPU，メモリのリソース消費量及び，リソース情報の偽装について評価を行う．

5.1.1 プロビジョニング時間

表 7 に観測システム全体のプロビジョニングに要した時間の平均をクライアントコンテナの台数ごとにまとめる．コンテナイメージ構築後の攻撃観測システム起動コマンド”docker-compose up -d”の完了に要した時間を time コマンドで計測する．

表 7 プロビジョニングに要した平均時間

Table 7 Average Time taken for Provisioning

プロビジョニングに要した平均時間	
10 Units	0m15.3784s
50 Units	1m43.8358s
100 Units	2m34.608s

100Units の構築が 3 分以下で完了しており，物理機器を使い, 同環境を構築する場合よりも効率が良いと考えられる．

5.1.2 リソース消費量評価

表 8 にシステム稼働時のホストに対するリソース使用率をまとめる．それぞれクライアントコンテナの台数ごとに top コマンドで計測する．

表 8 top コマンド実行結果

Table 8 Result of Executing the top Command

Container 数	CPU 使用率	RAM 使用率
10 Units	42.4%	19.5%
50 Units	43.9%	23.3%
100 Units	46.1%	25.7%

いずれの台数においても，ホストに対する CPU 及び，メモリのリソース消費量が構築したシステム規模と比べて，少ないことが分かる．

5.1.3 リソース情報の偽装

リソース情報の偽装として，**図 8**～**図 13** に結果を示す．それぞれ偽装前・偽装後に表示される cat /proc/cpuinfo, cat /proc/meminfo のコマンド実行結果の一部を抜粋したものである．

図 8 偽装前 cpuinfo

Figure 8 Before Spoofing cpuinfo

```
model       : 158
model name      : Intel(R) Core(TM) i9-9880H CPU @ 2.30GHz
cpu MHz         : 2304.000
……
```

図 9 偽装後 cpuinfo (Server)

Figure 9 After Spoofing cpuinfo (Server)

```
model       : 79
model name      : Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz
cpu MHz         : 2194.711
……
```

図 10 偽装後 cpuinfo (Host)

Figure 10 After Spoofing cpuinfo (Host)

```
model       : 126
```

```
model name        : Intel(R) Core(TM) i7-1065G7 CPU
@ 1.30GHz
cpu MHz           : 1497.588
……
```

図 11 偽装前 meminfo

Figure 11 Before Spoofing meminfo

```
MemTotal:        20530984 kB
MemFree:          15267152 kB
MemAvailable:    17547116 kB
Buffers:           332544 kB
Cached:           2235472 kB
……
```

図 12 偽装後 meminfo (Server)

Figure 12 After Spoofing meminfo (Server)

```
MemTotal:        16402220 kB
MemFree:          225776 kB
MemAvailable:     2664040 kB
Buffers:           307932 kB
Cached:           2058264 kB
……
```

図 13 偽装後 meminfo (Client)

Figure 13 After Spoofing meminfo (Client)

```
MemTotal:        2045560 kB
MemFree:          72300 kB
MemAvailable:     112984 kB
Buffers:         3412 kB
Cached:           156856 kB
……
```

このように，それぞれの偽装前後を比較すると，リソース情報を偽装していることが分かる．

5.2 マルウェア動作評価

リソース情報を偽装した状態で，攻撃コードが実行されることを確認するため，**表 9** に示すマルウェア検体を実行し，攻撃観測システム上での観測を行う．尚，マルウェア検体は，長崎県立大学情報セキュリティ学科の攻撃情報収集ネットワークより入手したものである．

表 9 マルウェア検体一覧

Table 9 List of Malware Samples

マルウェア検体(md5checksum)	
8ea8fb778d597896e021acc529fd9a4d	検体 A
aee3a5bbbf100e52cc7717d11aeaf101	検体 B
67b9b790ef50c08300943c2e50638581	検体 C

マルウェアを動かしていないコンテナと動かしているコンテナのプロセス状態について比較を行い，マルウェアが何らかのプロセスを残し，動作の様子が見られるかを分析する．さらに，キャプチャしたパケットについても同様に分析を行う．**図 14** は，マルウェアを動かしていないコンテナのプロセスである．

図 14 正常なコンテナのプロセス

Figure 14 Normal Container Processes

```
CMD
/sbin/init
/lib/systemd/systemd-journald
/lib/systemd/systemd-timesyncd
/lib/systemd/systemd-logind
/usr/sbin/atd -f
/usr/bin/dbus-daemon --system
/usr/sbin/sshd -D
/sbin/agetty --noclear tty1 li
/bin/bash
ps -ef
```

次に，マルウェアの実行後のプロセス状態，キャプチャしたトラフィックを示す．

5.2.1 検体 A

図 15 は，検体 A 実行後プロセスを”ps -ef”コマンドで，表示した内容の一部である．

図 15 検体 A 実行後プロセス（一部）

Figure 15 A part of post-process (Sample A)

```
CMD
/sbin/init
/lib/systemd/systemd-journald
/lib/systemd/systemd-timesyncd
/lib/systemd/systemd-logind
/usr/sbin/atd -f
/usr/bin/dbus-daemon --system --add
/usr/sbin/sshd -D
./8ea8fb778d597896e021acc529fd9a4d
/usr/bin/pythno
/bin/bash
ps -ef
sh -c /usr/bin/dpkgd/ps -ef
/usr/bin/dpkgd/ps -ef
```

特徴的なプロセスについて下記に示す．

- ./ 8ea8fb778d597896e021acc529fd9a4d

これはマルウェアが実行中であることを示している．

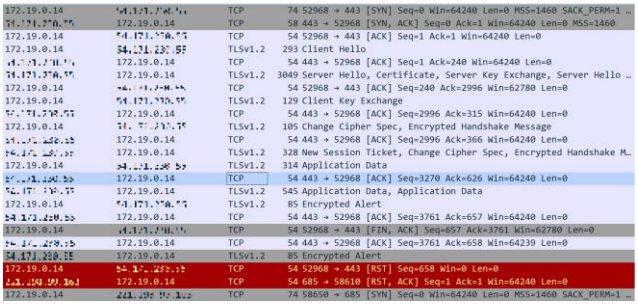
- /usr/bin/pythno

pythno というファイルは md5checksum がマルウェアと同じ 8ea8fb778d597896e021acc529fd9a4d であり，マルウェアが自身をリネームして/usr/bin/以下にコピーを行い，バックドアプログラムとして動作しているものと考えられる．

図 16 に動作中のパケットキャプチャの画面を示す．実際に外部と何らかの通信を行っている様子が見られる．

図 16 キャプチャパケット（検体 A）

Figure 16 Capture Packet (Sample A)



5.2.2 検体 B

図 17 は，検体 B 実行後プロセスを，”ps -ef”コマンドで表示した内容の一部である．

図 17 検体 B 実行後プロセス（一部）

Figure 17 A part of post-process (Sample B)

```
CMD
/sbin/init
/lib/systemd/systemd-journald
/lib/systemd/systemd-timesyncd
./aee3a5bbbf100e52cc7717d11aeaf101
./tmp/svcco -B -o stratum+tcp://pool.supportx
/usr/bin/dbus-daemon --system --address=syste
/lib/systemd/systemd-logind
/usr/sbin/atd -f
/usr/sbin/sshd -D
/sbin/agetty --noclear tty1 linux
/bin/bash
ps -ef
```

特徴的なプロセスについて下記に示す．

・ ./aee3a5bbbf100e52cc7717d11aeaf101

これはマルウェアが実行中であることを示している．

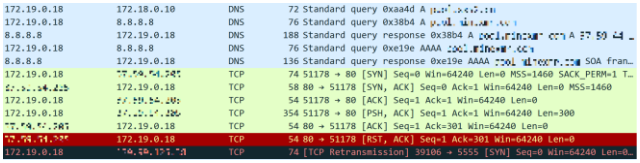
・ ./tmp/svcco

svcco という ファ イル は， md5checksum が 5016c7fd66084cde8897e1d262cc7ea4 となっており，マルウェアと異なるファイルであることが分かった．実行したマルウェアはダウンローダーであった可能性が推測される．なお，svcco のコマンド引数で指定されていたアドレスを Web ページで確認したところ，コインマイニングを行うマルウェアであると思われる．

図 18 に動作中のパケットキャプチャの画面を示す．実際に外部と何らかの通信を行っている様子が見られる．

図 18 キャプチャパケット（検体 B）

Figure 18 Capture Packet (Sample B)



5.2.3 検体 C

図 19 は，検体 C 実行後プロセスを”ps -ef”コマンドで表示させたものの一部である．

図 19 検体 C 実行後プロセス（一部）

Figure 19 A part of post-process (Sample C)

```
CMD
/sbin/init
/lib/systemd/systemd-journald
/lib/systemd/systemd-timesyncd
./67b9b790ef50c08300943c2e50638581
/usr/bin/.sshd
/usr/bin/dbus-daemon --system --add
/usr/sbin/atd -f
/lib/systemd/systemd-logind
/usr/sbin/sshd -D
/bin/bash
ps -ef
sh -c /usr/bin/dpkgd/ps -ef
/usr/bin/dpkgd/ps -ef
```

特徴的なプロセスについて下記に示す．

・ ./67b9b790ef50c08300943c2e50638581

これはマルウェアが実行中であることを示している．

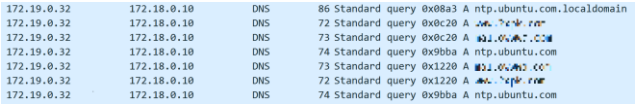
・ /usr/bin/.sshd

.sshd というファイルについては，md5checksum がマルウェアと同じ 67b9b790ef50c08300943c2e50638581 となっており，検体 A と同じくマルウェアが自身をリネームして/usr/bin/以下にコピーを行い，バックドアプログラムとして動作しているものと考えられる．

図 20 に動作中のパケットキャプチャの画面を示す．DNS クエリで C&C サーバと思われるサーバの名前解決を行う通信が見られる．

図 20 キャプチャパケット（検体 C）

Figure 20 Capture Packet (Sample C)

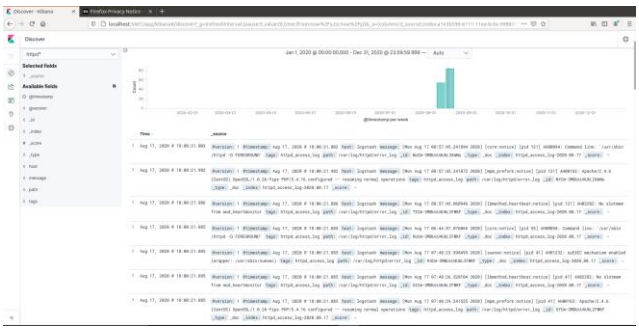


5.3 ログ分析システム動作評価

ログ探索を行っている Kibana の稼働画面を**図 21** に示す．このようにログを集約することで，効率的に攻撃観測を行うことが可能である．

図 21 Kibana 稼働画面

Figure 21 Kibana Screen



6. 考察

6.1 コンテナ

コンテナを使ったシステム環境の構築は，5.1 に示す通り，物理機器で同環境を構築する場合と比べて大幅に短時間で構築可能である．また，ホスト OS 側のリソース消費量も少ないことが分かった．さらに，起動コマンド"docker-compose up -d"で全ての環境を構築できることから構築の容易性が確認できた．一方，次のような課題が存在する．観測終了後にシステム全体を停止させようとした際に，マルウェア実行中の一部のコンテナ停止に失敗した．root 権限でコンテナの停止を行ったが，Permission Denied と表示されることから，コンテナとホストの間に何らかの問題が発生していることが推測される．加えて，コンテナで BIND などのソフトウェアを動かすために，cgroup などの Linux 特有ファイルの mount を行うため，Windows で観測システムを動作させることが現時点でできていない．この問題は Linux 側から Docker for Windows を操作できる Windows Subsystem for Linux 2 を用いることで解決が可能と思われる．前述の課題解決のためには，さらなる検証が必要である．

6.2 リソース偽装

5.1.3 で行った評価の通り，偽装前と偽装後で異なる結果を得ることができ，偽装に成功していることが分かった．一方，次のような課題が存在する．この手法では，fakefile の事前作成が必要である．さらに，より本物の環境に近づけるためには複数の fakefile を用意する必要がある．前述の課題を解決するためには，様々なコンピュータのリソース情報を学習することで，fakefile の自動生成を行うことができるシステムの構築などが挙げられる．

7. まとめ

本稿では，Docker を用いたリソース偽装型攻撃観測システムを提案した．実際にシステムを構築し，リソース情報の偽装を行い，低リソースで実装を行うことができた．さらに，低リソースコンテナ内でマルウェアを動作させ，パケットキャプチャ及び，サーバのログ分析を行うことができた．

今後は，攻撃観測結果を集約することによる攻撃観測の利便性の向上，Linux コンテナにおける対応アプリケーションの増加，Linux コンテナだけでなく，Windows コンテナを用いての攻撃観測システム構築を行い，標的型攻撃のより効率的な観測を目指す．

謝辞

本研究は富士通研究所の支援により行われている．

参考文献

[1] 令和元年上半期におけるサイバー空間をめぐる脅威の情勢等について https://www.npa.go.jp/publications/statistics/cybersecurity/data/R01_kami_cyber_jousei.pdf
[2] 情報セキュリティ白書 2019 https://www.ipa.go.jp/files/000079041.pdf
[3] 日本年金機構における個人情報流出事案に関する原因究明調査結果 https://www.nisc.go.jp/active/kihon/pdf/incident_report.pdf
[4] Developments of the Honeyd Virtual Honeypot http://www.honeyd.org/
[5] GHH – The "Google Hack" Honeypot http://ghh.sourceforge.net/
[6] Kippo – The Honeynet Project https://www.honeynet.org/projects/old/kippo/
[7] Cuckoo Sandbox – Automated Malware Analysis https://cuckoosandbox.org/
[8] Norman SandBox Analyzer Product Sheets http://download01.norman.no/product_sheets/eng/SandBox_analyzer.pdf
[9] The Sandbox | Understanding CyberForensics https://cwsandbox.org/
[10] サイバー攻撃誘引基盤 STARDUST CSS 2017 IPSJCSS2017067.pdf
[11] Empowering App Development for Developers | Docker https://www.docker.com
[12] docker/compose: Define and run multi-container applications with Docker https://github.com/docker/compose
[13] Elasticsearch | オフィシャルの分散型検索/分析エンジン | Elastic | Elastic https://www.elastic.co/jp/elasticsearch/
[14] Logstash | データの一元化，変換，保管 | Elastic | Elastic https://www.elastic.co/jp/logstash/
[15] Kibana | データを探索，可視化，分析 | Elastic | Elastic https://www.elastic.co/jp/kibana/
[16] 古賀政純．Docker 実践ガイド 第 2 版．2019．