模倣 DNS によるマルウェア隔離解析の解析能向上

三輪 信介[†]　宮本 大輔[‡]　櫨山 寛章[‡]　井上 大介[†]　門林 雄基[‡,†]

[†]独立行政法人 情報通信研究機構
〒 184-8795 東京都小金井市貫井北町 4-2-1

[‡]国立大学法人 奈良先端科学技術大学院大学
〒 630-0192 生駒市高山町 8916-5

あらまし ウィルス・ワーム・ボットなどのマルウェアの技術は日々進歩しており，その対策のためには，動作の検証・解析を行う必要がある．マルウェアの動作の検証・解析を行うためには，外部への感染や攻撃を避けるために隔離環境を用いることが必須である．しかし，最近のマルウェアには，隔離環境かどうかを判別する技術が導入されている．外部との接続性があるかや仮想環境上で実行されているかなどから隔離環境かどうかを判別し，隔離環境上では自身の実体の隠蔽や動作の停止などを行い，解析を困難にする対策を採っている．そこで，本稿では，マルウェアの隔離解析環境において，マルウェアによるリアリティチェックへの対策の一つとして模倣 DNS サーバを導入し，その効果を検証した．

Improving Isolated Sandbox using Fake DNS Server

Shinsuke MIWA[†]　Daisuke MIYAMOTO[‡]　Hiroaki HAZEYAMA[‡]
Daisuke INOUE[†]　Youki KADOBAYASHI[‡,†]

[†]National Institute of Information and Communications Technology
4-2-1, Nukui-Kita-machi, Koganei 184-8795, Japan

[‡]Nara Institute of Science and Technology
8916-5, Takayama, Ikoma 630-0192, Japan

Abstract Recent viruses, worms, and bots, called malwares, often have anti-analysis functions such as mechanisms that confirm connectivity to certain Internet hosts. Although, to avoid any impacts to/from the Internet, we conclude that analyzing environments should be disconnected from the Internet. We discuss how malwares can be kept alive in an analyzing environment by disabling their anti-analyzing mechanisms on isolated sandboxes. To reconcile these cross-purposes, we designed an isolated sandbox with fake DNS server that called “NINFD”. We implemented a prototype system and conducted an experiment to test the efficiency of our fake DNS server.

1　はじめに

ウィルス・ワーム・ボットなどのマルウェアは日々進化を続けている．これらに対抗するためには，その動作の仕組みを解析し，問題を検証する必要がある．検証や解析を行うためには，解析環境の外部への二次感染や外部からの実験対象とは違う攻撃による影響を避ける目的で隔離環境を用いることが有効である．

近年では，プロセッサやハードウェアの仮想化技術により，容易に隔離環境を構築できるようになってきている．さらに，破壊的な活動をするマルウェアの検証では，実験環境の再構築を頻繁に行う必要があるため，再構築が容易な仮想化技術が広く利用されている．

これに対し，最近のマルウェアの中には，隔

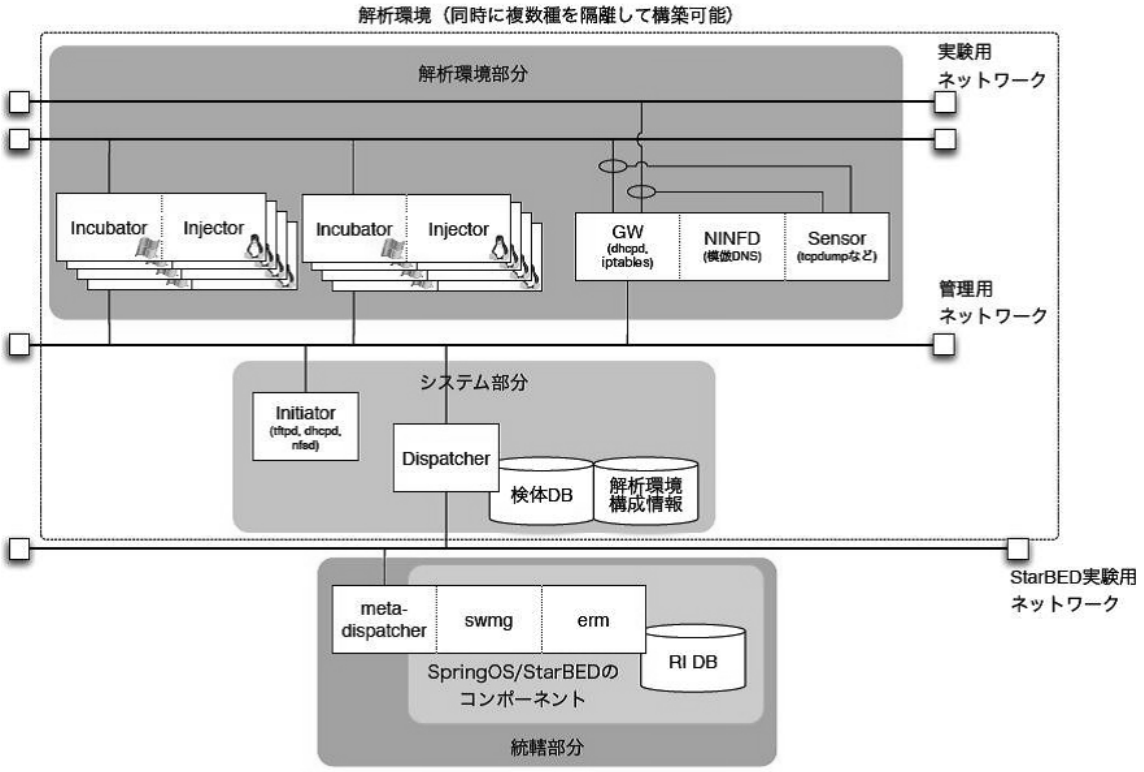

図 1: 実ノードによるマルウェア隔離解析環境の概要

離環境や仮想化技術上で実行されているかを判別し，解析を困難にする機能を持ったものが登場している．このようなマルウェアの検証や解析を行うためには，隔離環境や仮想化技術などを用いることができない．

そこで，本稿では，このようなマルウェアの検証・解析のため，実ノードを用いながら容易な再構築を実現する機能を持った隔離解析環境 [1, 2] に，リアリティチェックの一つである DNS へのクエリを騙すための模倣 DNS サーバを導入し，その効果について，研究用データセット CCC DATAset 2008 のマルウェア検体を用いて検証した結果について述べる．

2 実ノードによるマルウェア隔離解析環境

実ノードによるマルウェア隔離解析環境の個々の解析環境（以降，解析環境）は，解析のたびに，指定された解析環境構成情報に従って，内部の浄化と環境の再構築を行い，指定されたマルウェア検体を検体データベースに記録された検体情報に従い実行する．

提案環境では，複数種の解析環境を同時に構築可能で，それぞれは VLAN によって隔離されている．また，各解析環境はインターネットには接続されていない．図 1 に実ノードによるマルウェア隔離解析環境の概要を示す．なお，基本的な構成はインシデント体験演習環境の演習環境システム [3] と同じである．構成要素と動作について以下に述べる．

2.1 構成要素

提案環境は，解析のために隔離された解析環境部分と解析環境部分を制御するシステム部分，解析環境全体を統轄する統轄部分に分かれる．

なお，提案環境は StarBED[4] 上に構築され

ているため，ノードの資源管理，死活管理やVLANなどのスイッチの設定に関しては，StarBEDを利用するための支援ソフトウェア群であるSpringOS[5] の erm と swmg，および専用のスクリプトを利用している．

解析環境部分の基本的な構成は，マルウェア検体を実行したり，実際に攻撃を受けたりするIncubatorと，Incubatorにマルウェア検体や模擬攻撃を仕掛けるInjectorが8台用意され，4台ずつ別の実験用ネットワークに接続されており，それらの実験用ネットワークをつなぐルータであるGW，その上で実行されるリアリティチェックを回避するための模倣DNSサーバ（NINFD），実験用ネットワークを監視することができるSensorからなる．

システム部分は，解析環境の構築，構成変更，浄化，撤収を管理するDispatcherと，IncubatorとInjectorが起動時に利用するDHCPサーバ，TFTPサーバ，NFSサーバであるInitiatorからなる．Dispatcherは，meta-dispatcherからの指示に従って，あらかじめ用意された解析環境構成情報を元に，Incubator/Injectorの死活や隔離の制御や実行するマルウェア検体の管理を行う．

2.2 動作

Dispatcherは，実行するマルウェア検体を指示に従って検体情報データベースから取得し，Injectorに送信し，Incubatorの設定をInjectorに指示する．この際，Incubatorが動いていた場合には，IPMI (Intelligent Platform Management Interface) を用いて強制的に停止させ，マルウェアの活動や攻撃の影響が次の解析に及ばないようにしている．

Injectorは，Incubatorとデュアルブート構成になっており，Dispatcherからの指示に従って必要なOSのディスクイメージでIncubatorを浄化し，必要な場合はIncubatorの起動スクリプトにマルウェア検体の実行命令などを追加する．

GWとNINFD，Sensorは，1台の実ノード上に構築されており，解析開始時に実験用ネットワークのパケットキャプチャとNINFDの起動，解析終了時にパケットキャプチャの結果やNINFDへのアクセスログを解析記録ファイルに記録する．

```
<?xml version="1.0" encoding="utf-8" ?>
<NEED ID="..ID 情報.." version="0.1">
  <node ID=".. ノードの ID.." role=".. ノードの役割..">
    <functions>
      <!-- ノード操作スクリプトの名前と渡すパラメタの指定 -->
      <!-- @cmdname でノード操作スクリプトのコマンド名を指定 -->
      <!-- @param でそのスクリプトに渡すパラメタを指定 -->
      <!-- '!XPath 式!' の形にするとその部分は../parameters/からの
           相対パスから XPath 式に基づいて探索した結果に置換される -->
      <change_boot_default
         cmdname="scripts/change_boot.sh"
         param="!software/default_os! !network/hostname!"/>
    </functions>
    <parameters>
       <network type="isolated">
       ...
         <hostname>sintcla010</hostname>
         <interface name="m0" type="manage" media="FastEthernet"
           macaddr="00:00:..:..:..:.." osname="eth0"
           bindport="mgsw..:../.." ipaddr="172.16.0..." />
       </network>
       <software>
         <default_os ostype="hdd">boot_p5.bin</default_os>
        ...
         <packages>
          ...
           <dhcpserver packagetype="yum">dhcp.i386</dhcpserver>
          ...
         </packages>
       </software>
     </parameters>
   </node>
...
</NEED>
```

図 2: 演習環境構成情報の概略（XML 文書）

2.3 解析環境構成情報

解析環境構成情報には，解析環境の構成要素となる各ノードとそれをDispatcherが操作するためのインタフェイスの定義をXML文書で記してある．Dispatcherは，この情報に従って，解析環境の構築，構成変更，浄化，撤収を行う．図2にその概略を示す．

2.4 検体情報データベース

検体情報データベースには，マルウェア検体の基本的な情報と実体が格納されている．マルウェア検体の情報は，XML文書で記されている．Dispatcherは，指定されたマルウェア検体をこのデータベースから取り出し，Injectorに送信する．図3にその概略を示す．

```
<?xml version="1.0" encoding="utf-8" ?>
<NEDS ID="..ID 情報..">
  <property>
    <db_dir>..DB 実体の PATH..</db_dir>
  </property>
  <specimen ID=".. 検体 ID.." CID=".. コンテンツ ID..">
    <profile>
      <!-- 検体の基本情報 -->
      <name>.. 検体の名前..</name>
      <filename>.. ファイル名..</filename>
      <filesize>.. ファイルサイズ bytes..</filesize>
      <hash type=".. ハッシュ値の種類（md5,sha1,..etc.）..">
.. ハッシュ値..</hash>
      <captor captor_name=".. 捕獲に利用したもの.."
              engine_version=".. バージョン情報.."
              pattern_version=".. バージョン情報.."
              date=".. 捕獲した日時.."/>
    </profile>
    <classifications>
      <!-- 検体を Virus Scanner などにかけて簡易分類した結果のリスト -->
      <classification scanner_name="file" engine_version="4.10"
pattern_version="v 1.4 2003/03/23 04:17:27" date=".. 分類した日時..">
MS-DOS executable PE  for MS Windows (GUI) Intel 80386 32-bit
      </classification>
      <classification scanner_name="ClamAV" engine_version="0.91.1"
pattern_version="main.inc version: 44, daily.cvd version: 4721"
date=".. 分類した日時..">OK</classification>
    </classifications>
    <!-- data 実体の Hex dump か ref, ref は現在 file のみ -->
    <data type=".. 実体の種類（exe,mail,html,..etc.）.."
          ref="file" name=".. 実体のファイル名.."/>
  </specimen>
...
</NEDS>
```

図 3: 検体情報の概略（XML 文書）

3 NINFD — 模倣 DNS サーバ

名前解決などを行うマルウェア検体を活性化させるためには，DNS 応答を行う DNS サーバの設置が必要である．

実際に解析環境の中に実インターネットで稼働している DNS サーバを摸した名前解決環境を構築しようとすると，RootDNS サーバの解析環境内への設置やマルウェア検体が名前解決する可能性のあるすべての IPv4/IPv6 アドレスおよび名前が記入されたゾーンファイルの作成，各 DNS サーバごとの実アドレス設定やルーティング設定など，模倣 DNS トポロジを構築しなければならず，非常に労力が高い．

そこで，著者らは模倣 DNS サーバ（NINFD）の開発を行った．NINFD (NAIST Internet Name Fake Daemon) は非常に単純な仕組みでマルウェア検体からの名前解決に応答する．

- A レコードによる IPv4 アドレスの名前解決に対しては設定ファイルで指定したドメイン名を応答する．設定ファイルに無い IPv4 アドレスに関しては引数で指定したデフォルトのドメイン名を応答する

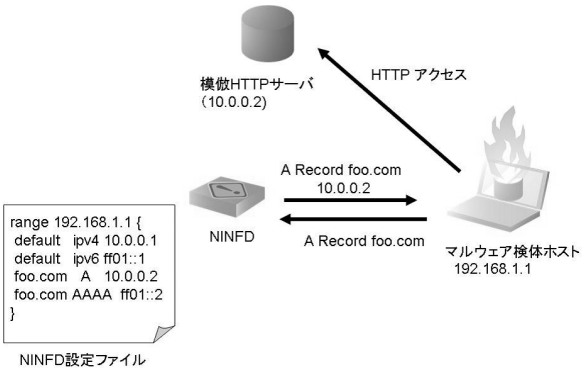

図 4: NINFD の挙動例

- AAAA レコードによる IPv6 アドレスの名前解決に対しては設定ファイルで指定したドメイン名を応答する．設定ファイルに無い IPv6 アドレスに関しては引数で指定したデフォルトのドメイン名を応答する

- A レコードによるドメイン名のアドレス解決に対しては設定ファイルで指定したドメイン名を応答する．設定ファイルに無いドメイン名の場合は引数で指定したデフォルトの IPv4 アドレスを応答する

- AAAA レコードによるドメイン名のアドレス解決に対しては設定ファイルで指定したドメイン名を応答する．設定ファイルに無いドメイン名に関しては引数で指定したデフォルトの IPv6 アドレスを応答する

上記のように，特定のドメイン名，ホスト名に対して個別の設定を行えるため，図.4 に示す NINFD の挙動例のように模倣 HTTP サーバへの誘導など，名前解決後の別のリアリティチェックに対応する模倣サービスサーバに誘導できるように設計している．

4 評価

研究用データセット CCC DATAset 2008 のマルウェア検体を，NINFD なしの解析環境と

NINFD ありの解析環境上で実行し，比較を行った．

4.1 概要

NINFD なしの解析環境と NINFD ありの解析環境でマルウェア検体を 10 分間動作させ，Sensor 上での tcpdump によるパケットキャプチャ，NINFD のアクセスログ，Incubator 上でのメモリダンプを収集した．メモリダンプは，mdd コマンドで取得し，Volatility Framework 1.1.2[6] でその中から検体が展開されている領域のメモリダンプの抽出を行った．

NINFD には，すべてのアドレス関連クエリには，Sensor の IP アドレスを回答し，すべてのドメイン関連のクエリには”starbed.org” を回答する最も単純な設定のみを行った．

なお，今回の提案環境では，NINFD の効果を確認するため，模倣 DNS サーバ以外の他の模倣サーバ（Web サーバやその他のサービスサーバなど）は導入していない．

4.2 解析結果

NINFD なしの解析環境と NINFD ありの解析環境でのパケットキャプチャとメモリダンプについて，比較を行う．

4.2.1 パケットキャプチャ

パケットキャプチャの結果から見てみる．

まず，”rx7.teensmutbox.com” の A レコードの DNS クエリが行われる．実際のインターネット上でのこのクエリへの回答は図. 5 に示す内容となる．

NINFD なしの解析環境では，ICMP メッセージ”udp port domain unreachable” が送信されるため，その後，10 秒間隔で再度同じ DNS クエリが行われる．

これに対し，NINFD ありの解析環境では，Sensor の IP アドレスを回答するため，この回答にもとづいて次の動作に移る．次の動作では，Sensor の IP アドレスに対し，TCP/6667 への接続を試みる．しかし，Sensor では当該ポートが開いていないため，3 秒，6 秒，1 秒の順に間隔を置きながら 90 回接続を試みる．その後，再度 DNS クエリを行い，TCP/6667 への接続を試みるという動作を繰り返す．

```
;; ANSWER SECTION:
rx7.teensmutbox.com.         3581\
IN         A         216.188.26.235
rx7.teensmutbox.com.         3581\
IN         A         216.188.26.237
;; AUTHORITY SECTION:
teensmutbox.com.          172781\
IN         NS         pns2.trellian.com.
teensmutbox.com.          172781\
IN         NS         pns1.trellian.com.
```

図 5: マルウェア検体が最初に牽く DNS クエリへの実環境での回答結果

表 1: メモリダンプの strings 結果

解析環境	文字数	行数	word 数
NINFD なし	58,454	5,684	8,242
NINFD あり	20,412	2,979	3,543

4.2.2 メモリダンプ

次にメモリダンプの結果から，strings コマンドによって文字列を取り出し，見てみる．

NINFD なしの解析環境では，ボットネットへのログインの ID やパスワード，命令情報と思われる文字列，FTP サーバの返答文字列，コマンド名リスト，ホスト名リスト，スクリプトの断片，http エージェント文字列リスト，Debugger 検出時のメッセージ等々，多くの情報を展開された形で見ることができる．

これに対し，NINFD ありの解析環境では，Debugger 検出時のメッセージやいくつかのメッセージは展開された形で見ることができるが，NINFD なしの解析環境で見ることができた情報に比べ，明らかに少ない情報しか見ることができない．単純に比較することはできないが，参考として，メモリダンプの strings 結果の文字数と行数，word 数を表. 1 に示す．

4.3 考察

パケットキャプチャの結果を見る限り，NINFD ありの解析環境がマルウェア検体の名前解決の次の段階に進んでいることが見て取れる．しかし，メモリダンプを見る限り，NINFD なしの解析環境の `strings` の結果の方が情報量が多いことが見て取れる．

このような結果から，NINFD ありの解析環境では，マルウェア検体のリアリティチェックが次の段階に進んだが，その先のチェックを通過できなかったために，動作が変更されたことが類推できる．

このように，模倣 DNS サーバを導入することで，マルウェア検体のリアリティチェックの状態に影響を与えることができ，模倣 DNS サーバなしのときとは別の情報を収集することができるようになるといえる．

5 おわりに

本稿では，マルウェアの隔離解析環境において，マルウェアによるリアリティチェックを騙すために模倣 DNS サーバを導入し，研究用データセット CCC DATAset 2008 のマルウェア検体について，実際に動態解析を行い，その効果を検証した．

今後は，別の検体についても提案環境で動態解析を行い，詳細な評価を行いたい．また，NINFD への主要なサイトの現実に基づいた設定や外部の情報を採り入れた動的な NINFD の設定などにより，より現実に近い模倣 DNS サービスを提供することで，さらなる解析能の向上を目指す．

参考文献

[1] S. MIWA, T. MIYACHI, M. ETO, M. YOSHIZUMI, and Y. SHINODA. Design and Implementation of an Isolated Sandbox with Mimetic Internet used to Analyze Malwares. In *Proceedings of DETER Community Workshop on Cyber Security Experimentation and Test 2007 (DETER07)*, Aug. 2007.

[2] S. MIWA, T. MIYACHI, M. ETO, M. YOSHIZUMI, and Y. SHINODA. Design Issues of an Isolated Sandbox used to Analyze Malwares. In *Proceedings of Second International Workshop on Security(IWSEC2007)*, Oct. 2007.

[3] 三輪信介, 宮本大輔, 櫨山寛章, 樫原茂, 門林雄基, 篠田陽一. インシデント体験演習環境の設計と構築. 情報処理学会, コンピュータセキュリティシンポジウム 2008 (CSS2008), Oct. 2008.

[4] 宮地利幸, 中田潤也, 知念賢一, Razvan Beuran, 三輪信介, 岡田崇, 三角真, 宇多仁, 芳炭将, 丹康雄, 中川晋一, 篠田陽一. StarBED: 大規模ネットワーク実証環境. 情報処理, Vol. 49, No. 1, Jan. 2008.

[5] Y. Shinoda T. Miyachi, K. Chinen. StarBED and SpringOS: Large-scale General Purpose Network Testbed and Supporting Software. In *Proceedings of International Conference on Performance Evaluation Methodlogies and Tools (Valuetools) 2006*, Oct. 2006.

[6] Volatile Systems, LLC. The volatility framework: Volatile memory artifact extraction utility framework. Website, `https://www.volatilesystems.com/default/volatility`. 2006.