情報処理学会論文誌 Vol. 51 No. 9 1633–1644 (Sep. 2010)

マルウェア感染ホスト検出のためのネットワークスキャン手法と検出用シグネチャの自動生成

吉岡克成[†1] 村上洸介[†1] 松本勉[†1]

マルウェアには，外部ホストと通信するために，感染時に特定または任意のポートで待ち受け状態を維持するものが存在する．このようなマルウェアに対しては，スキャンによりポート待ち受け状態や起動中のサービスの応答を調べることでネットワーク越しに感染の有無を推定できる可能性がある．そこで本研究では，マルウェア検体を動的解析する際に，解析環境内の感染ホストのポート待ち受け状態やテスト入力に対する特徴的な応答を調査することで，検出用シグネチャを自動生成する方法を提案する．一般にマルウェアの挙動は確定的でないため，同じ検体を繰り返し解析し同一のテスト入力に対する応答を複数取得したうえで，最長共通部分文字列を検出用パターンとして抽出する．評価実験では，ハニーポットで収集した実マルウェア検体のうち，約 25%〜30%が実行時にポート待ち受け状態となった．また，提案手法により生成したシグネチャを用いて，実行時にポート待ち受け状態となるすべての検体の感染状態が誤りなく推定できた．提案手法は，検査対象ホストに専用の監視ツールをインストールすることなく，遠隔からマルウェア感染を判断できる点で有用といえる．

Network Scan Method and Its Automated Signature Generation for Detecting Malware Infected Hosts

Katsunari Yoshioka,[†1] Kousuke Murakami[†1] and Tsutomu Matsumoto[†1]

Some malware keep listening on a fixed or randomly decided port to communicate with other hosts. Thus, there is a chance to detect a host infected by such malware by remotely scanning its ports and investigating on its responses to some test requests. This paper proposes a novel method to detect a malware infected host by network scan with automatically generated signatures. Our method generates signatures using malware sandbox analysis, in which each malware sample's status of port listening and responses to test requests are closely investigated. In a series of experiments using samples captured in the wild, we confirmed that about 25%–30% of all samples, did start listening on some port and the proposed method can successfully distinguish them from benign network services. Our detection method does not require any dedicated monitoring tool installed on the target hosts for the investigation, which can be a useful advantage.

1. はじめに

重要ファイルやパスワードの流出，盗聴，個人情報の漏洩，迷惑メール，フィッシング，サービス妨害攻撃といった，インターネット上のセキュリティ脅威が大きな問題となっている．これらの脅威において，高度に機能化された悪意のあるソフトウェア，いわゆるマルウェアが果たす役割が大きいことから，マルウェア対策技術の研究開発が活発に進められている．

脅威の構造が近年複雑化し，マルウェアの感染手法や活動内容が多様化していることから，マルウェア対策技術もまた非常に多岐にわたるが，マルウェア感染というイベントを軸に整理すると，感染を未然に防ぐための「予防」，感染時にその事実を発見するための（感染の）「検出」，感染が検出された際に除去や隔離，システム修復，脆弱性の修正を行うなどの「対応」という 3 つの目的が考えられる[⋆1]．予防のための技術が完全であれば，検出，対応は必要ないと極論できるが，実際には予防のための技術は完全ではないため，予防，検出，対応のすべてが重要であることはいうまでもない．

本研究では，上記の目的の中で特に「検出」に注目し，マルウェアに感染したホストをスキャンによりネットワーク越しに検出する手法を提案する．提案手法は，感染時に特定または任意のポートで待ち受け状態を維持しつつ外部ホストと通信を行うマルウェアを検出対象とする．たとえば，Sasser[19] は自身の複製を感染先に転送するためにポート 5554/TCP 上で簡易 FTP サーバを起動することがよく知られているが，近年では，プロキシサーバとして働くマルウェア[17] や，Conficker の亜種[6] のように自身の IP アドレスと時刻情報から一意に定まるポートで待ち受けを行い，感染マシン間で通信を行うマルウェアも存在する．このようなマルウェアに対しては，スキャンによりポート待ち受け状態や起動中のサービスの応答を調べることでネットワーク越しに感染の有無を推定できる可能性がある．そこで提

†1 横浜国立大学
Yokohama National University

⋆1 本論文では，感染前にマルウェアを検出する技術は「予防」の一種と位置づけ，感染の事実を発見することを「検出」と呼ぶこととする．

案手法では，マルウェア検体を動的解析する際に，解析環境内の感染ホストに対してネットワークスキャンを行い，ポート待ち受け状態やテスト入力に対する特徴的な応答を調査することで，検出用シグネチャを自動生成する．一般にマルウェアの挙動は確定的でないため，同じ検体を繰り返し解析し同一のテスト入力に対する応答を複数取得したうえで，最長共通部分文字列を検出用パターンとして用いる．またテスト入力だけでなく，動的解析中に実際に外部ホストからリクエストを受信し，それに対する検体の応答を観測した場合にも同様にシグネチャを生成する．

評価実験では，ハニーポット Nepenthes[3] により 2009 年 6 月から 7 月の約 2 カ月間に収集した実マルウェア検体のうち，約 25%〜30%が実行時にポート待ち受け状態となることが確認された．また，上記の方法で生成したシグネチャにより，ポート待ち受け状態となるすべての検体の感染状態が正しく推定できた．提案手法は，ネットワークベースのマルウェア検出法の中でもマルウェアからの通信を受動的に検出するのではなく，スキャンによる能動的検出を行う点が特徴であり，典型的にはシステム管理者が管理対象ホスト群の監査を行う際に用いることができる．この際，管理対象ホスト群に専用の監視ツールをインストールすることなく，遠隔地から監査を行える点が有用といえる．加えて，ハニーポットに攻撃を仕掛けてきた攻撃ホストの調査などの応用も考えられる．また，提案手法は，マルウェア動的解析技術を具体的なマルウェア対策に結び付ける点で，動的解析の新たな応用例の提案という側面も持つ．

以降の本論文の構成は以下のとおりである．まず 2 章で関連研究を紹介し，3 章で提案手法の説明を行う．4 章では提案手法を実装したシステムによる評価実験について述べ，考察を行う．5 章でまとめと今後の課題について述べる．

2. 関 連 研 究

マルウェア感染を検出する技術は，ホストベースまたはネットワークベースという観点で分類できる．たとえば，広く普及しているエンドユーザ向けのセキュリティソフトやウイルス対策ソフトの中には，保護対象システムの状態を監視し，マルウェアの感染を検出する，ホストベースの検出機能を有するものがある．ホストベースの検出は有効であるが，一般に保護対象システムに専用のツールをインストールする必要があるため，適用が難しい場合がある．一方，侵入検知システムや侵入防止システム，アプリケーションファイヤウォールなどは，マルウェアからの攻撃やマルウェアの侵入を検出・遮断すると同時に攻撃元の特定を行っていることから，ネットワークベースのマルウェア感染ホストの検出技術ととらえられる．また，ハニーポットやインターネット定点観測もマルウェア感染ホストの検出技術ととらえられる．しかしながら，これらの技術は本来，保護・観測対象のネットワークに外部から届く攻撃を観測・遮断したり，内部から発生した攻撃を検知したりすることを目的としているため，感染ホストの検出を行えるのは当該ホストからの攻撃が発生した場合であり，その点で受動的な検出技術といえる．

一方，ネットワークベースで能動的なセキュリティ検査を行う技術として脆弱性スキャン[1]，ファジング[12]，ペネトレーションテスト[2] がある．これらの技術は検査対象の脆弱性の有無を検査することを目的とするが，製品化された脆弱性スキャンツールの中には，マルウェアのバックドア検査を行えるものがある[21]．また，セキュリティ監査ツール Nmap[14] の開発者が作成した Conficker 検出ツール[15] は，Conficker の亜種が P2P 通信に利用するポートの選択アルゴリズムを逆に利用し，検査対象ホストのポート待ち受け状態により感染の有無を判断する．しかしながら，このような能動的なマルウェア検出を行う技術において，検出のためのシグネチャやロジックを自動生成する方法は我々の知る限り提案されていない．

上記の関連研究の状況をふまえて，本研究では，マルウェア検体を動的解析することで，ネットワークベースの能動的マルウェア検出のためのシグネチャを自動生成する手法を検討する．

3. ネットワークスキャンによるマルウェア感染ホストの検出手法の提案

本章では，ネットワークスキャンによるマルウェア感染ホストの検出手法を提案する．提案手法の基本アイディアは，検査対象ホストのポート待ち受け状態およびテスト入力に対する応答を外部からネットワークスキャンにより調べることでマルウェア感染の有無を判断するというものである．そのため，まず実際のマルウェアが感染時にどのようなポート待ち受け状態となり，また，様々なテスト入力に対してどのような応答を行うかを調査し，それらの情報をシグネチャとして蓄積する．特に提案手法では，自動でシグネチャ生成を行うために，マルウェア動的解析を利用する．すなわち，動的解析環境内でマルウェア検体に感染中のホストに対して実際にスキャンを行い，ポート待ち受け状態やテスト入力に対する検体からの応答情報を収集し，シグネチャを自動生成する．このようにして得られたシグネチャを用いて，任意の検査対象ホスト群のマルウェア感染の有無や感染したマルウェアの種別を推定する．3.1 節では，提案手法の基本アイディアを実際のマルウェアの解析例を用いて説明する．3.2 節では，提案手法の実現形態であるマルウェア感染ホスト検出システムについ

て説明する．3.3 節では，システムの実装について説明する．

3.1　基本アイディア

提案手法の基本アイディアは，ネットワークスキャンによって検査対象ホストのポート待ち受け状態やテスト入力に対する応答を観測し，得られる情報をもとにマルウェア感染の有無を推定するというものである．以下では，テスト入力に対する，ある検体（MD5 値：259bd4099f06ab6f153b5b7699034eaf：Symantec Norton による名称：W32/Backdoor：McAfee による名称：New Malware.b）の応答の実例を用いて基本アイディアを説明する．

4 章の評価実験で用いた動的解析環境において上記の検体を実行すると，ポート 113/TCP で待ち受けを行うサービスが起動した．このようにマルウェア検体の実行に起因して起動するサービスを不正サービスと呼ぶこととする．この不正サービスに対して，HTTP などのプロトコルで送信データの終端を意味するデータ”¥x0a¥x0d¥x0a¥x0d” をテスト入力として送信すると，**図 1** の 1 行目のような応答が得られた．さらに，当該検体に対して同様のテスト入力を繰り返し送信すると，図 1 の 2 行目および 3 行目のような応答が得られた．これらの応答は，RFC912，RFC931，RFC1413 で定義されている Authentication Service [11] /Identification Protocol [13]（以下，auth/ident プロトコル）の応答メッセージに似せられているが，上記のテスト入力に対する応答としては異常でありプロトコルに従ったものではない．実際，当該検体は，113/TCP に対するどのような入力に対しても図 1 と類似の文字列を返信する．当該検体が C&C サーバに接続する際，C&C サーバからポート 113/TCP に対して認証要求メッセージが届くが，これに対してマルウェア側から上記の応答を行うことで認証を行っているものと思われる．ただし，このような認証を行わずに C&C サーバとの接続が行われる場合もある．上記の例のように，マルウェア作成者は目的を達成するために最小限のサーバ機能を実装する場合が多い．そこで，この実装の不完全性をとらえることでマルウェアと正規サービスとを区別する．図 1 の例においては，応答の一部はどのような入力に対しても同一であり，おそらくハードコーディングされているため，この文字列を応答パターンとして検出に用いることで，このような異常な応答を識別できる．

3.2　マルウェア感染ホスト検出システム

本節では，3.1 節で説明した基本アイディアの実現形態であるマルウェア感染ホスト検出システム（以降，単に本システムと呼ぶ）について説明する．まず，**図 2** に本システムの全体図を示す．本システムの処理は，検出に用いるシグネチャを生成・更新するために，マルウェア検体の動的解析を行う**シグネチャ生成・更新フェーズ**と．ネットワークスキャンにより任意の検査対象ホスト群のマルウェア感染有無を判別する**マルウェア感染ホスト検出フェーズ**からなる．以降では，システムの構成要素について述べ，それぞれのフェーズの処理について説明する．

```
1回目 : 1603, 6667 : US3R1D : T3CH : iouser184
2回目 : 5920, 6667 : US3R1D : T3CH : iouser158
3回目 : 4723, 6667 : US3R1D : T3CH : iouser155
```

図 1　あるマルウェア検体のポート 113/TCP へのテスト入力に対する応答
Fig. 1　Example of malware's response to a test request on port 113/TCP.

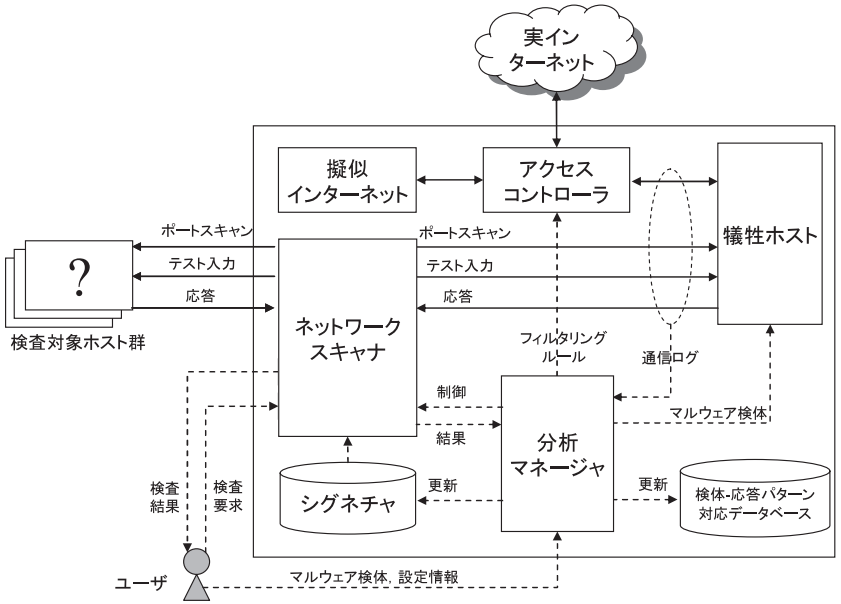

図 2　マルウェア感染ホスト検出システム
Fig. 2　Detection system for malware infected host.

シグネチャ　シグネチャとは，ネットワークスキャナが検査対象ホストのスキャンを行う方法とスキャンに対する応答の照合ルールを記述したデータベースであり，その実体は（テスト入力，応答パターン系列）からなるレコードの系列である．テスト入力とは，待ち受け状態が確認された検査対象ホストのポートに対して送信する要求メッセージを意味する．また

テスト入力1,応答パターン1;ID1,応答パターン2;ID2, 応答パターン3;ID3
テスト入力2,応答パターン4;ID4,応答パターン5;ID5
テスト入力3,応答パターン1;ID6,応答パターン4;ID7, 応答パターン6;ID8
テスト入力4,応答パターン7;ID9
テスト入力5,応答パターン8;ID10
……

図 3　概念的なシグネチャの例
Fig. 3　Example of signatures.

ID1, 検体ハッシュ値1, 検体ハッシュ値3, 検体ハッシュ値4,検体ハッシュ値10
ID2, 検体ハッシュ値5, 検体ハッシュ値9
ID3, 検体ハッシュ値6
ID4, 検体ハッシュ値7
ID5, 検体ハッシュ値2,検体ハッシュ値8,検体ハッシュ値11
……

図 4　概念的な検体–応答パターン対応データベースの例
Fig. 4　Example of correspondence of malware samples to their response patterns.

応答パターンとは，テスト入力に対して検査対象から返ってきた応答と照合するためのパターンである．**図 3** にシグネチャの概念的な例を示す．図 3 では各行がレコードとなっている．なお各応答パターンには ID 番号が付加される．

ネットワークスキャナ　ネットワークスキャナは，検査対象のホストにポートスキャンを行い，待ち受け状態にあるポートを確認したうえで，各待ち受けポートに対してシグネチャに定められた順序でテスト入力を送信する．テスト入力に対する応答があった場合は，シグネチャに定められた応答パターンとの照合を行い，照合結果を出力する．ネットワークスキャナの処理の流れを以下に示す．

(1)　検査対象ホストの全 TCP/UDP ポートに対してスキャンを行い，待ち受けポートを特定する．
(2)　特定された待ち受けポートに対してそれぞれ以下を行う．
　(ア)　シグネチャの各レコードの順に以下を行う．
　　①　検査対象ホストにテスト入力を送信する．
　　②　設定時間 T_w[秒] 以内に検査対象ホストからの応答があった場合は，応答パターン系列との照合を行う．
　　③　応答パターンと一致した場合は，当該応答パターンの ID 番号を出力した後，(2) に戻り次の待ち受けポートの検査に進む．いずれの応答パターンとも一致しなかった場合は，(ア) に戻り次のテスト入力を行う．

検体–応答パターン対応データベース　各応答パターンがどの検体から生成されたものであるかを記録・蓄積するためのデータベースであり，その実体は，(ID 番号，検体ハッシュ値系列）からなるレコードの系列である．ここで ID 番号とは応答パターンの ID 番号である．**図 4** に概念的な例を示す．図 4 では各行がレコードとなっている．

犠牲ホストとアクセスコントローラ　犠牲ホストは，マルウェア動的解析において検体を故意に実行し感染させるための環境である．犠牲ホストはアクセスコントローラを介して擬似インターネットおよび実インターネットに接続している．犠牲ホスト上で実行された検体が外部ホストに対して通信を行った場合は，すべてのパケットはまず，アクセスコントローラに届く．アクセスコントローラは，事前に設定されたフィルタリングルールに従い，当該通信を実インターネットまたは，擬似インターネットに転送する．逆に擬似インターネットや実インターネット側から犠牲ホストに対して届くパケットについては無条件に犠牲ホストに転送する．なお，アクセスコントローラのフィルタリングルールの設定については，当該検体の動的解析を複数回行い，通信量や宛先アドレスの決定方法など複数の基準に基づき危険性の判定を行う，文献 9) の手法の適用を想定している．

擬似インターネット　擬似インターネットは，犠牲ホストに対して擬似的なインターネット環境を提供する．そのため，HTTP，HTTPS，FTP，TFTP，SMTP，NTP，DNS，IRC といったプロトコルに対応した簡易サーバを持っており，犠牲ホストからの要求に従い，ダミー応答を返すように設定されている．また，ハニーポットプログラムである Nepenthes も擬似インターネット内で動作しており，139/TCP，445/TCP などエクスプロイト攻撃の対象となるポートに対するアクセスに対して応答し，脆弱なサービスを模擬することでマルウェア検体からの攻撃を観測することができる．また，疑似インターネット内の Nepenthes に対して犠牲ホストからエクスプロイト攻撃が行われた際は，Nepenthes から犠牲ホストに対してコネクトバックなどの通信が生じる場合があるため，シグネチャ生成時にこの通信をテスト入力として利用できる[7]．擬似インターネットの構成については文献 7)，9) で提案した手法を用いることを想定している．

シグネチャ生成・更新フェーズの流れ　シグネチャ生成・更新フェーズは分析マネージャによって管理される．フェーズの流れは以下のとおりである．

(1)　ユーザはマルウェア検体および解析の設定ファイル（アクセスコントローラのフィル

タリングルール，検体の解析回数，実行時間）を分析マネージャに入力し，解析を開始する．

(2) 指定された解析回数だけ (3)–(7) を繰り返す．

(3) 分析マネージャは，アクセスコントローラにフィルタリングルールを適用し，犠牲ホストを起動する．その後，分析マネージャは犠牲ホストの通信のフルキャプチャを開始する．

(4) 分析マネージャは，マルウェア検体を犠牲ホストに転送し，犠牲ホスト上で実行する．さらに設定時間 T_s[秒] が経過した後にネットワークスキャナを起動し，犠牲ホストに対してスキャンを行う．

(5) スキャンが終了すると，分析マネージャは，パケットキャプチャを停止し，通信ログを得る．また，犠牲ホストを停止し，OS イメージをリフレッシュする．

(6) 分析マネージャは，ネットワークスキャナの結果から検体のポート待ち受け状態を確認し，待ち受け状態のポートに関する TCP/UDP セッションを通信ログから再構築する．ここで各セッションについて通常双方向のストリームデータが得られることに注意する．ここで，再構築された TCP/UDP セッションの集合を S と記述することとし，セッション $s \in$ S に関して，犠牲ホスト向けストリームデータを要求 $\mathrm{Req_s}$ と呼ぶこととし犠牲ホストから外部ホスト向け（またはネットワークスキャナ向け）のストリームデータを応答 $\mathrm{Res_s}$ と呼ぶこととする．

(7) すべてのセッション $s \in$ S に対して以下を行い，シグネチャを更新する．

(ア) **（応答パターンの更新）**要求 $\mathrm{Req_s}$ がシグネチャ内のあるレコード r のテスト入力と完全に一致し，かつ，応答 $\mathrm{Res_s}$ がレコード r の応答パターン p と閾値 $Th_{\min}$ バイト以上一致する場合は，$\mathrm{Res_s}$ と p との最長共通部分文字列をパターン p と入れ替えることでレコードの更新を行う．また，検体–応答パターン対応データベース内の応答パターン p の ID 番号に対応する検体ハッシュ値として解析対象検体のハッシュ値を加える．

(イ) **（応答パターンの追加）**要求 $\mathrm{Req_s}$ がシグネチャ内のあるレコード r のテスト入力と完全に一致し，かつ，応答 $\mathrm{Res_s}$ と閾値 $Th_{\min}$ バイト以上一致する応答パターンが r に 1 つも存在しない場合は，当該テスト入力に対応する応答パターンとして $\mathrm{Res_s}$ を追加する．また，新たな ID 番号を発行し，検体–応答パターン対応データベースにレコードを追加する．

(ウ) **（レコードの追加）**要求 $\mathrm{Req_s}$ がシグネチャ内のあるテスト入力のいずれとも一致しない場合は，$\mathrm{Req_s}$ をテスト入力とし，$\mathrm{Req_s}$ を応答パターンとする新たなレコードをシグネチャに追加する．また，新たな ID 番号を発行し，検体–応答パターン対応データベースにレコードを追加する．

「応答パターンの更新」は，同一のテスト入力に対する応答群から最長共通部分文字列を抜き出し，これを新たな応答パターンとする処理である．これによって，応答群の中のランダム性をなるべく排除し，固定の文字列を検出用パターンとして抜き出せるようにする．また「応答パターンの追加」処理は，テスト入力に対してこれまで観測された応答とは大きく異なる応答が得られた際に発生する．さらに,「レコードの追加」処理は，ネットワークスキャナからではなく，実インターネットまたは擬似インターネット上のホストから犠牲ホストに対して要求があった場合のみ発生する．これは，外部ホストから犠牲ホスト上の検体に対して実際に送られた要求であり，当該検体に対して有効な要求であることが期待されるため，新規のレコードとしてシグネチャに追加し，今後の検出に有効利用する．なお，応答パターンには最大長 $Th_{\max}$ を設定し，$Th_{\max}$ バイトを超える応答が得られた場合には，当該パターンの先頭から $Th_{\max}$ バイトのみを応答パターンとして用いることとした．

マルウェア感染ホスト検出フェーズの流れ　マルウェア感染ホスト検出フェーズでは，シグネチャ生成・更新フェーズにおいて作成したシグネチャを用いて任意の検査対象ホスト群の検査を行う．ユーザがネットワークスキャナに対して検査要求を出すと，ネットワークスキャナはシグネチャ生成・更新フェーズで生成・更新したシグネチャを用いて検査対象ホストに対してスキャンを行い，検査の結果を出力する．提案手法は，正規サービス検出用のシグネチャの生成は行えないが，シグネチャの初期状態として正規サービスのシグネチャを登録しておくことができる．たとえば，4 章の評価実験では，Nmap v5.0 [14] のサービススキャン用のシグネチャを初期状態として用いているため，Nmap に登録済みの正規サービスを検出できる．検査結果は，待ち受け状態の各ポートについて，(1) 正規サービスとして検出，(2) 不正サービスとして検出，(3) 不明なサービスとして検出，のいずれかとなる．

3.3 実　　装

本節では，本システムの実装について説明する．まず**図 5** に全体図を示す．本システムは 1 台の実機上に実装した．犠牲ホストは VMware Server 1.0.6 のゲスト OS（Windows XP Professional SP1）を用いており，ホスト OS は CentOS 5.3 を用いた．犠牲ホスト以外のすべての構成要素はホスト OS 上に実装した．以下，各構成要素の実装について述べる．なお，本システムは完全自動化されており，解析対象の検体セットと初期設定を行うだけで，自動的に検体の解析を行いシグネチャの更新を行う．

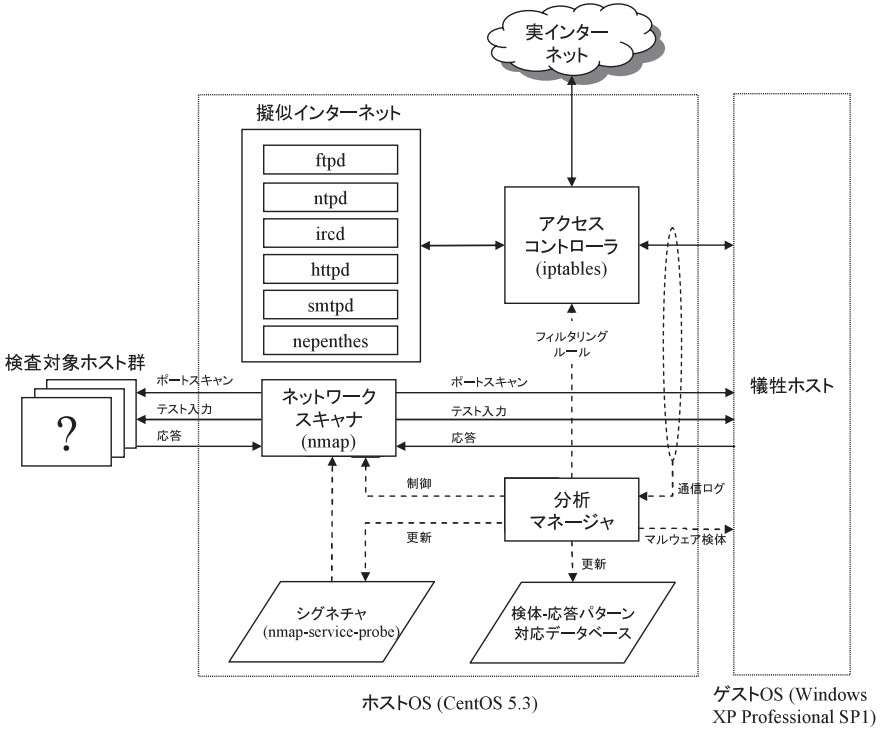

図 5　マルウェア感染ホスト検出システムの実装
Fig. 5　Implementation of detection system for malware infected host.

ネットワークスキャナとシグネチャ　ネットワークスキャナは Nmap v5.0 を用い，シグネチャは，Nmap のサービススキャン設定ファイルである nmap-service-probe のフォーマットに従うようにした．Nmap 自体には変更を加えていないため，当該設定ファイルをシグネチャファイルに入れ替えるだけで，任意のマシンで容易にマルウェア感染ホスト検出が行える．以下，nmap-service-probe ファイルの書式とシグネチャについて説明する．なお，設定ファイルの書式の詳細は文献 14) をご参照いただきたい．nmap-service-probe ファイルは，各行の先頭に命令（Directive）があり，各命令によりスキャンの方法を記述する形式になっている．以下に本システムのシグネチャと関連の深い命令である Probe と match の書式を示す．

```
##### NEXT PROBE #####
Probe TCP malware q|GET / HTTP/1.0¥r¥n¥r¥n|                                  # テスト入力1
match http m|^HTTP/1¥.[01] ¥d¥d¥d.*¥r¥nDate: . ¥r¥n| p/Apache httpd/        # 応答パターン
match http m|^HTTP/1¥.[01] ¥d¥d¥d.*¥r¥nConnecti| p/Apache httpd/            # 応答パターン
match http m|^HTTP/1¥.[01] ¥d¥d¥d. +¥(([^¥r¥n¥)]+)¥)¥r¥n| p/Apache httpd/ i/$1/  # 応答パターン
match http m|^HTTP/1¥.1 ¥d¥d¥d .*¥r¥r¥n| p/IBM HTTP Server/ i/Based on Apache/  # 応答パターン
match http m|^HTTP/1¥.[01] ¥d¥d¥d . |s p/Apache Tomcat/ v/$1/               # 応答パターン
match http m|Coyote/(¥d[-¥d.]+)¥r¥n|s p|Apache Tomcat/Coyote JSP engine| v|$1|  # 応答パターン
##### NEXT PROBE #####
Probe TCP GenericLines q|¥r¥n¥r¥n|                                           # テスト入力2
match malware m|¥x83¥xec¥x20¥x8b¥xec¥x89¥x5d¥x04| p/YNU:53/                 # 応答パターン
match malware m|¥x2e¥x68¥x74¥x6d¥x6c¥x20| p/YNU:71/                         # 応答パターン
match malware m|¥x50¥x4f¥x53¥x54¥x20¥x2f¥x2b¥x38¥x33¥x35¥x39¥x2e¥x68| p/YNU:76/  # 応答パターン
match malware m|¥x2c¥x20¥x36¥x36¥x36¥x37¥x20¥x3a¥x20¥x55¥x53¥x33¥x52¥x31| p/YNU:135/  # 応答パターン
##### NEXT PROBE #####
・・・
```

図 6　Nmap 設定ファイル（nmap-service-probes）の例
Fig. 6　Example of Nmap configuration file (nmap-service-probes).

Probe <protocol> <probename> q|test input|
match <service> m|pattern| [<info>]

Probe 命令は，検査対象ホストに送信するテスト入力を定義するためのものであり，3 つの引数をとる．第 1 引数 <protocol> は使用するプロトコルの種類であり，TCP または UDP である．第 2 引数 <probename> はテスト入力の名前である．第 3 引数は実際にテスト入力として送信するデータを記述する．Probe 命令は nmap-service-probe ファイル内に現れる順に実行される．一方，match 命令は，Probe 命令により送信されたテスト入力に対して応答を受信した場合に，照合を行う方法を定義するための命令であり，nmap-service-probe ファイル内では，対応する Probe 行から次の Probe 行の間に記載される．match 命令は少なくとも 2 つの引数をとる．第 1 引数 <service> は，照合されたサービスの名称である．第 2 引数は，照合を行うための応答パターンであり，正規表現により記述される．引数 <info> には，照合されたサービスを提供しているプログラムの名称やバージョン，動作中の OS 情報などが記載されるが，省略することができる．

以上のように Nmap 設定ファイルは本システムのシグネチャと親和性の高い形式であり，前節で説明したシグネチャの生成・更新は当該設定ファイルの作成・書き換えにより実現できる．**図 6** に本システムにより更新が行われた Nmap 設定ファイルの例を示す．図 6 において，第 1 引数が “malware” となっている match 行は本システムにより追加された行であり，マルウェア感染ホスト検出に用いられる．それ以外の行は Nmap がデフォルトで有するテスト入力と応答パターンである．マルウェア検出用の match 行では，第 3 引数が

“p/YNU:ID” となっているが，この引数により応答パターンの ID 番号が認識できるようにしている．さらに，検体–応答パターン対応データベースは単一の CSV ファイルにより構成することとし，各行に ID 番号と対応する検体の MD5 値を記載することとした．なお，図 6 では省スペースのため，応答パターンの一部を省略している．

犠牲ホスト　犠牲ホストは VMware Server 1.0.6 のゲスト OS として実現した．犠牲ホストの OS は Windows XP Professional SP1 としたが，簡単な準備により任意の OS との入れ替えが可能である．分析マネージャが犠牲ホストを起動すると，Windows のタスクスケジューラによってあらかじめ設定したタスクにより検体ファイルをホスト OS から SSH によりダウンロードし，これを実行するように設定した．また，VMware のゲスト OS 上で動作していることを検出し，実行を停止するマルウェアが存在するため[4]，一定のカモフラージュ処理を行い，マルウェアによる検出に対応している．さらに，犠牲ホストを実際のマルウェア感染被害にあう一般的なクライアント PC の設定に近付けるため，Windows RPC，Windows NETBIOS といった，Windows クライアント PC が起動していることが多いサービスを事前に起動した状態で検体の実行を行う．これらのサービスは，シグネチャ生成者にとって既知の正規サービスであるため，既知正規サービスと呼ぶこととする．シグネチャ生成時は，既知正規サービス以外のサービスを不正サービスとしてシグネチャ生成の対象とする．ただし，既知正規サービスをマルウェアが停止，改変する可能性があるため，犠牲ホスト内では，プロセスモニタ，API フック[5] による API 呼び出し監視，netstat によるネットワーク状態確認といった内部挙動の監視を行い，既知正規サービスが改変されていないことを確認する．

擬似インターネットとアクセスコントローラ　擬似インターネットを構成する簡易サーバ群は，Perl スクリプトにより実装し，ホスト OS 上で起動した．また，ハニーポットプログラム Nepenthes v0.2.2 も同様にホスト OS 上で起動し，犠牲ホストからの要求に応答できるようにした．次に，アクセスコントローラは，Linux のパケットフィルタリングツールである iptables を用いた．アクセスコントローラでは NAT を行っており，ゲスト OS から外部ホストへの接続要求に対して実インターネットへの接続を許可する場合には iptables の PREROUTING チェインで ACCEPT ターゲットを用い，POSTROUTING チェインにおいて MASQUERADE ターゲットを適用することでグローバルアドレスへの変換を行い，実インターネット上のホストと通信できるようにした．また，ゲスト OS からの接続要求に対して実インターネットへの接続を許可しない場合には，擬似インターネット側にパケットを転送するため REDIRECT ターゲットを用いた．擬似インターネットを構成する簡易サーバ群はそれぞれデフォルトポートで待ち受けを行っているが，待ち受けしていないポートへの TCP 接続要求に対しては ECHO サーバが応答するようにした．アクセスコントローラと擬似インターネットの実装の詳細については文献 9) をご参照いただきたい．

分析マネージャ　分析マネージャは C 言語と Perl により実装した．システム制御部分は主に Perl を用いているが，シグネチャ生成時の最長共通部分文字列の計算には C 言語で実装された Perl モジュール String::LCSS_XS[16] を用い，通信ログ（pcap）からのストリームデータ再構築については C 言語により実装した．

4. 評 価 実 験

提案手法は，テスト入力へのマルウェアの応答という，必ずしも確定的ではない挙動を基に検出を行う．そのため，まず基本的な検出能力を検証するため，実験 1 では評価用の検体セットからシグネチャを生成し，これを用いて同一の検体セットが実際に検出できるかどうかを確認した．次に，実験 2 ではシグネチャ生成用と検出実験用とで異なる検体セットを用意し，検出精度を評価した．最後に実験 3 では，実験 1 と実験 2 で生成したシグネチャにより，正規サービスのみが動作するホストの検査を行い，誤検出の有無を確認した．以降では，これらの評価実験について説明する．

4.1 実 験 方 法

検体　実験に用いた検体は，ハニーポット Nepenthes により 2009 年 6 月から 7 月の約 2 カ月間で収集した 434 体の実マルウェアである．これらの検体を 2009 年 1 月 1 日にウイルス対策ソフトによる検出結果を提供するサービスである VirusTotal.com[18] に提出し，得られた結果を**図 7** に示す．

（実験 1）シグネチャ生成時と同一の検体セットに対する検出精度の評価

実験 1 では，前述の 434 検体を用いて，シグネチャの生成を行った．シグネチャ生成に用いたパラメータは事前実験の結果に基づき以下のとおりとした．なお，アクセスコントローラのフィルタリングルールは文献 9) における評価実験により得られたルールを用いた．また，ネットワークスキャナのシグネチャの初期状態は，Nmap v5.0 の設定ファイル nmap-service-probe のデフォルト設定を用いた[⋆1]．

⋆1 Nmap v5.0 には，様々なサービスを判別するためにデフォルトで 57 種類のテスト入力と 5,482 種類の応答パターンが用意されているため，これらをシグネチャの初期状態として用いた．なお，上記のデフォルト応答パターンにより認識されたサービスと，提案方式により生成したマルウェア検出のための応答パターンを区別するため，後者については match 行の第 2 引数を “malware” とした（図 7 の例を参照のこと）．

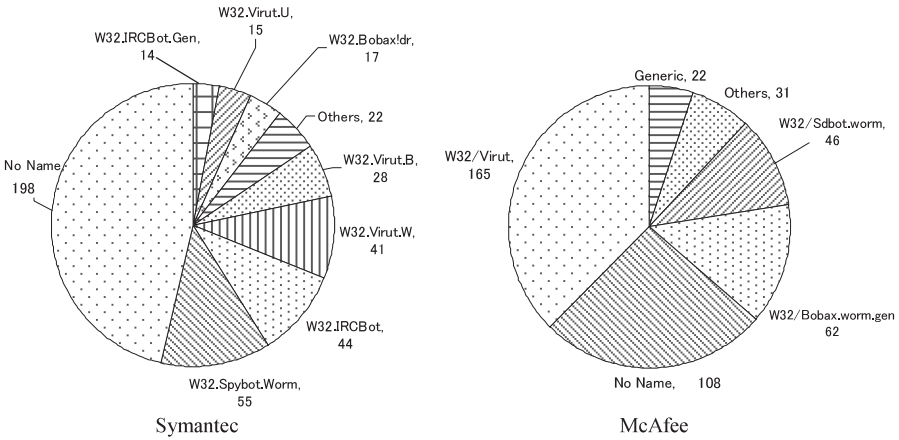

図 7　実験に用いたマルウェア検体のウイルス対策ソフトによる名称
Fig. 7　Malware families used for experiments obtained by anti-virus software.

検体解析回数：6[回]	**スキャン開始待ち時間**T_s：30[秒]	**最短応答パターン**$Th_{\min}$：4[バイト]
検体実行時間：30[秒]	**スキャン応答待ち時間**T_w：6[秒]	**最長応答パターン**$Th_{\max}$：25[バイト]

さらに，前述のとおり生成したシグネチャを用いて，同様の 434 検体の感染を正しく検出できるかを確認した．すなわち，これらの検体を動的解析環境内の犠牲ホスト上で実行したうえで，生成済みのシグネチャを用いて検査を行い，正しく感染が検出されるかを確認した．検出実験は各検体について 5 回ずつ行った．なお，犠牲ホスト上では，既知正規サービスとして Microsoft Windows RPC（135/TCP, 1025/TCP），Microsoft Windows netbios-ssn（139/TCP），Microsoft Windows XP Microsoft-ds（445/TCP），Microsoft Windows UPnP（5000/TCP）を起動しておき，これらのサービスに対する検査結果も確認した．

（実験 2）シグネチャ生成時と異なる検体セットに対する検出精度の評価

実験 2 では，前述の 434 検体を 100 検体からなるシグネチャ生成用検体セットと 334 検体からなる検出実験用検体セットにランダムに分割し，実験 1 と同様のパラメータおよび環境設定でシグネチャ生成を行い，検出精度の評価を行った．

（実験 3）正規サービスに対する検出精度の評価

実験 1 および実験 2 で生成したシグネチャを用いて，本論文末尾に示す**表 8** にあげる 33 種類の正規サービスの検査を行い，検出精度の評価を行った．具体的には，実験 1 と同様のパラメータおよび環境設定で，表 8 の正規サービスを実行し，各シグネチャによって不正サービスとして誤検知されることがないかを確認した．

表 1　シグネチャ生成の結果（実験 1）
Table 1　Result of signature generation in experiment 1.

全検体数	434 [体]
ポート待ち受け状態が確認された検体	132 [体]
生成されたテスト入力	231 [種類]
生成された応答パターン	336 [種類]

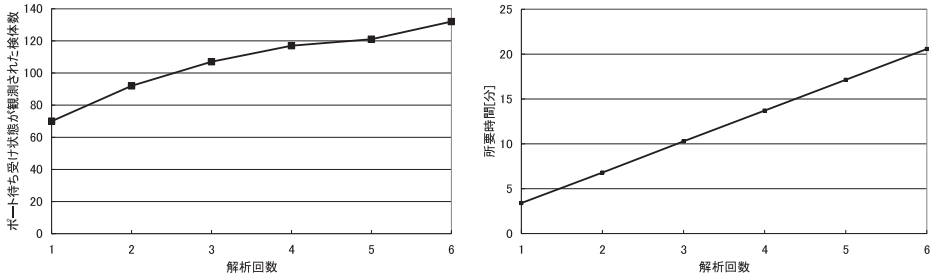

図 8　（左）検体の解析回数とポート待ち受け状態が観測された検体数の関係（実験 1）.（右）検体の解析回数とシグネチャ生成の所要時間の関係（実験 1）
Fig. 8　(Left) Relationship between number of executions of malware sample and number of malware samples that started listening on a port upon execution in experiment 1. (Right) Relationship between number of executions of malware sample and time for signature generation in experiment 1.

4.2　実 験 結 果

実験 1 の結果　実験 1 のシグネチャ生成の結果を**表 1** に示す．全 434 検体のうち，30%にあたる 132 検体から，6 回の解析の中で少なくとも 1 回はポート待ち受け状態が確認できた．また，これらの 132 検体の解析により生成されたテスト入力は 231 種類，生成された応答パターンは 336 種類となった．さらに，これらの不正サービスが Nmap のデフォルトシグネチャにより正規サービスとして誤検知されることはなかった．検体の解析回数とポート待ち受け状態となった検体数の関係を**図 8**（左）に示す．図 8（左）から，検体の解析回数を増やすことでポート待ち受け状態が確認された検体の数がほぼ一定に増加していることが分かる．これは，今回の実験で用いた検体のほとんどがボットであり，C&C サーバへの接続やマルウェアダウンロードサイトへの接続状況に依存して挙動が変わることに起因

表 2　検体 1 体あたりの 1 回の解析の所要時間内訳（実験 1）
Table 2　Breakdown of analysis time per sample per execution in experiment 1.

アクセスコントローラ設定	5 [秒]
犠牲ホスト起動	50 [体]
検体実行	30 [秒]
犠牲ホストのイメージリカバリ	65 [秒]
シグネチャ生成	54 [秒]
所要時間合計	**204 [秒]**

している[8]．これらのサーバへの接続の可否とポート待ち受け状態の関連性についてはさらに詳細な調査が必要であるが，いくつかの検体について調査した範囲では，これらのサーバに接続が成功した場合にポート待ち受け状態となっており，ポート待ち受け状態に何らかの影響を与えているものと思われる．今回の実験では各検体について 6 回の独立な解析を行ったが，図 8（左）の増加傾向から解析回数を増やすことでさらに多くの検体のポート待ち受け状態を観測できることが期待される．

所要時間　次に，検体の解析回数とシグネチャ生成の所要時間の関係を図 8（右）に示す．解析回数の増加にともない，シグネチャ生成の所要時間もほぼ一定に増加している．すなわち，解析回数を増やし，より多様なポート待ち受け状態を観測することで，シグネチャのカバレッジは高くなるが，一方で，解析の所要時間も増加するという，トレードオフの関係になっている．なお，所要時間の内訳（**表 2**）を見ると，本システムのシグネチャ生成・更新フェーズにおいて最も所要時間の大きな処理は，犠牲ホストのイメージリカバリであることが分かる．近年，マルウェア動的解析における利用を想定した，OS イメージ復元の高速化に関する研究が進められており[10]，将来的には所要時間の削減が期待できる．また，RAM ディスクや SSD（Solid State Drive）により，犠牲ホスト起動やリカバリの速度の大幅な向上が期待できる．

待ち受けポート　今回のシグネチャ生成時および検出実験時に待ち受け状態となったポートは全部で 414 種類であった．各ポートで待ち受け状態となった検体数を**表 3** に示す．3128/TCP は突出して多い 75 検体において待ち受け状態となった．3128/TCP はプロキシサーバプログラムである Squid のデフォルト待ち受けポートであるが，当該ポートでのマルウェア検体の応答は通常の Squid の応答とは異なっており検出は容易であった．次に 113/TCP は 3 検体において待ち受け状態となった．そのうち 2 体は 3.1 節で説明したとおりの応答だったが，残りの 1 体はどのような入力に対しても同様の文字列を返すように実装されていたため，いずれの場合も容易に応答パターンを生成できた．3128/TCP と 113/TCP 以外のポートは，434 検体の解析を通じて 1〜2 回しか待ち受け状態が観測されなかった．多くの検体は実行するたびに異なるポート番号で待ち受けを行うことから，このような多様なポートでの待ち受けが観測されたものと考えられる．しかしながら，これらのポートで実際に動作するサービスの種類は限定的であり，後述する検出実験からも分かるとおり，提案手法によって生成したシグネチャにより検出を行えた．

表 3　待ち受けポート番号と検体数の関係（実験 1）
Table 3　Ports that were listened by malware samples in experiment 1.

検体数[体]	ポート番号
75	3128/TCP
3	113/TCP
2	2326, 2329, 6284, 10171, 17975/TCP
1	407種類

表 4　検出実験結果（実験 1）
Table 4　Result of detection of infected host in experiment 1.

全検体数	434 [体]
各検体の検出実験回数	5 [回]
ポート待ち受け状態となった検体数	121 [体]
検出に成功した検体数	121 [体]
シグネチャ生成時のみポート待ち受け状態となった検体数	11[体]
検知に成功した不正サービス数	457 [件]
誤検知した正規サービス数	0 [件]

検出実験結果　上記のとおり生成したシグネチャを用いて，434 検体の検出実験を行った結果を**表 4** に示す．表 4 のとおり，各検体について 5 回ずつ検出実験を行った結果，5 回のうち少なくとも 1 回はポート待ち受け状態となった検体数は 121 体であり，いずれも不正サービスとして正しく検出が行えた．なお，これら 121 検体はいずれもシグネチャ生成時にポート待ち受け状態となった検体だった．この際，検出に成功したマルウェアによる不正サービスの総数は 457 件だった．なお，シグネチャ生成時にはポート待ち受け状態となった検体のうち 11 検体については，検出実験時にポート待ち受け状態にならないケースが確認されたが，その理由としては前述のとおり，C&C サーバやマルウェアダウンロードサーバへの接続状態の変化やマルウェア自体の確率的な挙動が考えられる．また，犠牲ホスト上で動作中の Windows の既知正規サービスである Microsoft Windows RPC（135/TCP, 1025/TCP），Microsoft Windows netbios-ssn（139/TCP），Microsoft Windows XP Microsoft-ds（445/TCP），

表 5　シグネチャ生成の結果（実験 2）
Table 5　Result of signature generation in experiment 2.

全検体数	100 [体]
ポート待ち受け状態が確認された検体	25 [体]
生成されたテスト入力	94 [種類]
生成された応答パターン	135 [種類]

表 6　検出実験結果（実験 2）
Table 6　Result of detection of infected host in experiment 2.

全検体数	334 [体]
各検体の検出実験回数	5 [回]
ポート待ち受け状態となった検体数	97 [体]
検出に成功した検体数	97 [体]
検知に成功した不正サービス数	392 [件]
誤検知した既知正規サービス数	0 [件]

Microsoft Windows UPnP（5000/TCP）が，マルウェアによるサービスとして誤検出されることはなかった．

実験 2 の結果　次に実験 2 のシグネチャ生成の結果および検出実験結果を**表 5** と**表 6** にそれぞれ示す．実験 2 では，シグネチャ生成に用いた 100 検体中，25 検体が 6 回の解析の中で少なくとも 1 回はポート待ち受け状態となった．この際生成されたテスト入力は 94 種類であり，応答パターンは 135 種類であった．検出実験に用いた 334 検体のうち，5 回の検出実験中，少なくとも 1 回はポート待ち受け状態となった検体数は 97 体であり，そのいずれもマルウェア感染状態を正しく検出できた．また，既知正規サービスを不正サービスとして誤検知することはなかった．

実験 3 の結果　表 8 に示した 33 種類の正規サービスは，実験 1 および実験 2 において生成したシグネチャによって，誤検知されることはなかった．

4.3 考　　察

実験 1 では，シグネチャ生成時と検出実験で同一の検体を用いた場合に，ポート待ち受け状態になったすべての検体の感染を見逃しなく検出できた．さらに，実験 2 では，シグネチャ生成時と異なる検体セットについても，同様に見逃しなく検出できた．さらに，実験 3 では，正規サービス群を誤検出することはなかった．このことから，提案手法により，評価実験で用いた検体の不正サービスの特徴をとらえたシグネチャが生成できていると考える．本節では，提案手法の限界と課題について考察する．

提案手法の限界　原理的に提案手法は感染時にポート待ち受け状態となるマルウェアにしか適用できない．したがって，提案手法による検査によってマルウェア感染が確認されなかったからといって，マルウェアに感染していないと結論付けることはできない点に注意が必要である．

また，提案手法では，同一のテスト入力に対するマルウェアからの応答に一定の規則性があることを前提とし，複数の応答の共通部分文字列を照合用の応答パターンとして用いている．そのため，この前提が成り立たない場合は，検出精度が極端に落ちたり，生成されるシグネチャの数が爆発的に増加したりする可能性がある．たとえば，SSL や SSH などの暗号通信では毎回異なる鍵を用いて通信を行うため，応答メッセージが一定でなくなることから注意が必要である．しかしながら，これらの暗号通信においても，プロトコルの初期段階では規則性の高いメッセージがやりとりされるため，提案手法はある程度有効であると考える．独自のプロトコルにより，通信の初期段階から規則性の導出が困難な応答メッセージを送信するマルウェアに対しては，提案手法による検出は困難と思われるが，現在のところそのようなマルウェアは確認できていない．

さらに，正規サービスに忠実に実装されたサーバ機能を有するマルウェアは，正規サービスとして検出するため，マルウェア感染を見逃すことになる．このような場合は，スキャンにより収集可能な情報，たとえば OS の種類やバージョンなどとの整合性を含めた総合的な判断により，ある程度の異常状態を推定することができると思われるが，さらに詳細に検討する必要がある．

シグネチャの増大にともなうネットワーク負荷　提案手法の運用を重ね，多数の検体に対するシグネチャを有するようになると，検査パケットの増大やそれにともなうネットワーク負荷の増大といった問題が発生することが懸念される．そのため，検査パケットの増大に直接影響するテスト入力数の増加を抑えるため，類似のテスト入力の統合や不必要なテスト入力の削除といった効率化の方策をとる必要がある．**表 7** に実験 1 および実験 2 において生成されたシグネチャと検出した不正サービス数を示す．表 7 から 3 種類のテスト入力 “GET /HTTP /1.0¥r¥n¥r¥n”，“NULL”（テスト入力なし），“¥r¥n¥r¥n” によって，実験 1 では不正サービスの 98.9%にあたる 452 サービス，実験 2 では 97.7%にあたる 383 サービスの検出が行えたことが分かる．このことから，テスト入力数を削減する余地が残されているといえる．

フィルタリングの影響　提案手法は遠隔の検査対象ホストに対してネットワーク越しに検査を行う手法であるため，検査者と検査対象ホストの間のネットワークにおいてパケットフィ

表 7　評価実験で生成されたシグネチャと検出した不正サービス数
Table 7　Signatures generated in experiments and number of detected malicious services.

ID	テスト入力	応答パターン(一部省略)	検出した不正サービス数(実験1)	検出した不正サービス数(実験2)	類似サービス・プロトコル(推定)
67	GET / HTTP/1.0¥r¥n¥r¥n	HTTP/1.0 200 OK¥r¥n¥r¥nYES¥r¥n	352	305	HTTP Proxy
53	NULL	¥x83¥xec¥x20…¥x8b¥x43	86	71	Backdoor
91	¥r¥n¥r¥n	: US	7	7	Auth/Ident
16	GET / HTTP/1.0¥r¥n¥r¥n	¥x83¥xec¥x8b…¥x8b¥x43	4	0	Backdoor
135	NULL	, 6667 : ... : T3C	3	0	Auth/Ident
111	OPTIONS / RTSP/1.0¥r¥n¥r¥n	3639, 6667 : ¥x55¥x53…¥x43	1	0	Auth/Ident
	その他のシグネチャ		4	9	

表 8　検出実験の対象とした正規サービスリスト
Table 8　List of legitimate services used in experiment 3.

Authd v1.4.3	**Microsoft ESMTP 6.0.2600.5512**
Epson printer httpd 1.0	**Microsoft Windows kerberos-sec**
Epson Network print server telnetd	**Microsoft Windows XP microsoft-ds**
Vsftpd 2.0.5	**Microsoft Windows 2000 microsoft-ds**
Polycom VSX 8000 ftpd	**Microsoft Terminal Service microsoft-rdp**
Polycom VSX 8000 http config (NetPort httpd 1.1)	**Microsoft Windows XP RPC**
Polycom VSX 8000 MGC Manager	**Microsoft Windows RPC over HTTP 1.0**
Polycom VSX 6000A telnetd 8.7.0.1	**Microsoft Windows XP netbios-ssn**
Apache httpd 2.2.3	**Microsoft Windows 2000 Server Wins**
Apache httpd 2.2.6	**Microsoft IIS webserver 5.1**
Cyrus-imapd 2.3.7	**Microsoft Windows 2000 Server ldapd**
Dovecot imapd 1.0	**Microsoft Windows 2000 Server dnsd**
Epson ippd 1.0	**OpenSSH 4.3 (protocol 2.0) sshd**
Dovecot pop3d	**OpenSSH 4.5 (protocol 2.0) sshd**
Skype v3.8	**OpenSSH 5.1 (protocol 2.0) sshd**
Postfix smtpd 2.3	**VMware Authentication Daemon 1.10**
Telnet server0.1.7	

ルタリングなどが行われていると，検出が正しく行えない可能性がある．しかし，このような状況は遠隔の攻撃者側にとっても同様であるため，当該ポートの用途によっては，フィルタリングをされる可能性が低いポート番号を攻撃者は選択することも予想される．

検査対象ホストからの攻撃　脆弱性スキャンなどの検査のためのネットワークスキャンを受けると，スキャン元に対して攻撃を仕掛けるマルウェアの存在が報告されている[20]．文献 20) では Storm ボットのノードが脆弱性スキャナの検査を検知すると，自動的にスキャナに対してボットネットによる DDoS を行う例が報告されている．このようなケースでは，スキャンを受けたノードからボットネットの制御サーバに対して DDoS のトリガとなる通信が発生しているはずであるため，このトリガを検出・遮断することができれば，攻撃を受けずに検出を行える可能性がある．

5. おわりに

本論文では，ネットワークスキャンにより，検査対象ホストのポート待ち受け状態やテスト入力に対する応答を観測し，マルウェア感染の有無を推定する方法を提案した．提案手法では，マルウェア動的解析によって，マルウェア検体のポート待ち受け状態やテスト入力に対する応答を観測し，自動的に検出用シグネチャを生成する方法を示し，システム実装を行った．実マルウェア検体を用いた評価実験の結果，評価実験に用いた検体については，高い精度で検出を行えることが確認された．また，実験で用いた正規サービスに対して，誤検出がないことを確認した．今後の課題は，より多くの検体と正規プログラムを用いた評価，シグネチャ生成の効率化である．

参考文献

1) Foreman, P.: *Vulnerability Management*, Auerbach Pub., ISBN 1439801509 (2009).
2) Wilhelm, T.: *Professional Penetration Testing: Creating and Operating a Formal Hacking Lab.*, Syngress, ISBN 1597494259 (2009).
3) Baecher, P., Koetter, M., Holz, T., Dornseif, M. and Freiling, F.C.: The Nepenthes Platform: An Efficient Approach to Collect Malware, *Proc. 9th International Symposium on Recent Advances in Intrusion Detection* (*RAID 2006*), pp.165–184 (2006).
4) Chen, X., Andersen, J., Mao, Z.M., Bailey, M. and Nazario, J.: Towards an Understanding of Anti-virtualization and Anti-debugging Behavior in Modern Malware, *Proc. 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (*DSN 2008*), pp.177–186 (2008).
5) Father, H.: Hooking Windows API: Techniques of Hooking API Functions on Windows, *CodeBreakers Journal*, Vol.1, No.2 (2004).
http://www.codebreakers-journal.com/downloads/cbj/2004/CBJ_1_2_2004_HolyFather_Hooking_Windows_API.pdf (Last Visit: 2010/03/30)
6) Porras, P., Saidi, H. and Yegneswaran, V.: Conficker C P2P Protocol and Imple-

mentation, SRI International Technical Report (2009).
http://mtc.sri.com/Conficker/P2P/index.html (Last Visit: 2010/03/30)
7) Yoshioka, K., Inoue, D., Eto, M., Hoshizawa, Y., Nogawa, H. and Nakao, K.: Malware Sandbox Analysis for Secure Observation of Vulnerability Exploitation, *IEICE Trans.*, Vol.E92D, No.5, pp.955–966 (2009).
8) Yoshioka, K., Kasama, T. and Matsumoto, T.: Sandbox Analysis with Controlled Internet Connection for Observing Temporal Changes of Malware Behavior, *Proc. 4th Joint Workshop on Information Security, JWIS 2009*, 3a-2 (2009).
9) Yoshioka, K. and Matsumoto, T.: Multi-pass Malware Sandbox Analysis with Controlled Internet Connection, *IEICE Trans.*, Vol.E93-A, No.1, pp.210–218 (2010).
10) 朝倉康生，曽根直人，森井昌克：マルウェア解析システムにおけるクライアントサーバモデルを用いた復元方法の提案とその実装，情報処理学会コンピュータセキュリティシンポジウム 2009 論文集，セッション F4 (2009).
11) Authentication Service, RFC 912. http://www.rfc-editor.org/rfc/rfc912.txt (Last Visit: 2010/03/30)
12) Fuzz Testing of Application Reliability, The University of Wisconsin. http://pages.cs.wisc.edu/∼bart/fuzz/ (Last Visit: 2010/03/30).
13) Identification Protocol, RFC 1413.
http://www.rfc-editor.org/rfc/rfc1413.txt (Last Visit: 2010/03/30)
14) Nmap v5.0. http://nmap.org/5/ (Last Visit: 2010/03/30)
15) Script p2p-conficker.nse.
http://nmap.org/nsedoc/scripts/p2p-conficker.html (Last Visit: 2010/03/30)
16) String::LCSS_XS v1.0. http://kobesearch.cpan.org/htdocs/String-LCSS_XS/String/LCSS_XS.html (Last Visit: 2010/03/30)
17) TR/Proxy.Smail.bt.3.
http://www.avira.de/jp/threats/section/fulldetails/id_vir/1285/tr_proxy.small.bt.3.html (Last Visit: 2010/03/30)
18) VirusTotal. http://www.virustotal.com/jp/ (Last Visit: 2010/03/30)
19) W32/Sasser.worm. http://www.mcafee.com/japan/security/virS2004.asp?v=W32/Sasser.worm (Last Visit: 2010/03/30)
20) Universities warned of Storm Worm attacks.
http://www.securityfocus.com/news/11482 (Last Visit: 2010/03/30)
21) Retina Vulnerability Management and Assessment, eEye Digital Security. http://www.eeye.com/Products.aspx (Last Visit: 2010/03/30)

(平成 21 年 11 月 30 日受付)
(平成 22 年 6 月 3 日採録)

吉岡　克成（正会員）

2005 年 3 月横浜国立大学大学院環境情報学府情報メディア環境学専攻博士課程後期修了，博士（工学）．同年 4 月独立行政法人情報通信研究機構研究員．2007 年 12 月より横浜国立大学学際プロジェクト研究センター特任教員（助教）．マルウェア解析やネットワーク攻撃観測・検知等のネットワークセキュリティの研究に従事．2009 年文部科学大臣表彰・科学技術賞（研究部門）受賞．

村上　洸介

2010 年横浜国立大学工学部電子情報工学科卒業．現在，同大学大学院環境情報学府博士課程前期に在学中．情報セキュリティ，特に，ネットワークセキュリティの研究に従事．

松本　　勉（正会員）

1986 年 3 月東京大学大学院工学系研究科電子工学専攻博士課程修了，工学博士．同年 4 月横浜国立大学講師．2001 年 4 月より同大学院環境情報研究院教授．現在，同大学教育研究評議員，日本学術会議連携会員，国際暗号学会 IACR 理事．暗号アルゴリズム・プロトコル，耐タンパ技術，生体認証，人工物メトリクス等の「情報・物理セキュリティ」の研究教育に 1981 年より従事．1982 年にオープンな学術的暗号研究を目指した「明るい暗号研究会」を 4 名で創設．1994 年第 32 回電子情報通信学会業績賞，2006 年第 5 回ドコモ・モバイル・サイエンス賞，2008 年第 4 回情報セキュリティ文化賞，2010 年文部科学大臣表彰・科学技術賞（研究部門）受賞．