Computer Security Symposium 2016
11 - 13 October 2016

フォールトトレランスを有するアタックレジリエントサーバの構築とその性能評価

佐野　史和[†1]　岡本　剛[†1]　Idris Winarno[†2]　畑　良知[†2]　石田　好輝[†2]

[†1] 神奈川工科大学　〒243-0292　神奈川県厚木市下荻野 1030
[†2] 豊橋技術科学大学　〒441-8580　愛知県豊橋市天伯町雲雀ヶ丘 1-1

概要：本研究は，サイバー攻撃に対して回復力のあるレジリエントサーバを構築することを目的とする．レジリエントサーバは，異なる OS とサーバアプリケーションの実装の仮想マシン群で構成され，いずれか 1 つの仮想マシンによりサービスを提供する．サイバー攻撃によりサービスが停止したら，他の仮想マシンに切り替える．これは，1 つの脆弱性が異なる OS やサーバアプリケーションの実装に影響しないことに基づく（仕様上の脆弱性や共有ライブラリの脆弱性を除く）．本稿では，VMware vSphere FT により，耐故障機能を有するアタックレジリエントサーバを構築し，サイバー攻撃によるダウンタイムを計測し，その結果を考察する．

キーワード：サイバー攻撃，仮想化，脆弱性，フォールトトレランス，多様性

An implementation and its evaluation of a cyber attack-resilient server with fault tolerance

Fumikazu Sano[†1]　Takeshi Okamoto[†1]
Idris Winarno[†2]　Hata Yoshikazu[†2]　Ishida Yoshiteru[†2]

Abstract: This study focuses a cyber attack-resilient server inspired by the concept of biological diversity. The server consists of several virtual machines running different operating systems and different implementations of the same server protocol specification. This approach is based on the observation that not all implementations are affected by the same vulnerability, except for vulnerabilities in specifications and on shared libraries. If cyber attacks are detected, a virtual machine changer switches the current virtual machine to the next virtual machine. This paper reports an implementation and its evaluation of the cyber attack-resilient server with fault tolerance.

Keywords: cyber attack, virtualization, vulnerability, fault tolerance, diversity

1. はじめに

IoT 時代の幕開けとともにミッションクリティカルなインターネットサービスの需要が高まりつつある．ミッションクリティカルなサーバの運用技術として，フォールトトレランス設計がある．しかし，フォールトトレランス設計は，サーバアプリケーションの脆弱性を突くようなサイバー攻撃を想定していない．特に，任意のコードを実行できる脆弱性に対するサイバー攻撃は，機密性，完全性，可用性のあらゆるセキュリティ侵害が予想される．

ノートンセキュリティやウイルスバスターなどのパーソナルコンピューター用のセキュリティ対策ソフトは，既知のサイバー攻撃を検知し防ぐことができるが，未知の攻撃は検知も防ぐこともできないおそれがある．この他に，Palo Alto Networks traps，CrowdStrike Falcon Host，FFRI yarai，Microsoft EMET などの次世代型セキュリティ対策ソフトは，未知のサイバー攻撃を防ぎ，検知することができる．学術研究機関では，ROPGuard[1]，ROPecker[2]，SecondDEP[3]など様々な未知の攻撃を検知する技術がある．

セキュリティ対策ソフト以外には，ファイル操作やメモリ操作など様々な OS の機能をアクセス制御する強制アクセス制御（Mandatory access control：MAC）がある．MAC を適用すれば，ネットワーク外部や内部からのゼロデイ攻撃から，OS とアプリケーションを保護できる．

しかし，これらの製品や技術がサイバー攻撃を検出したとき，すでにサーバアプリケーションは正常な実行の制御を失っているので，サーバアプリケーションを強制的に終了せざるを得ない．ここで，サーバアプリケーションを再起動しても，同じ攻撃により，再び，実行の制御を失うため，再起動だけでは攻撃を防げない．

この問題を解決するために，本研究では，生物学的多様性をヒントにして，サイバー攻撃に対して頑健なアタックレジリエントサーバを設計・実装してきた[4, 5]．アタックレジリエントサーバは，異なる OS と異なるサーバアプリケーションの仮想マシン群から構成され，その中の 1 つの仮想マシンがサービスを提供する．サイバー攻撃を検知し

†1 神奈川工科大学　Kanagawa Institute of Technology †2 豊橋技術科学大学　Toyohashi University of Technology

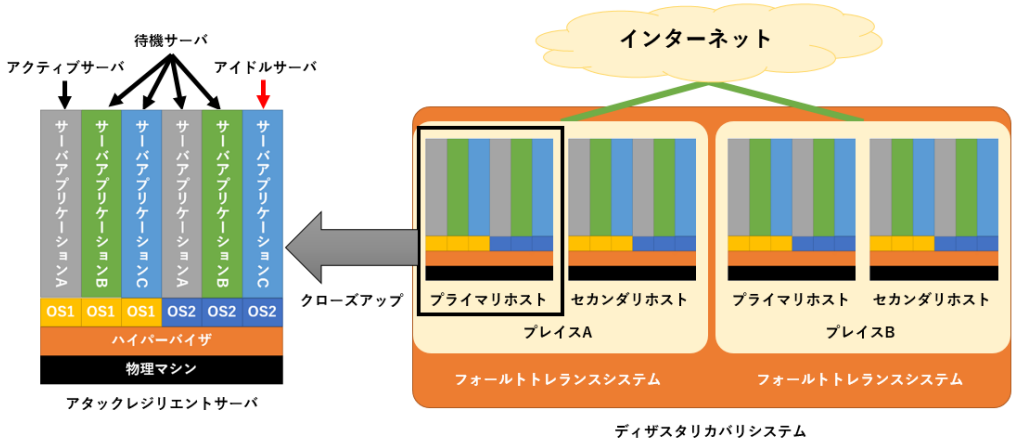

図 1 アタックレジリエントサーバの構成

たら，他の仮想マシンに切り替えてサービスを継続する．このアプローチは，1 つのサーバアプリケーションの実装に脆弱性があっても，他の実装では同様の脆弱性はないという観測に基づいている（ただし，libc など共有ライブラリの脆弱性や仕様上の脆弱性を除く）．このアプローチは，「1 つの種の感染による影響が，その他すべての種に同様の影響を与えることはない」という種の生存と適応において，重要な役割を果たす生物の遺伝的多様性をヒントにしている．

本稿では，これまで設計・実装を行ってきたアタックレジリエントサーバに耐故障機能を導入する方法と，導入前と導入後の性能の違いについて比較評価する．

2. サイバー攻撃

サーバの代表的なセキュリティインシデントには，ハッキング，ウェブサイトの改ざん，情報漏洩，サービスの妨害（DoS）がある．サーバやネットワークのリソースを枯渇させるフラッド攻撃を除けば，これらのインシデントは脆弱性によるものが多い．また，これらの脆弱性の原因は，「システムの設計や実装の不備」，または，「サーバアプリケーションの設定や運用の方法の不備」に分類できる．前者は， OS のセキュリティ機能の利用など技術的なアプローチによって解決でき，後者は，組織全体のセキュリティリテラシー向上を図るといった組織的アプローチによって改善できる．本研究は，前者の脆弱性に対するサイバー攻撃を対象とする．

3. ハイブリッド仮想化によるアタックレジリエントサーバ

アタックレジリエントサーバは，1 台の物理サーバから構成される最小構成と，耐故障機能を有するフォールトトレラントシステム構成，災害に耐性をもつディザスタリカバリシステムの構成に分けられる．図 1 にその構成を示す．

3.1 物理サーバの構成

物理サーバは，複数の仮想マシンから構成され，いずれか 1 つの仮想マシンがサービスを提供する．仮想マシンには，それぞれ実装の異なるサーバアプリケーションと異なる OS が稼働する．仕様上の脆弱性と共有ライブラリの脆弱性を除けば，脆弱性はサーバアプリケーションの実装に依存するため，構成の異なるすべてのサーバアプリケーションが同様の脆弱性の影響を受けることはほとんどないと予想される．このアプローチは，仮想マシンの多様性により攻撃の耐性を得る．したがって，物理マシンには，最低でも 2 つの異なる OS と，実装が異なる 2 つのサーバアプリケーションの組み合わせからなる合計 4 つの仮想マシンが必要である．DNS であれば，実装は 20 種類あり，代表的な 4 種の OS（BSD， Linux， Solaris， Windows）がこれらの DNS アプリケーションをすべてサポートした場合，組み合わせは 80 通りになる．この組み合わせ数が多いほど，より耐性が高くなると考えられる．ただし，アタックレジリエントサーバすべてのサーバアプリケーションにおいて共通のライブラリがあってはならない．共通のライブラリに脆弱性があれば，そのライブラリを利用するサーバアプリケーションが同様の影響を受けるためである．例えば，共有ライブラリである OpenSSL のハートブリードと呼ばれる脆弱性（CVE-2014-0160）は，ウェブサービスを提供する多くの実装に影響を与えた．

ここから，物理サーバが利用する仮想化技術と仮想マシンを管理する仕組みについて述べる．

3.1.1 仮想化技術

物理サーバは，仮想マシンを利用するために，2 つの仮想化技術を使う．1 つは，マシンレベルでサーバマシンを仮想化するハイパーバイザを，もう 1 つは，アプリケーションレベルでサーバアプリケーションを仮想化するコンテナを使用する．ハイパーバイザは多くの OS をサポートできるが，マシンレベルの仮想化であるため，物理サーバのリソースを多く必要とする．一方，コンテナは，OS ごとに仕組みが異なるため，OS の多様性を確保できないが，ハイパーバイザに比べて極めて少ないリソースでサーバアプリケーションを実行できる．これらの欠点を補うために，ア

タックレジリエントサーバは，これらを組み合わせて仮想マシンを管理運用する．

3.1.2 仮想マシンの状態

各仮想マシンは，アクティブ，アイドル，サスペンドのいずれかの状態にある．アクティブ状態の仮想マシンは，サーバアプリケーションを起動する．アイドル状態の仮想マシンは，アクティブ状態の仮想マシンから瞬時にサービスを切り替えられるように起動した状態でサービスを提供しないで待機している．サスペンド状態にある仮想マシンは，アクティブ状態の仮想マシンからアイドル状態の仮想マシンに切り替わったときに，速やかにアイドル状態の仮想マシンを提供するために，サスペンドした状態で待機する．

仮想マシンはサイバー攻撃を検知するため，追加のセキュリティモジュールまたは OS が提供するセキュリティ機能を利用する．サイバー攻撃を検知したら，アクティブ状態の仮想マシンはサスペンド状態に移行し，アイドル状態の仮想マシンがアクティブ状態に移行し，サービスの提供を開始する．

3.1.3 仮想マシンの管理サーバ

仮想マシンの管理は，管理サーバが行う．管理サーバは，ネットワーク経由でハイパーバイザと接続して，サービスモニタと仮想マシンチェンジャーにより仮想マシンを制御する．サービスモニタは，アクティブ状態の仮想マシンが提供するサービスの状態を定期的にチェックし，サービスの停止を検知したら，仮想マシンチェンジャーに通知する．つづいて，アクティブ状態の仮想マシンをサスペンドさせ，攻撃されたことを示すラベルを付加する．同時に，アイドル状態の仮想マシンをアクティブ状態に昇格させ，サービスの提供を開始する．また，バックグラウンドでサスペンド状態にある仮想マシンはアイドル状態に昇格する．キューに待機している仮想マシンが尽きるまで，これらの一連の処理が繰り返される．最後に，仮想マシンチェンジャーはサイバー攻撃を検知したことを管理者やセキュリティアナリストに通知する．図 2 は，サイバー攻撃を検知したときの仮想マシンを切り替える流れである．

管理者やセキュリティアナリストは，サイバー攻撃の通知を受けると，攻撃された仮想マシンをレジュームし，停止したサーバアプリケーションのプロセスメモリを分析できる．脆弱性の発見後は，セキュリティ更新プログラムを適用した後に，待機サーバに再び追加できる．すべての仮想マシンが攻撃された場合，管理者やセキュリティアナリストが攻撃された仮想マシンにセキュリティ更新プログラムを適用するまで，アタックレジリエントサーバはサービスを停止する．

3.2 フォールトトレラントシステムの構成

フォールトトレラントシステムは，2 つ以上の物理サーバから構成される．本稿では，これを実現する方法を 4 章で，その性能を 5 章で詳しく述べる．

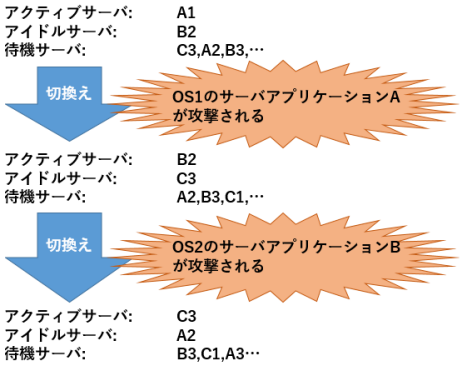

図 3　仮想マシンの切り替えの流れ

3.3 ディザスタリカバリシステムの構成

ディザスタリカバリシステムは，災害復旧システムである．ディザスタリカバリシステムは，2 つ以上のフォールトトレラントシステムによって構成され，洪水や地震による災害に対して回復力を実現する．

4. プロトタイプシステム

サービスの継続性を評価するため，DNS サーバのためのプロトタイプシステムを構築した（図 3）．DNS サービスはインターネットにおける基盤であり，DNS サービスの継続能力の向上がインターネットの強靭化につながる．プロトタイプシステムは，管理アプライアンス，ハイパーバイザ，NAS（Network attached storage）から構成される．管理アプライアンスは，VMware vCenter Server Appliance 6.0.0 である．ハイパーバイザには，VMware vSphere 6 Enterprise Plus に含まれる VMware ESXi 6.0 を使用する．NAS は NETGEAR ReadyNAS 102 であり，2 つのハイパーバイザ間で仮想マシンの構成ファイルとタイブレーカファイルを共有するために使用する．フォールトトレランスを有効にする方法やフォールトトレランスの機能に利用する 3 つのネットワーク（管理トラフィック用ネットワーク，FT ログ用ネットワーク，vMotion トラフィック用ネットワーク）は 4.4.2 節で詳しく述べる．

ハイパーバイザをインストールした物理マシンの仕様は以下の通りである．

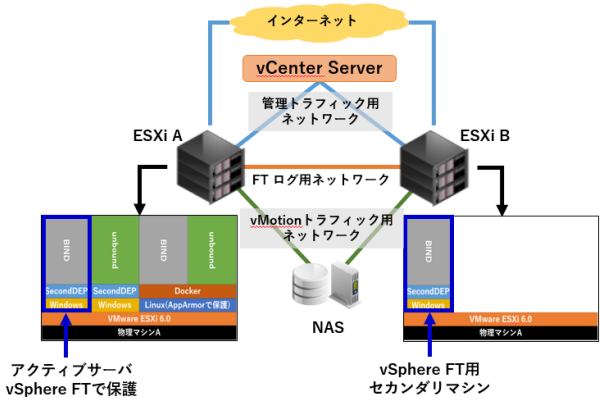

図 2　プロトタイプシステムの構成

- CPU: AMD Opteron 6234 2.4GHz×2CPUs
- Motherboard: Supermicro H8DGi
- Memory: 32GB
- Network: GbE×2 and 10GbE×1

4.1 仮想マシン

ハイパーバイザ上には，2 台の Windows と 1 台の Linux の仮想マシンがある．Linux は，Docker 1.10.3 を使用して，2 つのサーバアプリケーションを動作させる．

4.1.1 Windows 仮想マシン

仮想マシンには，ISC BIND 9 または unbound が稼働し，サーバアプリケーションは，SecondDEP で保護される．SecondDEP は，サイバー攻撃で利用されるシェルコードの実行を検知するセキュリティモジュールである．SecondDEP は，データ領域から Windows API を呼び出すのを監視して，その実行を防止する．また，SecondDEP は，サイバー攻撃を検知したら，サーバのプロセスをサスペンドする．これは，検知後に，管理者やセキュリティアナリストがサーバのプロセスを分析できるようにするためである．SecondDEP はサーバのプロセスを停止させるため，攻撃を検知するのと同時に，サーバモニタが DNS サービスの停止を検知する．

Windows はレジストリの `ArpRetryCount` の値により，IP アドレスの重複を検知し初期化するための Gratuitous ARP の送信回数を制御できる．デフォルトでは，`ArpRetryCount` は 3 回に設定されているため，仮想ネットワークアダプタへの接続は少なくとも 3 秒かかるが，これを 0 に設定することにより，OS が仮想ネットワークアダプタに接続するための時間を短縮させるた．

その他の仮想マシンの設定は次の通りである．

- OS: Windows Server 2012
- CPU: 1 CPU × 1 Core
- Memory/Disk: 4GB / 40GB
- SCSI controller: VMware 準仮想化
- Disk provisioning: Thin Provision
- NIC: VMXNET3 enabled RSS

4.1.2 Linux 仮想マシン

仮想マシンは，サーバアプリケーションをアプリケーションレベルで仮想化するため，Docker を使用する．Docker により ISC BIND 9 と unbound を稼働させる．Docker の利点は，ハイパーバイザの仮想マシンと比べてコンテナを迅速に切り替えられ，メモリやストレージなどのリソースの消費が少ないことである．その一方で，欠点は，Docker が開発段階であるため，セキュリティの強度が低い点である．この欠点は，6 章で詳しく述べる．

すべてのサーバアプリケーションは，AppArmor で保護される．AppArmor は，プログラムのネットワークアクセス，ソケットへのアクセス，ファイルに対しての読み込み，書き込み，実行を制限できる．AppArmor は SELinux に比べて制限できる機能が少ないが，簡単な操作や設定でセットアップできる利点がある．プロトタイプシステムでは，AppArmor が，不正な操作を検知したら，ファイルにログを出力し，`inotify` `cron` デーモンによって，管理者やセキュリティアナリストに通知をする．この処理はシェルスクリプトにより実行される．この時点で，サーバアプリケーションは正常な制御を失って，サービスモニタがサービスの停止を検出する．

その他の仮想マシンの構成は次の通りである．

- OS: Ubuntu 15.10（Kernel: 3.19.0）
- CPU: 1 CPU×2 Cores
- Memory/Disk: 4GB / 40GB
- SCSI controller: VMware 準仮想化
- Disk provisioning: Thin Provision
- NIC: VMXNET3

4.2 管理アプライアンス

管理アプライアンスは，VMware vCenter Server 6 standard を使用する．これは，VMware vSphere スイートを管理する中央集中型プラットフォームを提供する．管理アプライアンスは，サービスモニタと仮想マシンチェンジャーを実行する．これらの機能は Bash スクリプトとして実装した．サービスモニタは，アクティブ状態の仮想マシンが DNS サービスを提供しているかを，`nanonslookup` コマンドを使用して確認する．`nanonslookup` コマンドはナノ秒の精度でタイムアウト時間を設定できるように実装したコマンドである．プロトタイプシステムでは，タイムアウト時間を 0.5 秒に設定し，コマンドが連続で 2 回タイムアウトしたとき，サービスが停止したと判断する．このとき，サービスモニタはサービスの停止を仮想マシンチェンジャーに通知する．

また，サービスモニタはスレッドの数が最大値に達していないかを，毎秒，確認する．スレッドの数が最大値に達している場合，サービスは DoS 攻撃を受けていると判断し，仮想マシンの切り替えを仮想マシンチェンジャーに要求する．仮想マシンチェンジャーは，サービスモニタからの通知を受けると，アクティブ状態の仮想マシンとアイドル状態の仮想マシンの情報を取得する．もし，アクティブ状態の仮想マシンが Windows から別の Windows に切り替わる場合と，仮想マシンが Linux から Windows に切り替わる場合は，仮想マシンチェンジャーはアイドル状態の仮想マシンをアクティブ状態へ昇格させる．それ以外の場合は，仮想マシン間を切り替えないで，コンテナを切り替える．

仮想マシン間とコンテナ間の切り替えは，`vmrun` コマンドによって行う．これは，VMware VIX API ライブラリに含まれ，仮想マシンの制御と操作を自動化するためにある．しかし，仮想マシン上でプログラムを実行するための `vmrun` ユーティリティに含まれる，`RunProgramInGuest` コマンドでは，1 回の実行に約 2.6 秒かかる．また，サービスモニタがサービスの停止を検出するのに約 1 秒かかるため，仮想マシン間の切り替えに要する時間は合計 3.6 秒かかる．`RunProgramInGuest` コマンドは，ESXi への接続，

ESXi の認証，仮想マシンのハンドル取得，仮想マシンへのログイン，仮想マシンでのコマンド実行の順に行われる．これらの処理の中で，最初の 2 つの処理は他の vmrun の実行と共通するので, ESXi の認証後のセッションを保持して, これ以降の処理でそのセッションを共有できるように，最初の 2 つの処理とそれ以降の処理を分離したコマンドを VIX API により実装した. 前者の処理を vmrun-server が，後者の処理を vmrun-client が行う．

4.2.1 仮想マシン間の切り替え

仮想マシンチェンジャーは, vmrun-client コマンドにより，次のコマンドをアクティブ状態の仮想マシンで実行する．このコマンドにより，アクティブ状態の仮想マシンの IP アドレスをアイドル状態の仮想マシンに割り当てる．

```
> netsh. exe interface ip set address "Ethernet0 2" sta
   tic IDLE-MACHINE-ADDRESS SUBNETMASK GATEWAY-ADDRESS
```

同時に，仮想マシンチェンジャーは，アイドル状態の仮想マシンの IP アドレスをアクティブ状態の仮想マシンの IP アドレスに変更し，アイドル状態の仮想マシンのサービスを起動する．例えば，vmrun-client により，Windows で次のコマンドを VBS スクリプトで実行する．これにより，ISC BIND 9 が起動する．

```
> netsh. exe interface ip set address "Ethernet0 2" sta
  tic ACTIVE-MACHINE-ADDRESS SUBNETMASK GATEWAY-ADDRES
  S
> C:¥Program Files (x86)¥ISC BIND 9¥bin¥named. exe -f
```

このとき，アイドル状態の仮想マシンはアクティブ状態の仮想マシンに昇格する．また，サスペンドされているマシンは再開され，その仮想マシンはアイドル状態の仮想マシンに昇格する．攻撃されたマシンは，サスペンドされ，攻撃されたことを示すラベルを付加する．

4.2.2 Docker コンテナ間の切り替え

Linux の起動時に，サービスを提供するすべてのコンテナは異なるポート番号を設定し起動する．なお，1 つのコンテナには 53 番ポートを設定し，それ以外のコンテナには 60050 番ポート以降の番号を割り当てる．例えば，ISC BIND 9 を実行するコンテナを起動し，サービスを提供させるには vmrun-client により以下のコマンドを実行する．

```
$ docker run --name bind --security-opt apparmor:usr.sb
   in.bind --publish  ACTIVE-MACHINE-ADDRESS:53:53/udp
   --publish ACTIVE-MACHINE-ADDRESS:53:53/tcp  -td Ubun
   tu-bind-image
$ docker exec bind start-stop-daemon --start --oknodo -
   -quiet -exec /usr/sbin/named --pidfile /var/run/name
   d/named.pid -- -u bind
```

1 つ目のコマンドは, 4 つのオプションを指定して, バックグラウンドで実行する．

- name オプション
  コンテナに名前を付ける
- security-opt オプション
  AppArmor 用のプロファイルを指定する
- publish オプション
  コンテナに割り当てるポート番号をしてする
- td オプション
  コンテナに/bin/bash 用の仮想端末をバックグラウンドで立ち上げる

2 つ目のコマンドは，6 つのオプション指定し，start-stop-daemon コマンドでコンテナ内のプログラムを起動する．

- start オプション
  exec オプションで指定したデーモンを起動する
- oknodo オプション
  終了時にステータス 0 を返す
- quiet オプション
  エラーメッセージ以外のメッセージを表示しない
- pidfile オプション
  引数に指定したファイルが作成されているかを確認する
- --オプション
  start-stop-daemon を終了する
- u オプション
  ISC BIND9 デーモンのオプション

これらのオプションをデーモンのプロセスに設定することにより, bind ユーザが root 権限でできることを制限できる．コンテナ間で切り替える場合，仮想マシンチェンジャーはアクティブ状態のコンテナの停止と削除を行う．同時に，コンテナのホスト 0S のネットワークアドレス変換テーブルに記述されたポート番号を 53 番に変更する．最後に，停止したコンテナには攻撃されたことを示すラベルを付加する．

例えば，アクティブ状態の ISC BIND 9 のコンテナを unbound のコンテナに切り替えるには，vmrun-client により次のコマンドを root 権限で実行する．

```
# docker stop bind && docker rm bind
# iptables -t nat -R DOCKER 2 -d ACTIVE-MACHINE-ADDRESS
  /32 ! -i docker0 -p udp -m udp --dport 53 -j DNAT --to
  -destination 172.17.0.2:53
# iptables -t nat -R DOCKER 3 -d ACTIVE-MACHINE-ADDRESS
  /32 ! -i docker0 -p tcp  -m tcp --dport 53 -j DNAT --t
  o-destination 172.17.0.2:53
```

1 つ目のコマンドは，bind コンテナを停止，削除するコマンドで，2 つ目と 3 つ目のコマンドは，53 番ポートにホスト上のポート番号を変更するコマンドである. 最終的に，すべてのマシンが攻撃された場合，管理者やセキュリティアナリストが攻撃されたマシンにセキュリティ更新プログラムを適用するまで, 仮想マシンチェンジャーは待機する．

4.3 管理者およびセキュリティアナリスト

管理者やセキュリティアナリストは，攻撃されたサーバアプリケーションのプロセスを分析するために，VMware Workstation がインストールされている物理マシンに，停止した仮想マシンのイメージをコピーできる．サーバアプリケーションを再開させることにより，サービスが停止した原因を調べられる. 原因が攻撃であれば, 脆弱性を確認し，適切なセキュリティ更新プログラムを適用した場合は，VMware ESXi 上のサスペンド状態の仮想マシンをキュー

に追加できる．

4.4 フォールトトレラントシステムの実装

本稿では，VMware vSphere Enterprise Plus 6 に含まれる vSphere FT の機能を使用することにより，物理サーバである ESXi に耐故障機能を導入した．これにより，物理サーバまたはネットワークに障害が発生したとき，2 台目の ESXi に仮想マシンを移行できる．

4.4.1 vSphere FT の仕組み

仮想マシンの FT を有効にすると，別の ESXi 上に仮想マシンのコピー（セカンダリマシン）が作成される．プライマリマシンとセカンダリマシンは，FT 用のネットワークを介して常に同期されて，プライマリが動作している ESXi に障害が発生すると，セカンダリマシンに処理が引き継がれる．また，vSphere FT を 3 台の ESXi で構成している場合には，セカンダリマシンがプライマリマシンに昇格するのと同時に，３台目の ESXi にセカンダリマシンが構成される．

4.4.2 vSphere FT の設定

vSphere FT を有効にするには，次の仕様を満たす必要がある．

- 2 台以上の ESXi でクラスタを作成し，vSphere HA の機能を有効にする（FT による保護を最大限に得るには 3 台以上の ESXi が必要である）
- クラスタを構成する各 ESXi 上で，2 つの異なる仮想スイッチ（vMotion 用と FT ログ用）を構成する
- FT ログ用のネットワークは 10Gbps ネットワークを使用する
- vMotion 用と FT ログ用の NIC は，それぞれ異なるサブネットに配置する
- vMotion 用仮想スイッチに割り当てた VM カーネルは vMotion の機能のみ有効にする
- FT ログ用仮想スイッチに割り当てた VM カーネルはフォールトトレランスのログの機能のみ有効にする
- FT で保護する仮想マシンを共有ストレージ（NAS 等）に配置し，スナップショットを削除する

vSphere FT を有効にするには，図 3 に示す構成になるように設定する必要がある．ここで示した構成以外でも FT を有効にできるが，故障発生時に意図しない動作により，フェイルオーバーできないときや，フェイルオーバーしても，FT の同期が機能しなくなるなどの不具合が発生する．

vSphere FT は 3 つのネットワーク（管理トラフィック用ネットワーク、FT ログ用ネットワーク、vMotion トラフィック用ネットワーク）がある．管理トラフィック用ネットワークは，ESXi を vCenter から管理するためのネットワークである．FT ログ用ネットワークは，プライマリとセカンダリをリアルタイムに同期するために使用するネットワークである．vMotion トラフィック用ネットワークは，vMotion での仮想ディスクの移行に使用するネットワークである．

4.4.3 フォールトトレラントシステムの構成

アタックレジリエントサーバをフォールトトレランス設計にするため，仮想マシンに対して vSphere FT を次のコマンドで有効にする．

```
# eval ft.pl -username VCENTER-ID -password VCENTER-PAS
  SWORD -operation on -vmname VIRTUAL-MACHINE-NAME
```

vSphere FT は仮想マシンの状態を同期するため，メモリ，CPU，ネットワーク帯域，ディスクのすべてを大量に消費するため，アクティブ状態の仮想マシンのみにする．仮想マシンの切り替えがあれば，FT も切り替える．

仮想マシンの切り替え直後は，物理サーバに大きな負荷が掛かるため，サービスの応答が安定しない．そのため，サービスモニタを 5 秒間だけ停止させる．また，仮想マシンチェンジャーは，IP アドレスの変更後でも，応答を速く受信できるようにするために，次のコマンドにより ARP の静的エントリを追加した．

```
# arp -s ACTIVE-VIRTUAL-MACHINE-MACADDRESS
```

サービスモニタがサービスの提供を確認したら，その 2 分後にアイドル状態の仮想マシンを起動する．アイドル状態の仮想マシンが起動したら，攻撃によりサービスが停止した仮想マシンの FT を解除する．つづいて，2 分秒後に，アクティブ状態の仮想マシンが FT による保護が開始され，保護が完了した後，2 分秒後に攻撃された仮想マシンがサスペンドされる．

5. パフォーマンステスト

フォールトトレランスが有効になったアタックレジリエントサーバの回復力を評価するため，プロトタイプシステムによるサービスの継続性を評価した．まず，脆弱性を有する DNS サーバアプリケーションを実装し，サーバの切り替え時のダウンタイムを計測した．この DNS サーバアプリケーションは，名前問い合わせの QNAME フィールドを解析するときに，スタックバッファオーバフローを起こす．

サービスの継続性を評価するために，Linux の Docker コンテナ間，Linux の仮想マシンと Windows の仮想マシン間，Windows の仮想マシン間の切り替えに掛かる時間（サービスのダウンタイム）を計測した．

表 1 にフォールトトレランスの無効時と有効時におけるサービスのダウンタイムを計測した結果を示す．計測値は，10 回の試行から得られたダウンタイムの平均値である．フォールトトレランスの無効時の計測結果は，これまでの研究により得られた値である[4]．計測結果から，FT の有効によりダウンタイムが長くなったことがわかる．それぞれの内訳は，コンテナ間では約 1.03 秒，Linux-Windows 間では約 3.09 秒，Windows 間では 1.74 秒であった．これは，サービスモニタが誤検知とみなす確認回数を 1 回から 3 回に引き上げたためである．確認回数を引いあげた理由は，FT 有効時にサービスモニタからアクティブ状態の仮想マシン

に対してサービス稼働確認の応答が連続的に 3 回以上遅れることがあったためである．サービスモニタが誤検知をすると，500 ミリ秒間隔でサービスの稼動状態の確認を再送するため，誤検知の許容回数を 1 回増やすごとにダウンタイムが 500 ミリ秒長くなる．したがって，コンテナ間の切り替えは FT を無効化しているときに比べ，ダウンタイムが約 1 秒長くなった．

仮想マシン間の切り替えが長くなった原因は，仮想マシンの切り替え直後にサービスモニタへの応答が遅れるためである. FT による負荷が想定以上に大きいためと考えられ, サービスモニタを 5 秒間停止することにより，この問題を回避した．そのため，サービスのダウンタイムは切り替わりに掛かった時間に加え, 最大 5 秒長くなる可能性がある．

また，Linux-Windows 間のダウンタイムが他に比べて長い原因は，BIND 9 を実行する Windows の VBS スクリプトにのみ，ネットワークの切り替えとサーバアプリケーションの起動の間に 100 ミリ秒待機するプログラムを追加したためである．これは，Windows 上の BIND 9 の FT を有効にしたとき，サービスの応答が不安定になったためである．

表 1　ダウンタイムの計測結果

	平均値（秒）		最大値（秒）	
	無効	有効	無効	有効
コンテナ間	2.49986	3.53334	2.84766	3.87390
Linux-Windows 間	3.07220	6.16394	3.37173	6.53952
Windows 間	3.02181	4.76351	3.50282	7.71868

6. 考察

6.1 DoS 攻撃に対する回復力

DoS 攻撃は 3 つのタイプに分けられる．1 つ目は，脆弱性を悪用して，サービスを停止させる攻撃である．この種類の脆弱性は，BIND 9 のバージョンで見つかっている（CVE-2012-1667 や CVE-2015-5722 等）．2 つ目のタイプは，SYN フラッド攻撃や HTTP POST DoS 攻撃などサーバのリソースを枯渇させる攻撃である．3 つ目は，サーバに対し大量のリクエストを送信することにより，ネットワーク帯域を消費させる攻撃である．これには, DNS や NTP サービスを使用したリフレクション攻撃が知られている．

プロトタイプシステムは，1 つ目と 2 つ目の攻撃を検知できる．3 つ目の攻撃は，プロトタイプシステムは攻撃者から送信される大量のリクエストメッセージを止めることはできないため，防げない．この攻撃の有効な対策は，サービスを提供するサイトを増やすか，インターネットサービスプロバイダまたはコンテンツデリバリネットワークなどが提供する DoS 攻撃対策用のサービスを利用することである. Docker は, コンテナを Amazon Web Services (AWS), Microsoft Azure，Google Compute Engine など様々なプラットフォームにデプロイできる. そのため Docker コンテナなら容易にサービスサイトを増やせる．また, AWS や Microsoft Azure などには，DoS 攻撃を低減する機能が提供されている[6].

6.2 Docker セキュリティ

Linux にはコンテナ技術に対して，いくつかの脆弱性が発見されたが（たとえば[7]など），これまでにこれらの大部分は改善されてきた[8]．Docker は，Linux カーネル名前空間を使用して，プロセスをホスト OS のプロセス空間から分離させ，ホスト OS のプロセスにアクセスできないようにした．また，Docker は，Linux カーネルのケーパビリティにより，ルート権限を複数に分割して，コンテナに対して必要最低限の権限に制限した．これに加えて，Docker 1.10 は Linux ユーザ名前空間により，コンテナのルートユーザを Docker ホストの一般ユーザに対応づけ，ホスト OS に対してルート権限を行使できないようにした[7]．しかしながら，Docker には，まだ，次のセキュリティ上の課題が残されている．

- Docker デーモンをルート権限で実行する必要がある
- Docker コンテナはホスト OS のカーネルを共有する

これらの弱点は，権限昇格の脆弱性を持つコンテナが乗っ取られた場合，実行中のすべてのコンテナと，ホスト OS が攻撃を受ける可能性がある．これらの課題は，強制アクセス制御（MAC）と Secure computing mode（seccomp）[9] を使用することにより影響を軽減できる．

Linux における MAC の実装としてよく知られているのは SELinux と AppArmor である．AppArmor は，SELinux に比べ容易に制御機能を設定できる利点があるが，制御機能が少ないという欠点がある．例えば，AppArmor は，Metasploit Framework が提供する Meterpreter のシェルコードの実行を防げない．これは，AppArmor が，ファイルのパスに基づいてアクセス制御するためである. それに対して, SELinux は，メモリ上のコード実行をアクセス制御できるため，Meterpreter のシェルコードの実行を防げる．したがって，今後はプロトタイプシステムに SELinux でサーバアプリケーションを保護する方がよい．

6.3 管理運用コスト

アッタクレジリエントサーバは，OS とサーバアプリケーションの多様な組み合わせがサービスの可用性を強化するというコンセプトに基づく．アッタクレジリエントサーバの導入を検討したとき，すべての OS とサーバアプリケーションのセットアップにかかるコストや各仮想マシンのセキュリティ更新等のメンテナンスを含む管理運用コストを無視できない. しかし, アタックレジリエントサーバは, サーバの実装に共通のライブライリを使用していなければ, 2 つの OS と 2 つのサーバアプリケーションの実装の組み合わせに最小化できると考えられる．

6.4 ハイパーバイザの脆弱性

ハイパーバイザの停止を引き起こすような脆弱性が, 複数のハイパーバイザで見つかっている（CVE-20160-1571 や

CVE-2014-8370)．本稿のプロトタイプシステムは，一つのハイパーバイザで構成されているが，本稿のプロトタイプと並行して，2 台の物理サーバで異なる仮想化技術により仮想マシンを稼働させる研究も進めている[10]．今後は，Xen や KVM などを使用することによりハイパーバイザの多様性の向上が必要である．

7. 関連研究

OS の多様化は，Synthetix プロジェクトの Tempo-C スペシャライザーにおいて初めて導入された[11]．Tempo-C スペシャライザーは，OS 内のカーネルモジュールを修正することによって OS を多様化している．しかし，OS の設計や仕様は同じであるため，OS 固有の脆弱性に弱いと考えられる．それに対して，アタックレジリエントサーバは複数の異なる OS を使用するため，OS 固有の脆弱性は他の OS に切り替えることにより解決できる可能性がある．

Tempo-C スペシャライザーに関連したアプローチとして，すでに主要な OS に導入されている ASLR（Address space layout randomization）がある．ASLR は，サーバアプリケーションの実行可能コードを変更せずにモジュールやデータ領域などのアドレス位置を多様化（ランダム化）する点が Tempo-C スペシャライザーと異なる．

フォールトレランス設計の 1 つである N バージョンプログラミングを応用して，ゼロデイ攻撃を検知するハニーポットがある[12]．このハニーポットは，ハイパーバイザで複数の異なるサーバアプリケーションの実装と OS を並行して動作させ，すべてのサーバアプリケーションのレスポンスを比較することにより，攻撃を検知する．複数の実装を利用する点は，アタックレジリエントサーバと共通するが，サーバの目的が異なる．ハニーポットはサイバー攻撃の検知が目的であるのに対して，アタックレジリエントサーバは，サービスの可用性を目的にしている．そのため，アタックレジリエントサーバは，できる限り物理マシンの負荷がかからない設計になっている．

サーバのレジリエンスを高めるアプローチとして，免疫系を模擬して，適応的にサイバー攻撃を学習し，1 度目の攻撃によりサービスは停止することはあっても，2 度目以降の攻撃は未然に防ぐセキュリティモジュールがある[13]．このアプローチの利点は，複数の OS やサーバアプリケーションの実装を必要としないことである．一方，欠点は，誤検知の可能性と脆弱性そのものを解決できないことである．このアプローチは，アタックレジリエントサーバ上のすべての仮想マシンがサイバー攻撃により尽きてしまったときにサービスを提供する最終手段となりうる．

8. おわりに

本稿では，これまで設計・実装を行ってきたアタックレジリエントサーバに対し，耐故障機能である vSphere FT によりフォールトトレランスなアタックレジリエントサーバを実装した．また，プロトタイプシステムに対し，サービスの継続性を計測し，FT の無効時と有効時の計測値を比較評価して，FT のオーバーヘッドを確認した．

プロトタイプシステムは，これまでのダウンタイムを短縮できなかったが，アタックレジリエントサーバをフォールトトレラントシステムにすることができた点は評価できる．また，このダウンタイムが縮小できなかった原因として，物理サーバの性能が vSphere FT が要求する性能に達していなかったという点が挙げられ，その点を改善すれは，より短いダウンタイムを実現できると考える．

今後は，アタックレジリエントサーバのハイパーバイザを多様にし，より短いダウンタイムでシステムの安定した稼働を目指す．

参考文献

[1] Fratric, I.. Runtime prevention of return-oriented programming attacks. University of Zagreb. 2012. https://code.google.com/p/ropguard/

[2] Cheng. Y. et al.. ROPecker: A generic and practical approach for defending against ROP attack. In: Proceedings of the 21st annual network and distributed system security symposium 2014.

[3] Okamoto, T.. SecondDEP: Resilient computing that prevents shellcode execution in cyber-attacks. Procedia Computer Science 60:691-669; 2015

[4] Sano, F. at el.. A cyber attack-resilient server using hybrid virtualization, In: Proc. of International Conference on Knowledge-Based and Intelligent Information and Engineering systems, 2016 (to be appeared).

[5] Sano. F. et al.. A cyber attack-resilient server inspired by diversity. Artificial Life and Robotics Journal, Springer, 10.1007/s10015-016-0286-5, 2016.

[6] Amazon Web Services. AWS Best Practices for DDoS Resiliency. 2015. https://d0.awsstatic.com/whitepapers/DDoS_White_Paper_June2015.pdf

[7] LXC. gentoo linux wiki. https://wiki.gentoo.org/wiki/LXC

[8] Frazelle, J.. Docker Engine 1.10 Security Improvements. 2016, https://blog.docker.com/2016/02/docker-engine-1-10-security/

[9] Hayden, M.. Securing Linux Containers. InfoSec Reading Room, SANS Institute, 2015, https://www.sans.org/reading-room/whitepapers/linux/securing-linux-containers-36142

[10] Winarno, I. at el.. Increasing the diversity of resilient server using multiple virtualization engine technology. Procedia Computer Science, 2016 (To be appeared).

[11] Pu, C.. A specialization toolkit to increase the diversity of operating systems. PhD Thesis. 1996. Portland State University.

[12] Nagy, L. et al.. N-version programming for the detection of zero-day exploits. In: IEEE Topical Conference on Cybersecurity, 2016.

[13] Tarao, M. and Okamoto, T.. Toward an Artificial Immune Server against Cyber Attacks : Enhancement of Protection against DoS attacks. In: Proc. of the 20th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems; 2016 (to be appeared).