Vol.2021-IOT-54 No.8
2021/7/9

異種のソフトウェアを並走する環境における異常検知システムの適用

川村慎太郎[1]　水野修[1]

概要：システムの構成情報を難読化する技術である，Moving Target Defense(MTD)によって，サイバー攻撃を抑止することが可能である．しかし，MTD の効能に対する副作用として，オペレータがシステムの構成情報やサイバー攻撃による被害状況を把握することが難しくなる．
MTD の例として，外部からの通信を処理するソフトウェアを任意のタイミングで変更するシステムを想定する．オペレータはログからサイバー攻撃を検知するが，セッションの処理途中でソフトウェアの切り替えが発生すると，アクセスログからセッションの追跡を行うことが困難になる．そのため，session id をキーにアクセスログからセッションの追跡を行うための処理が必要になる．
本稿では，MTD 上で，アクセスログからセッションを追跡するために必要となる動作について，単一のホストで検証した．検証結果から，複数のサーバを連携させた場合に必要となる機能について検討した．
検討の結果，複数のサーバを連携させて MTD を動作させるためには，パフォーマンスの維持や複数のサーバ間でセッションを共有する仕組みが必要となることが判明した．

キーワード：Moving Target Defense,サイバー攻撃，システムアラート

Consideration of System Quality Assessment on Running Heterogeneous Software Environment

SHINTARO KAWAMURA[1]　OSAMU MIZUNO[1]

Abstract: Moving Target Defense (MTD), a technology to obfuscate system configuration information, can deter cyber attacks. However, one of the side effects of MTD is that it makes it difficult for operators to understand the configuration information of the system and the damage caused by cyber attacks.
We assume that the MTD is applied to a system that changes the software that processes external communication at any given time. Operators can detect cyber-attacks from the logs, but if the software is switched in the middle of processing a session, it will be difficult to track the session from the access log. In this paper, we describe the operation of MTD.
In this paper, we simulate the behavior of MTDs and verify the behavior required to track sessions from access logs on a single host. Based on the verification results, we examined the functions required when multiple servers are linked.
As a result of the study, it was found that in order to run MTD in cooperation with multiple servers, it is necessary to maintain performance and a mechanism to share sessions among multiple servers.

Keywords: Moving Target Defense, Cyber Attack, System Alert

1. はじめに

システムの仮想的な IP アドレスを動的に変更するなど，システムへ到達するためのネットワーク的な経路を中央でコントロールすることで，サイバー攻撃の成功確率を低下させる技術として，Moving Target Defense(MTD)[1][2]が提唱されている．

本項では，MTD の考え方を Web サービスが動作するシステムへ適用する[3][4][5]．Web サービスは一般的に Apache[6]や Nginx[7]，Mysql[8]のような特定の機能を提供するソフトウェアプロダクトで構成される．
このようなシステムにおいて，同じ機能を提供する異なるソフトウェアプロダクトを複数並列に実行することで，それぞれの脆弱性に対するリスクを下げることが期待できる．

このような方針で構築されるシステムを本項では MTD と呼ぶ．

MTD を実行すると，任意のタイミングで動作するソフトウェアが切り替わる．そのため，外部からはソフトウェアのバージョンや種別は難読化し，特定の脆弱性を狙った攻撃は成立しづらくなる．しかし，セッションを処理している途中でソフトウェアの切り替えに遭遇すると，Web サーバと Web ブラウザ間で管理する Web サービスの状態に不整合が発生する恐れがある．これに呼応して，オペレータがアクセスログから処理の流れを分析する場合もアクセスログが分散してしまう恐れがある．そのため，セッションの途中で切り替えが発生した場合に，対象のログを追跡できる仕組みが必要になる．

本稿では，MTD の動作検証を行ったうえで，MTD の処

1 工学院大学大学院
Graduate School of Kogakuin Univ., Shinjuku, Tokyo 163-8677, Japan

理によって，セッションの途中でソフトウェアの切り替えが発生しても，session id によってアクセスログから HTTP リクエストの追跡を行う方法について確認することを目的とする．動作検証は，単一のサーバで実施するが，最終的な目的として，複数のホストによって動作する MTD を想定している．

2 章では関連研究について記載し，3 章では想定する MTD と検証システムについて記載し，4 章では検証内容について記載し，5 章に考察，6 章では今後の展望について述べる．

2. 関連研究

MTD は，ソフトウェアの切り替えによってシステムの構成情報を隠ぺいするが，オペレータがログ分析を容易に行うためには，セッション管理の問題から，デフォルトのアクセスログだけでは不十分である．そのため，MTD としてシステムを構成したうえで，ログ分析を容易に行える仕組みが必要になる．アラートを集約することで，オペレータによるシステムログ分析を効率化する研究や，サイバー攻撃を可視化することで，MTD のようにサイバー攻撃のプロアクティブな対策を行う研究は数多く存在する．本章では，先行研究や既存のログ集約ツールについて紹介する．

サイバー攻撃や不審な通信を検知・収集する手段として，ダークネットと呼ばれる未使用の IP アドレスに対する通信の解析を行う研究[9]や，アラートの形式を統一化することでオペレータの分析作業を効率化する研究[10]や，HIDS(Host-based IDS)を用いて Web アプリケーションへの攻撃元や被害を受けた URL を特定する研究[11]，syslog の総量を統計的に分析しオペレータへ異常を検知する研究[12]などが存在している．

[9]の研究は，サイバー攻撃の傾向を把握することが目的である．しかし，本稿で想定するオペレータは，実際に使用されている IP アドレスや特定のプロセスへの通信をアラートから観測することで分析を行う.そのため，サイバー攻撃の傾向を把握する研究とはシステムを適用する目的が異なる

また，[10]の研究は，オペレータの分析効率化のために，アラートの形式を統一的に表現し，アラートを構造的に提示することで，アラートが継続的に発生しているのか，一時的に発生しているのかの判断を容易にするシステムを実現している．しかし，ログ収集の対象がセキュリティアプライアンスに限定されており，アラートの形式も IP アドレスや Port 番号など，ネットワーク機器に限定された要素である．

そのため，サイバー攻撃を発生させた HTTP リクエストからアプリケーションに生じた被害を特定するためには，出力するアラートの要素に不足がある．

[11]の研究では HIDS を用いて，HTTP リクエストとシステムコールを関連づけることで Web アプリケーションに対するサイバー攻撃の原因と被害内容を特定する方式を提案している．サイバー攻撃の成功時に，アラートを発砲する点が HIDS の利点であるが，アラート量の肥大化が欠点である．

[11]は，HIDS のアラート量をリクエストとシステムコールの関連付けによって抑制する方式であるため，サイバー攻撃検知に対して有効である．しかし，MTD 環境の目的はサイバー攻撃の成功確率を低下させることにある.そのため，攻撃の成功時のみアラートを発砲するのではなく，攻撃の予兆があった場合にアラートを発砲することが求められる．

[12]の研究においては，syslog の総量を分析することで，形式の異なるアラートからでも，機械的に異常を検知できる．

本稿で想定するオペレータは，分析のために syslog ではなく，Web サーバのアクセスログや DB サーバのクエリログを主に使用する．そのため，分析対象のログが異なる．

関連研究としては，以上のような報告が挙げられるが，オペレータの分析作業を効率化するツールも存在する．既存のログ収集ツールとして，fluentd[13]や logstash[14]が存在する．いずれのツールも，ログを一元的に集約し定義ファイルを記述することで，収集したアラートの形式を変換することが可能である．

しかし，いずれのツールも単体では MTD のシステム構成を想定していないため，デフォルトでの適用は困難である．

そのため，MTD に適応するログ収集ツールを実現するためには，MTD の特徴を捉え，特徴に応じたツールを作成することが必要である．

3. MTD の検証システム

3.1 想定する MTD の概要

図 1 に，想定する MTD の概要について示す．図 1 の構成は[5]を参考にしている．

クライアントはフロントエンドの MTD Server にアクセスし，MTD Server がバックエンドの Web server と DB Server にリクエストを振り分ける．各サーバに搭載されるソフトウェアはバージョンや種別が異なるため，潜在的に存在する脆弱性も異なる．さらに，リクエストを処理するサーバの組み合わせに応じて，攻撃すべき脆弱性も異なる．そのため，攻撃者は特定の脆弱性を狙った攻撃が困難になる．

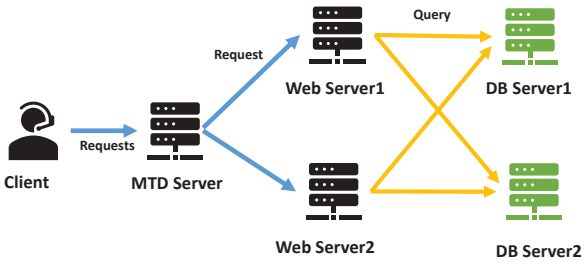

図 1 想定する MTD のシステム構成

3.2 検証システムの処理の全体像

図 1 で示した MTD の動作を検証するために，1 台のサーバで MTD が HTTP リクエストを処理するために必要な機能を実装する．

図 2 に MTD の検証システムの概要図を示し，表 1 に検証システムの構成情報を示す．

図 1 で想定する MTD を検証するために，検証システムの構成として，Web サーバのみを対象に構築を行った．
導入するソフトウェアとして，代表的な Web サーバである，Apache と Nginx を導入する．また，セッションの生成のために，PHP を用いる．パケットフィルタリング機能として，Centos7 系に標準の firewalld を用いる．

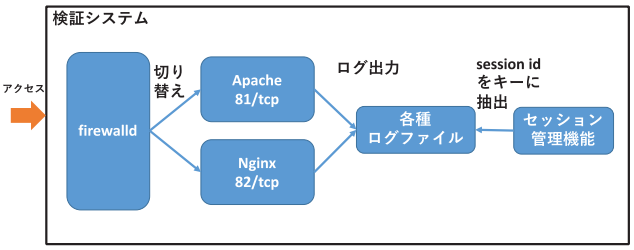

図 2 検証システムの概要図

表 1 検証システムの構成

構成情報	詳細
OS	Centos7.9
PHP	PHP 5.4.16
Web サーバ	Apache 2.4.6
	Nginx 1.20.1

検証システムは 1 台のホストで構成される．外部からのアクセスを，Centos7 の標準のパケットフィルタリング機能である firewalld[15]で受信し，Apache と Nginx に振り分けることで MTD の動作を模擬する．単一のサーバで複数の Web サービスを稼働させるために，Apache は Port81 番で動作させ，Nginx は Port 82 番で動作させる．

firewalld は，受信した HTTP アクセスの宛先 Port を任意の Port へ変換する Masquarade 処理によって，HTTP リクエストを処理するソフトウェアの切り替えを実現する．

Masquarade 処理の設定を任意のタイミングで上書きすることで，HTTP リクエストを処理する Web サーバを切り替える．ソフトウェアの切り替えの動作検証によって，各 Web サーバのアクセスログへ session id を含むログが出力される．

セッション管理機能は，セッションの処理途中で firewalld による切り替えが発生した場合，session id をキーに Web サーバのアクセスログから HTTP リクエストを追跡する機能である．

各機能の検証方法について，以下に記載する．

3.3 セッション管理機能の検証方法

検証の目的は，サーバ側に保存されたセッションファイルから，Web サーバのアクセスログに記録された POST リクエストを追跡する手段を確認することである．

前提として，セッションの処理途中でソフトウェアの切り替えが発生した状況を想定する．ここで，セッションファイルを生成するファイルへ外部からアクセスし，任意のセッション ID を持ったファイルを生成することで検証を行う．セッションファイルは Apache と Nginx でそれぞれ別のディレクトリへ格納する．

図 3 に，検証に用いるスクリプトの動作の概要図を示す．検証に用いるスクリプトは，セッションファイルの格納されたディレクトリの差分を取得し，差分が存在しない場合はセッションファイル名の文字列をファイルへ出力する．ディレクトリの差分が存在しないというは，セッションの途中で Web サーバが切り替わったということである．そこで切り替えの発生したセッションについては，session id をキーにアクセスログから追跡を行うため，session id の情報を別ファイルへ出力する．

以上の処理から，オペレータは，出力された session id の文字列をキーに，アクセスログを抽出する．抽出の結果，追跡すべきログの行数がどの程度軽減されるかを確認する．

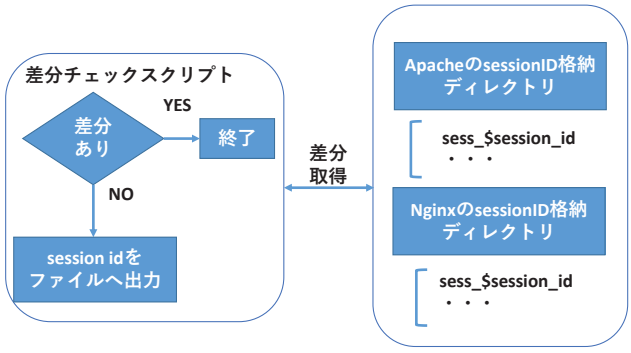

図 3 スクリプトの動作の概要図

3.4 firewalld の切り替え動作の検証

MTD の処理によって，HTTP リクエストを処理する Web サーバが切り替わった際に，通信断がどの程度発生するかを確認するために，firewalld の切り替え動作を検証する．検証の手段としてソフトウェアの切り替えをパケットフィルタリングによって実現する．firewalld の Masquarade 処理によって，検証システムへの Port 80 番宛のアクセス (HTTP アクセス)を Apache 向けの Port 81 番と Nginx 向けの Port 82 番へ変換する．Masquarade 処理による変換処理を上書きすることで，HTTP アクセスが常に片方の Web サーバによって処理される．

切り替えのタイミングは，片方の Masquarade 処理が設定されてから，5 秒間隔で上書きするように設定した．

一連の動作を繰り返すスクリプトを作成し，クライアント側から 0.1 秒間隔で HTTP アクセスが送信されている状態でスクリプトを実行し，検証を行った．

4. 検証実験

4.1 検証による確認ポイント

3.で提案した検証システムの動作確認のために，各機能を単体で動作させる実験を行った．

図 4 に各機能の実験のポイントを示す．

・実験 1 は，sesion id によるログの追跡を行う手段の 1 つとして，セッションファイルの重複チェックによるログ追跡が目的通りに機能することを確認する．

・実験 2 では，ソフトウェアの切り替えを実現する手段の 1 つとして，パケットフィルタリングが目的通りに機能することを確認する．

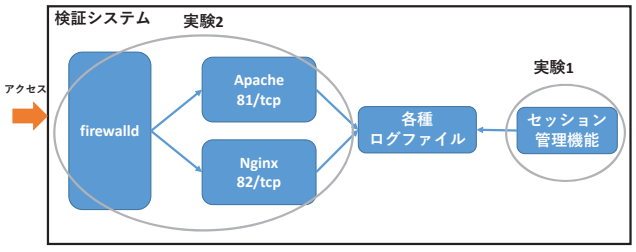

図 4 検証実験のポイント

4.2 実験 1 セッション管理機能の動作確認

Curl[16]コマンドを用いて，任意の session id を持つ HTTP リクエストを検証システムへ送信する．

HTTP リクエストの送信対象のコンテンツは，PHP によって作成されており，送信された session id を取得し，ファイル名に session id を含むファイルを生成する．

実験 1 の結果，Apache と Nginx の両方で，同名のセッションファイルが生成された．しかし，セッションに格納された文字列は，最初にセッションを処理していた Web サーバに残るため，切り替えの発生によって，クライアント側からは，送信したセッションに紐づく情報を取得できない．

また，Web サーバのアクセスログから POST されたセッションの処理を追跡するために，スクリプトによって出力された session id をキーにアクセスログを抽出した．

アクセスログへ出力されたログの全行数と，session id をキーとして抽出を行った場合，オペレータが追跡すべきアクセスログの行数を表 2 に示す．

対象は Apache であり，POST リクエストの詳細な追跡のために，mod_dumpio[17]を有効にしている．

また，測定のために送信した POST リクエストは，curl コマンドで 10 回送信している．

表 2 ログ出力の行数

抽出対象	ログ出力数
全出力数	528 行
session_id のみ	16 行

表 2 より，アクセスログの総出力数は 528 行であったが，session_id をキーにした場合は 16 行が抽出された．

4.3 実験 2 ソフトウェア切り替えの動作確認

通信断の発生時間を確認するために，Masquarade 処理の設定を 5 秒間隔で切り替えるスクリプトによって，検証を行った．外部ホストからの HTTP 通信の模擬のために，ab コマンド[18]を用いた．ab コマンドによる HTTP 通信の送信間隔は 0.1 秒に 1 回である．図 5 に検証結果を示す．

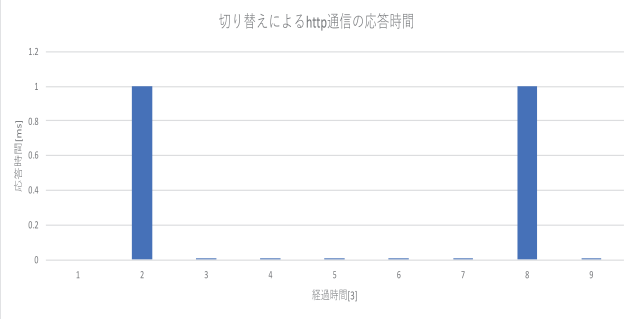

図 5 Masquarade 処理による通信断時間の検証結果

図 5 より，通信が開始してから 2 秒後と 8 秒後に切り替えが発生している．切り替えの発生時間には，1[ms]程度の遅延が発生している．

Firewalld では，処理を上書きするために，サービスをリロードする必要があるため，リロード中に HTTP 通信が発生した場合に，遅延が発生している．

また，図 6 にポートスキャンツールによって，ホストの外部から認識される Web サーバが切り替わる様子を示す．

切り替え前は Apache が動作しているが，切り替え後は Nginx が動作している．

```
root@omizweb102 shell]# nmap -sV -p80 $server_address

Starting Nmap 6.40 ( http://nmap.org ) at 2021-06-03 15:57 JST
Nmap scan report for $server_address
Host is up (0.00030s latency).
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.6 ((CentOS) PHP/5.4.16)

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.57 seconds



[root@omizweb102 shell]# nmap -sV -p80 $server_address

Starting Nmap 6.40 ( http://nmap.org ) at 2021-06-03 15:57 JST
Nmap scan report for $server_address
Host is up (0.00030s latency).
PORT   STATE SERVICE VERSION
80/tcp open  http    nginx 1.20.1

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.54 seconds
```

図 6 スキャンツールによる切り替えの確認

5. 考察

4.2 の検証結果より, オペレータのログ分析の効率化のためには，取得すべきログの出力数を抑えるべきであり，session id をキーに POST リクエストを追跡することは，有効な手段の一つであると思われる．

しかし，特定のセッションのみを追跡するだけでなく．POST された HTTP リクエストのヘッダ情報を詳細に分析する必要がある場合は．抽出のためのキーを増やす必要がある．

その場合，注目すべきログの出力数は増加するため，オペレータが分析すべきログの抽出条件について，検討する必要がある．

以上より, 抽出のキーを session_id に限定することで Web サーバのアクセスログから POST リクエストの追跡が可能であることがわかった．そのため，ソフトウェアの切り替えが発生しても複数の Web サーバでセッションを共有することで，ログ分析が容易に行える．

しかし，今回の検証では，セッションをファイルに出力することで差分をチェックしているので，複数のサーバ間でセッションを共有するためには，ファイルによるセッションのチェックでは不十分である．

そのため，memcached[19] などのツールによって，複数のサーバ間でセッションを共有する仕組みが必要になる．

また，4.3 の図 5 の検証結果より，1[ms]程度の遅延が発生する．1[ms]の遅延は，エンドユーザの体験を大きく損なう値ではないと考えられるため，Masquarade 処理による切り替え処理の影響は大きくない．しかし，複数のサーバを連携させた場合，連携するサーバのシステムパフォーマンスやサーバ間のネットワークの状態に影響を受けることで，より大きな遅延が発生する恐れや，システムの動作に不整合を生じる恐れもある．

さらに，切り替えの条件を時間間隔で設定すると，大量のアクセスを処理する環境においては，セッションの処理中で切り替えが頻繁に発生することになる．その場合，切り替えの際に複数のリクエストで遅延が発生し，ユーザ体験は低下する恐れがある．

図 6 の検証結果より，MTD の処理を用いることで，システムの構成情報を隠蔽し，攻撃者によるスキャンをある程度撹乱することができる．切り替えの方式や稼働するソフトウェアの数に応じて，よりセキュアなシステムが構成できるようになると想定している．

しかし，スキャンツールによる処理の結果が異なる場合，攻撃者は法則を見つけ出そうと，繰り返しスキャンを実施する恐れがある．その場合，システムに対する負荷が増加するなど, 切り替えの法則を分析されてしまう恐れもある．そのため，MTD をよりセキュアに機能させるためには，切り替えの条件について検討する必要がある．

たとえば，[20]のように，稼働している VM への通信量をネットワーク機器で統計的に取得し，統計量が閾値を超えれば，VM のマイグレーションを行い，ルーティングを新規に作成された VM へ向けることでシステムリソースの消費を効率化する方式が考えられる．

あるいは, [1]のように Software Defined Network(SDN)技術を用いて，不正な通信を検知した場合のみ，ルーティングテーブルを更新し，攻撃がシステムへ到達することを防ぐ方法が考えられる．

以上より，単一のホストによる MTD の動作は検証できたが，想定する MTD がパフォーマンスを維持しながら動作するためには，課題が存在する．

そのため，今後の研究課題である，複数サーバの連携による MTD の実現のために，VM のマイグレーションによるシステムパフォーマンスの効率化や，SDN によるルーティングの制御によって，攻撃の到達率を低下させる方式の評価を行う予定である．

6. 今後の展望

単一のホストで MTD の動作を検証することで，ソフトウェアの切り替えと，POST リクエストの分析を行う動作を検証した．

しかし，想定する MTD を動作させるためには，システム全体の構成やパフォーマンスについて検討すべき課題が存在する．

そのため，今後は複数のサーバが連携してリクエストを処理するシステムを構築し，パフォーマンスの測定や，ソフトウェアの切り替え方式について検討する．

参考文献

[1] Panos Kampanakis; Harry Perros; Tsegereda Beyene，SDN-based solutions for Moving Target Defense network protection，Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014，19-19 June

2014
[2] Thomas E. Carroll; Michael Crouse; Errin W. Fulp; Kenneth S. Berenhaut，Analysis of network address shuffling as a moving target defense，2014 IEEE International Conference on Communications (ICC)，10-14 June 2014
[3] Marthony Taguinod; Adam Doupé; Ziming Zhao; Gail-Joon Ahn, Toward a Moving Target Defense for Web Applications，2015 IEEE International Conference on Information Reuse and Integration，13-15 Aug. 2015
[4] Michael Thompson; Marilyne Mendolla; Michael Muggler; Moses Ike，Dynamic Application Rotation Environment for Moving Target Defense，2016 Resilience Week (RWS)，16-18 Aug. 2016
[5] Zhang J., Feng, X., Wang, D. and Ming, L., “Web Service Applying Moving Target Defense,” IEEE International Conference on Data Science in Cyberspace (DSC), 2018.
[6] Apache, https://www.apache.org/ (2021 年 6 月 15 日参照)
[7] Nginx，https://nginx.org/ (2021 年 6 月 15 日参照)
[8] Mysql，https://www.mysql.com/ (2021 年 6 月 15 日参照)
[9] Christian Rossow，“Amplification Hell: Revisiting Network Protocols for DDoS Abuse”，Horst G ¨ ortz Institute for IT-Security, Ruhr University Bochum, Germany
[10] 岩崎 信也，角田 朋 ，関口 悦博， 小西 幸洋， 大鳥 朋哉，薦田 憲久，“ アウトソース型セキュリティセンタにおける インシデント対応迅速化のためのアラート調査支援システム”，情報処理学会論文誌，2019-04-15
[11] 鐘本 楊 青木 一史 三好 潤 小谷 大祐 岡部 寿男，“HIDS アラート調査のための HTTP リクエストとホストイベントの関連付け手法”，情報処理学会論文誌，2020-05-15
[12] 阿部 博 敷田 幹文 篠田 陽一，“イベントネットワークにおける syslog を用いた異常検知手法の提案と実データを用いた評価”，情報処理学会論文誌，2018-03-15
[13] fluentd https://www.fluentd.org/ (2021 年 6 月 15 日参照)
[14] logstatsh https://www.elastic.co/jp/logstash (2021 年 6 月 15 日参照)
[15] firewalld https://firewalld.org/ (2021 年 6 月 15 日参照)
[16] curl https://curl.se/ (2021 年 6 月 15 日参照)
[17] mod_dumio https://httpd.apache.org/docs/2.4/mod/mod_dumpio.html (2021 年 6 月 15 日参照)
[18] Apache bentch https://httpd.apache.org/docs/2.4/programs/ab.html (2021 年 6 月 15 日参照)
[19] Memcached https://memcached.org/ (2021 年 6 月 15 日参照)
[20] Saptarshi Debroy; Prasad Calyam; Minh Nguyen; Allen Stage; Vladimir Georgiev，Frequency-minimal moving target defense using software-defined networking，2016 International Conference on Computing, Networking and Communications (ICNC)，15-18 Feb. 2016