L-032

複数台のおとりマシンによる HTTP-GET Flood 攻撃対策
HTTP-GET Flood Prevention by Using Multiple Decoy Machines

吉田 祥真 †1　三上 烈史 †1　小林 良太郎 †1　金岡 晃 †2　加藤 雅彦 †2
Shoma Yoshida, Tsuyoshi Mikami, Ryotaro Kobayashi, Akira Kanaoka, Masahiko Kato

1. はじめに

現在，ウェブサービスは企業の情報公開やオンラインサービス，電子商取引など，様々な用途で広く一般的に利用され，日常生活の中で必要不可欠となりつつある．そのため，ウェブサービスが停止した際の社会的影響も大きくなっている．その中で，ウェブサービスへ数多くの攻撃が行われ問題となっている．サービス妨害（**DoS**：Denial of Service）攻撃は，サーバや回線といったネットワーク上のリソースに対して攻撃を行い，攻撃対象のシステムのサービス提供を不能にする攻撃である．

DoS 攻撃は, 大きく以下の種類に分けることができる．

- システムの脆弱性を利用した攻撃
- ネットワークリソース浪費型
- サーバリソース浪費型

システムの脆弱性を利用した攻撃は，システムやアプリケーションの脆弱性を利用し，サービス提供を妨害する攻撃である．ネットワークリソース浪費型は，サイズの大きいパケットを大量に送りつけるなどして，ネットワーク帯域を消費して接続しにくくすることにより，サービス提供を妨害する攻撃である．サーバリソース浪費型は，攻撃対象に大量のリクエストの発行やコネクションの確立を行うことで，サーバの処理能力を超える要求でサーバに負荷をかけ，サービス提供を妨害する攻撃である．

本研究では, サーバリソース浪費型 DoS 攻撃の一種である HTTP-GET Flood 攻撃に焦点を当てる．HTTP-GET Flood 攻撃は，HTTP を使用して大量の GET リクエストを送ることでサーバに負荷をかける攻撃である．攻撃方法としては，ウェブブラウザでの F5 攻撃や，ツールを使用する方法がある．なお，本稿でいう攻撃は，人によるオペレーションを前提としており，Bot による攻撃は考慮しない．

DoS 攻撃への一般的な対策としては，ファイアウォールやルータによるフィルタリングや帯域幅制限などがある [1]．しかし，サーバ側で攻撃をブロックしていることが攻撃者に気づかれると，攻撃者は別のサーバに攻撃を移したり，手段を変更して対策を回避するように攻撃を行う可能性がある [2]．

先行研究では，過去に上記の問題を考慮した対策を提案している [3]．本稿では，この手法を「従来手法」とする．従来手法では，攻撃者には攻撃が成功しているかのように見せながら，通常利用者には正常なサービスを提供可能な状態を実現させている．しかし，この手法では，ツールを用いた強力な攻撃を行う攻撃者が現れた場合，攻撃が成功しているかのように見せる手法が乱れてしまう．

そこで本研究では，従来手法を改良し，ツールを用いた強力な攻撃者が存在する場合においても，攻撃が成功しているかのように見せる制御手法を提案する．また，提案手法の実装を行い，評価実験により提案手法の有効性を示す．

以下，２章では従来手法を説明し，３章で提案機構の構成や動作を述べる．４章では実装について述べ，５章では提案手法の評価を行い，最後に６章で本論文をまとめる．

2. 従来手法

従来手法では，以下の点を考慮して HTTP-GET Flood 攻撃対策を行っている [3]．

- 攻撃者に対策したことが気づかれにくい
- 通常利用者に影響を与えにくい

従来手法では，通常利用者に影響を与えないため，攻撃者が利用するマシン（おとりマシン）と通常利用者が利用するマシン（通常マシン）にコンピュータ資源を分離する．そして，通常利用者のアクセスは通常マシンへ，攻撃者のアクセスはおとりマシンへ誘導する．一時的な攻撃のために物理マシンを複数台用意することは現実的ではないため，仮想マシンを用いて複数のウェブサーバを構築する．また，攻撃者のおとりマシンへの誘導は，管理ドメインでの NAT を用いて行う．

おとりマシンは通常マシンと同一のウェブサービスを提供する．おとりマシンでは攻撃が成功しているように見せるため，リソースを制御して通信処理に時間が掛かる状況を作り出す．リソース制御では，極端なエラー発生は不自然であるため，エラー発生率が管理ドメインで設定した目標値に近づくように制御を行う．

†1 豊橋技術科学大学大学院工学研究科
Graduate School of Engineering, Toyohashi University of Technology
†2 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, Tsukuba University

従来手法では，攻撃者のエラー発生率をおとりマシンのリソース制御によって制御できることを評価実験で確認している．しかし，ツールを用いたような強力な攻撃者が現れた場合，おとりマシンのリソースは不足し，エラー発生率は極端に高くなる可能性がある．おとりマシンがこの状態のときは，攻撃量の少ない攻撃者や新たに現れた攻撃者に対しても極端にエラーが発生してしまう．このような攻撃者にとってはこの状況は不自然であるため，攻撃者は自分の攻撃がブロックされていると思い込む可能性がある．

3. 提案手法

3.1 メインアイデア

攻撃者が自分の攻撃がブロックされていると判断する状況を考察すると，以下の二つの場合が考えられる．

- 自分の攻撃によるウェブサーバへの影響が全くない
- 自分ではウェブサーバがダウンするほどの攻撃を行っていないのに極端にエラーが発生する

上記のことを踏まえると，以下の理由によりツール攻撃者に対しては必ずしもエラー発生率を制御する必要は無いと思われる．

理由 1 F5 攻撃者はブラウザ上でエラーを確認している可能性が高いが，ツール攻撃者は常に確認しているとは限らない

理由 2 ツール攻撃は強力なので，極端なエラー発生は不自然ではない

提案手法では，ツール攻撃者のような強力な攻撃者をエラー発生率の制御対象から除外することで，おとりマシンのリソース不足によるエラー発生率の制御の乱れを抑えている．

提案手法ではツール攻撃者は強攻撃者，F5 攻撃者は弱攻撃者とする．おとりマシンは強攻撃者用と弱攻撃者用に 2 台用意し，強攻撃者と弱攻撃者のアクセスはそれぞれのおとりマシンへ誘導する．上記の理由から，強攻撃者へはエラー発生率を制御する必要がないため，リソース制御は弱攻撃者用おとりマシンのみ行う．このように，強攻撃者に使用するリソースと弱攻撃者に使用するリソースを分類することで，弱攻撃者へのエラー発生率を保つことを容易にする．

3.2 構成と動作

本提案手法のシステム構成を図 1 に示す．図 1 では，ホストマシン 1 台に仮想マシンを 4 台作成し，WebServer1〜4 として動作させる．4 台の仮想マシンのうち，WebServer1，2 を通常マシンとし，WebServer3 を弱攻撃者用おとりマシン，WebServer4 を強攻撃者用おとりマシンとして動作させる．すべての仮想マシンは同様のサービスを提供する．

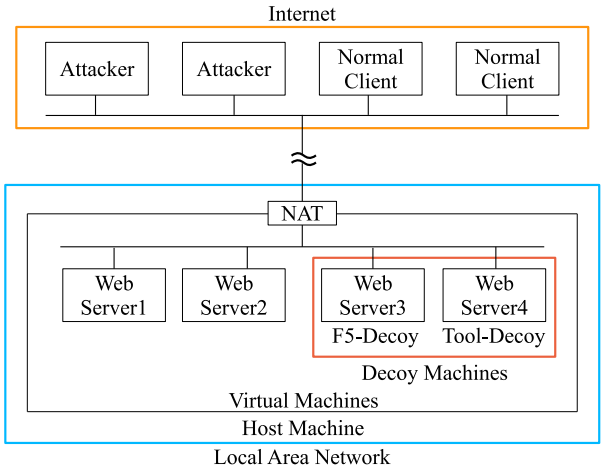

図 1 システム構成

また，管理ドメインでの NAT を行うことによって，通常利用者は通常マシンへ，弱攻撃者は弱攻撃者用おとりマシンへ，強攻撃者は強攻撃者用おとりマシンへ転送される．

4. 実装

本章では，提案手法を実現するための実装の説明をする．

4.1 Xen

本研究では仮想化ソフトウェアは Xen を使用し，準仮想化環境を使用する．Xen は，ハイパーバイザ型の仮想化システムであり，ハードウェアの制御や管理を行う特権を持つドメインである Domain0 と，その他のオペレーティングシステムが動作し特権を持たないドメインである DomainU から成る． 3.2 節における管理ドメインが Domain0，管理ドメイン以下の VM が DomainU に当たる．

4.2 リソース制御プログラム

Domain0 では，リソース制御プログラムを実行する．リソース制御プログラムは，弱攻撃者のウェブページ取得のエラー発生率を目標値に近づけるために，弱攻撃者用おとりマシンのリソースを動的に制御するプログラムである．プログラムでは，リソース制御を行うための指標として，Domain0 から弱攻撃者用おとりマシンに定期的に HTTP-GET リクエストを発行し，エラー発生率を算出する．HTTP-GET リクエストは 1 秒間に 1 回発行し，現在までの応答結果からエラー発生率を算出する．算出したエラー発生率を設定した目標値と比較し，リソースの増減を決定する．

本研究では，リソース制御は CPU 使用率を制御することで行う．現在の Xen のデフォルトスケジューラは credit スケジューラである．credit スケジューラ

は，各ドメインに credit と呼ばれる値を割り当て，その値に従って CPU 時間が公平に分配される．credit スケジューラは CAP と weight の 2 つの値をもとにスケジューリングを行う．CAP は，物理 CPU1 個を 100 とした絶対的な値（ただし 0 は無制限を意味する）であり，一方で weight は，他のドメインに対しての相対的な値である．リソースは，弱攻撃者用おとりマシンの CAP 値を変更することで増減させる．プログラムでは，算出したエラー発生率が目標値より高い場合は CAP 値を増加させ，低い場合は CAP 値を減少させる．CAP 値の増減は以下のように行う．

$$0 \longleftrightarrow 30 \longleftrightarrow 28 \longleftrightarrow \cdots \longleftrightarrow 4 \longleftrightarrow 2 \longleftrightarrow 1$$

なお，全てのウェブサーバとする VM の weight 値は同じ値にしている．

5. 評価

強攻撃者のアクセスを強攻撃者用おとりマシンに集約することにより，リソース制御の乱れが抑えられているかどうか確認するため，実装した提案手法の評価を行う．本章ではまず評価環境について述べ，次に評価方法・結果について述べる．

5.1 評価環境

評価環境を図 2 に，各マシンの構成を表 1〜4 に示す．1 台の物理マシン上にウェブサーバとして動作させる仮想マシンを 4 台用意し，VM1〜VM4 とする．VM1〜VM4 はそれぞれ WebServer1〜WebServer4 として動作させる．ウェブサーバはホームサーバ程度の規模を想定する．WebServer1〜WebServer4 は，同一のウェブサービスを提供する．VM1，VM2 は通常マシン，VM3 は弱攻撃者用おとりマシン，VM4 は強攻撃者用おとりマシンとして稼働する．

攻撃を行うマシンは 20 台とする．具体的には，4 台の物理マシン上の仮想マシン各 5 台，合計 20 台の仮想マシンを攻撃マシンとする．このうち，15 台を弱攻撃者，5 台を強攻撃者として動作させる．攻撃には負荷テストツールである JMeter を使用し，HTTP-GET Flood 攻撃を行わせる．なお，ここでは弱攻撃者には秒間 20 回以下のアクセスを，強攻撃者にはそれ以上のアクセスを行わせる．これは，1 台のマシンに F5 キーを押したままにしてリクエストを発行させ続けたところ，1 秒間に約 20 回のアクセスログが記録されていたことが理由である．

評価尺度は，HTTP-GET リクエストを発行したときのエラー発生率とする．測定を行うには，測定を行わせるマシンを 4 台使用する．具体的には，1 台の物理マシン上の仮想マシン 4 台を測定マシンとする．測定には，攻撃マシンと同様に JMeter を使用する．このとき各測定マシンからのリクエストを，Domain0 で NAT を行うことにより，それぞれ異なるウェブサーバへ転送する．

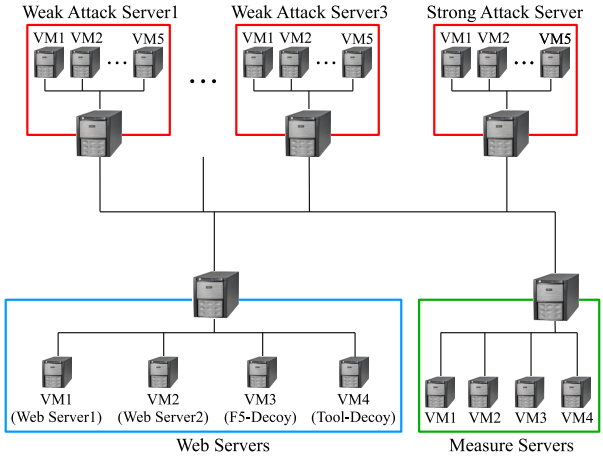

図 2 実験環境

表 1 ウェブサーバのホストマシン

OS	openSUSE 11.3 64bit(2.6.34.7-0.5-xen)
CPU	Intel Xeon L3426(1.87GHz)
Memory	4GByte
仮想計算機	Xen 4.0.0_21091_06_0.1

表 2 攻撃サーバのホストマシン

OS	openSUSE 11.3 64bit(2.6.34.12-xen)
CPU	AMD Phenom II X6 1055T(2.8GHz)
Memory	8GByte
仮想計算機	Xen 4.0.0_21091_05_6.6

表 3 ウェブサーバ

OS	openSUSE 11.3 64bit(2.6.34.7-0.3)
CPU	1Unit
Memory	512MByte
Web Server	Apache 2.2.15

表 4 攻撃サーバ

OS	openSUSE 11.3 64bit(2.6.34.7-0.3)
CPU	1Unit
Memory	512MByte
負荷テストツール	jakarta-jmeter 2.4 r961953

5.2 評価実験

攻撃者の中に強攻撃者が存在する場合，従来手法ではリソース制御が乱れることを確認するために，従来手法を用いた測定を行う．また，提案手法の有効性を確認するために，提案手法を用いて同様の測定を行う．このように，従来手法を用いた場合と提案手法を用いた場合で測定を行い，その評価結果を比較する．

5.2.1 評価方法

提案手法では，Domain0 で，弱攻撃者からのアクセスは弱攻撃者用おとりマシンへ，強攻撃者からのアクセスは強攻撃者用おとりマシンへ NAT を行うように予め登録する．一方，従来手法では，弱攻撃者からのアクセ

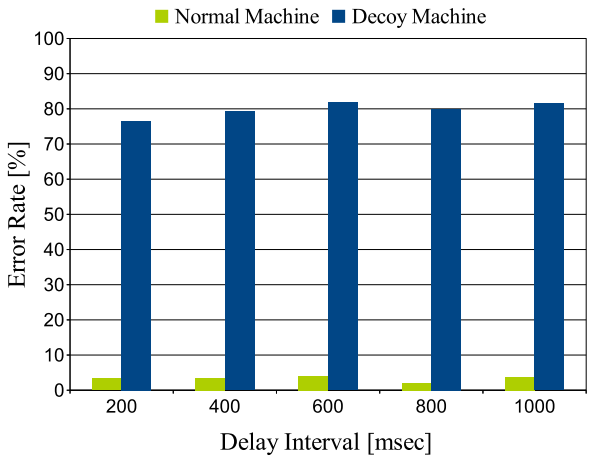

図 3　従来手法で強攻撃者が存在するの場合の各ウェブサーバのエラー発生率

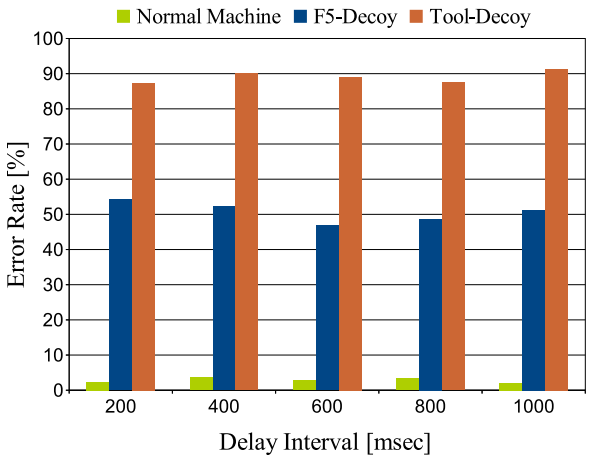

図 4　提案手法で強攻撃者が存在するの場合の各ウェブサーバのエラー発生率

スと同様に強攻撃者からのアクセスも弱攻撃者用おとりマシンへ NAT を行うように予め登録する．ウェブサーバとする VM の CAP 値は，従来手法・提案手法のどちらの場合も通常マシン，弱攻撃者用おとりマシンは 0 と設定する．また，強攻撃者用おとりマシンの CAP 値は，従来手法・提案手法のどちらの場合もリソースを節約するために 1 と設定する．従来手法の場合，強攻撃者用おとりマシンへの NAT は行わないが，提案手法と使用するリソースを等しくするため強攻撃者用おとりマシンを起動している．リソース制御プログラムでは，エラー発生率の目標値を 50% として設定する．

攻撃のパラメータとして，攻撃マシンは弱攻撃者 15 台，強攻撃者 5 台の計 20 台を使用する．JMeter が一度に発行するスレッド数は，弱攻撃者が 4，強攻撃者が 50 とする．測定は，攻撃マシンの JMeter のパラメータであるスレッド遅延間隔 [msec] を弱攻撃者のみ 1000，800，600，400，200 と変更した 5 パターンをそれぞれ 10 分間行う．スレッド遅延間隔とは，JMeter が次のスレッドを発行するまでの時間のことである．強攻撃者のスレッド遅延間隔は，どのパターンにおいても 200 に固定する．このパラメータでは，1 台の弱攻撃者からの秒間アクセスは 4〜20 回，1 台の強攻撃者からの秒間アクセスは 250 回に相当する．

測定マシンは，各ウェブサーバへ 1 秒間に 1 回の HTTP-GET リクエストを発行し，応答結果を記録する．この応答結果から，エラー発生率を算出する．

5.2.2　評価結果

従来手法での測定結果を図 3 に，提案手法での測定結果を図 4 に示す．なお，どちらの場合も通常マシンは 2 台存在するが，結果が酷似しているため，ここでは 2 台の結果の平均値を示している．図 3 の結果では，通常マシンのエラー発生はほとんど見られないが，おとりマシンのエラー発生率は目標値である 50% より高くなっていることが分かる．このことから，従来手法では強攻撃者が存在する場合，おとりマシンのエラー発生率の上昇を抑えることができないことが分かる．

図 4 の結果では，通常マシンのエラー発生はほとんど見られず，弱攻撃者用おとりマシンのエラー発生率も目標値である 50% に近づけることができている．これは，強攻撃者用おとりマシンが強攻撃者のアクセスを集約しているため，弱攻撃者用おとりマシンへの負荷を下げているためである．そのため，強攻撃者用おとりマシンのエラー発生率は高くなっている．

以上の結果から，提案手法では，ツール攻撃を行う強攻撃者が存在する場合においても，F5 攻撃者のアクセスを集約する弱攻撃者用おとりマシンのエラー発生率は制御できることが分かる．

6.　まとめ

攻撃者に攻撃が成功しているかのように見せる HTTP-GET Flood 攻撃対策を，従来手法を改良して提案した．本提案手法により，強力な攻撃者が存在する場合においても，攻撃が成功しているかのように見せる制御を可能とした．評価の結果，従来手法で強力な攻撃者が存在しない場合と同様の効果を，提案手法では強力な攻撃者が存在する場合においても得られた．

参考文献

[1] @police, “ DoS/DDoS 対策について, ” http://www.npa.go.jp/cyberpolice/server/rd_env/pdf/DDoS_Inspection.pdf, 2003.

[2] 情報処理推進機構, “「サービス妨害攻撃の対策等調査」報告書, ”http://www.ipa.go.jp/security/fy22/reports/isec-dos/index.html, 2010.

[3] 高橋朝英, “ 仮想計算機のリソース制御による HTTP-GET Flood 攻撃対策, ” 電子情報通信学会論文誌, Vol.J94-D, No.12, pp.2058-2068, 2011.