

# シグネチャ型 IDS のシグネチャ に基づいた ペネトレーションテストツールの実装

LOW LIP NGHEE†  
Low Lip Nghee

川橋 裕‡  
Kawahashi Yutaka

## 1. はじめに

近年、サイバー攻撃の脅威が顕著化し、情報セキュリティインシデントによる被害が拡大している。そこで、セキュリティ対策の一つとして、IDS(Intrusion Detection System)が利用されている。

シグネチャ型 IDS とはシグネチャと呼ばれる既知のインシデントに対応したパターンを元に、異常なイベントを検知できるシステムである。しかし、導入した IDS が攻撃を検知できるかは攻撃が実際に発生するまで確認できないため、その有効性を保証するための検証が不可欠である。

IDS を検証するツールとして、シグネチャに対応した不正通信のパケットを自動的に生成できる **Sneeze** が挙げられる。**Sneeze** を用いた検証には、シグネチャごとにツールを用意する必要がないというメリットがある。

しかし欠点として、TCP コネクションの状態を分析できる IDS の検証では、**Sneeze** で作成した TCP パケットが検知されない。また、現存手法ではサーバソフトウェアの導入が必要という問題点もある。

そこで本研究では、TCP コネクションの制御、および TCP 通信の待ち受けが可能なペネトレーションテストツールを実装した。

## 2. 既存手法

IDS の検証で用いられるペネトレーションテストツールとしてエクスプロイトコードが挙げられる。一つのエクスプロイトコードは基本的に一つの脆弱性しか利用できない。そのため、エクスプロイトコードを用いた IDS の検証では、シグネチャごとにエクスプロイトコードを収集・操作しなければならないという問題点がある。

一方、IDS のシグネチャを元に自動的にパケットを生成する手法では、IDS のシグネチャに定義されたインシデントごとにツールを用意する必要がないというメリットがある。

この手法を実装したツールの代表例として、**Sneeze** が挙げられる。

しかし、**Sneeze** では TCP 通信のコネクションを管理する機能が実装されてないため、生成したパケットが TCP コネクションの状態を分析できる IDS に検知されない問題が発生する。また、生成したパケットを受け取るために、通信先ではインシデントごとにサーバソフトウェアを導入しなければならないという共通な問題がある。

## 3. 研究目的

本研究では、TCP コネクションの制御、および TCP 通信の待ち受けが可能なペネトレーションテストツールを実装することを目的とする。それにより、不正な TCP 通信のパケットを送信する前に、サーバとの TCP コネクションを確立するため、生成したパケットが IDS に検知されない問題を解決する。また、サーバソフトウェアを通さず、直接ネットワークソケットを操作することで、TCP 通信の接続を待ち受ける。従って、本研究の提案ツールを用いた IDS の検証では、サーバソフトウェアの導入は不要である。

## 4. 提案手法

本研究の提案ツールは TCP コネクションを確立した上で、生成したパケットを送信する。そのため、TCP 通信の状態を分析できる IDS を検証する際に、TCP 通信のパケットが検知されないという **Sneeze** の問題点を解決できる。

TCP 通信ではサーバにインストールされたサーバソフトウェアがネットワークソケットを通信ポートと紐づけた上に、「LISTEN」という待ち受ける状態に設定することで、クライアントからの接続を待ち受ける。本研究ではサーバソフトウェアを用いず、ネットワークソケットを操作できるモジュールを導入することで、柔軟にネットワークソケットを管理できるようにした。

† 和歌山大学大学院 システム工学研究科

‡ 和歌山大学 学術情報センター

提案ツールのパケット生成機能についてはSneezeと同様、IDSのシグネチャを元にパケットの生成を行う。しかし、事前にパケットの生成とは無関係な記述を除去し、シグネチャの整形を行う必要がある。

また、提案ツールはPythonで実装し、Scapyと呼ばれるツールでパケットの生成・送信を行う。パケットの受信機能にはPythonのsocketモジュールが用いられている。そして、パケットの生成に使用されているIDSのシグネチャはバージョン29150のSnortのルールである。

## 5. 実験・評価

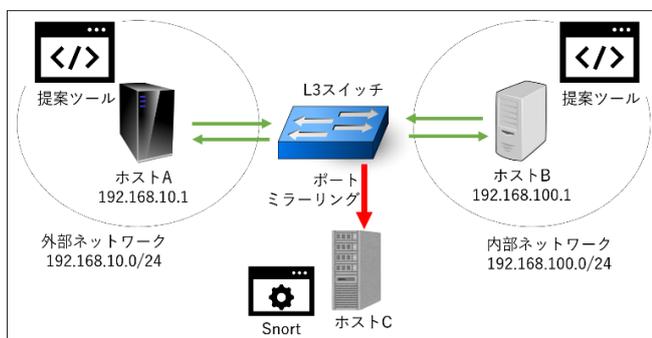


図 1 : 実験環境

評価実験として提案ツールを用いたIDSの検証を、図 1 に示した実験環境で行った。外部と内部ネットワークに設置したホストAとBで同時に提案ツールを実行し、パケットを送受信した。生成されたパケットはすべてホストCに転送し、Snortによる検知を行った。

本実験ではSyn Flood攻撃、Slow HTTP DoS攻撃、およびSQLインジェクション攻撃に対応したシグネチャを元に、ツールでパケットを生成した。また、比較実験としてエクスプロイトコードによるSnortの検証も行った。

```
01/26-22:24:34.138710 [**] [1:100002:1] SYN Flood Attack [**] [Priority: 0] [TCP] 192.168.10.1:47704 -> 192.168.100.1:80

01/26-22:24:34.109505 [**] [1:23952:4] MALWARE-TOOLS Tors Hammer slow post flood attempt [**] [Classification: Detection of a Denial of Service Attack] [Priority: 2] [TCP] 192.168.10.1:47686 -> 192.168.100.1:80

01/26-21:40:14.202377 [**] [1:30040:4] SQL_1 = 1 - possible sql injection attempt [**] [Classification: Web Application Attack] [Priority: 1] [TCP] 192.168.10.1:47246 -> 192.168.100.1:80
```

図 2 : Snortのログ

```
18:24:40.262201 IP 192.168.10.1.62392 > 192.168.100.1.http: Flags [S], seq 100, win 8192, length 0
18:24:40.262250 IP 192.168.100.1.http > 192.168.10.1.62392: Flags [S.], seq 1200261557, ack 101, win 29200, options [mss 1460], length 0
18:24:40.269461 IP 192.168.10.1.62392 > 192.168.100.1.http: Flags [], ack 1, win 8192, length 0
```

図 3 : tcpdumpの出力

```
tcp 0 0 192.168.100.1:80 0.0.0.0:* LISTEN
```

図 4 : netstatの出力

図 2, 3, 4 は実験結果を示している。図 2 は提案ツールを用いた検証の際にSnortが出力した検知結果であり、生成したパケットがSnortに正しく検知されることを確認できた。図 3 はtcpdumpの出力結果であり、TCP通信のパケットが送信される前に、あらかじめTCPコネクションが確立されたことを確認できた。図 4 はnetstatの出力結果であり、サーバ側のホストでネットワークソケットはTCP通信の80番ポートに結び付けられ、「LISTEN」という状態にあったため、TCP通信を待ち受けていたことを確認できた。

評価実験と比較実験で同じ結果が確認できたため、提案ツールによるIDSの検証が有用であると考えられる。

## 6. 今後の課題

Snort以外にもSuricata、OSSECなどのシグネチャ型IDSが存在しており、各IDSのシグネチャの書き方が異なる。本研究の提案ツールはSnortのシグネチャを元に実装したため、現在はSnort以外のIDSに対応していない。今後は逐次に対応する必要がある。

また、提案ツールの実装に使用されたSnortのシグネチャのバージョンは2.9であり、ベータ版であるバージョン3.0に対応していない。バージョン3.0が正式にリリースされてから、対応できるようにツールを修正する必要がある。

## 参考文献

[1] NIST アメリカ国立標準技術研究所, “侵入検知および侵入防止システム (IDPS) に関するガイド” (2007)

<https://www.ipa.go.jp/files/000025364.pdf>

[2] “Sneeze” (2001)

<https://securiteam.com/tools/5DP0T0AB5G/>