L-025

ns-3 と DNS サーバによる DNS リフレクタ攻撃エミュレーション

DNS Reflector Attacks emulation by using ns-3 and DNS server

申 河英† 鄭 俊俊† 齋藤 彰一‡ 毛利 公一† 瀧本 栄二†
Hayeong Shin Junjun Zheng Saito Shoichi Koichi Mouri Eiji Takimoto

1. はじめに

近年，インターネットの使用の増加によってネットワーク攻撃による被害が広がっている[1]．これにともないネットワークセキュリティ技術者の育成とセキュリティ研究が喫緊の課題となっている．この課題に対する 1 つの手法としてネットワークシミュレータの活用が挙げられる．ネットワークシミュレータを利用すれば低コストで多様なネットワーク環境を構築し，シミュレーションすることが可能である．

既存のネットワークシミュレータは，プロトコルの開発や通信の性能評価に重点を置いている．そのため，高速にシミュレーションすることを目的とした抽象化が行われている．しかし，多様な攻撃手法をシミュレートするためには実際のネットワーク動作を可能な限り再現する必要がある．そのため，既存のシミュレータにはセキュリティ的に考えなければならないことを全部簡略化されていると言える．

以上の背景から我々は，セキュリティに関する教育や研究での活用を想定したネットワークセキュリティシミュレータ(以下，NSS と記す)[2]を研究・開発している．

NSS の開発にはネットワークセキュリティに関連する要素を明らかにし，それらのモデル化する必要がある．その方法として実際のネットワーク攻撃をエミュレートし，その結果を利用することが有効であると考えられる．本稿では，DDoS 攻撃で多用される DNS リフレクタ攻撃を対象として検討を行う．DNS はインターネットに必須な要素である．しかし，Bind を代表とする DNS サーバは処理が複雑であるため，シミュレータに直接移植することは現実的でない．NSS を実現するためには DNS リフレクタ攻撃に関連する処理に絞ることが重要である．そこで，既存のシミュレータと Bind を用いて DNS リフレクタ攻撃をエミュレートし，その結果に基づいて NSS に反映することとした．

本稿では，シミュレータ側で DNS リフレクタ攻撃をシミュレーションするシナリオを作成し，そのシミュレーションシナリオで DNS サーバの動作を必要とする処理があれば，シミュレータ外の DNS サーバへクエリを送信する．そして，DNS サーバからクエリの内容についてレスポンスを返してシミュレータに送信する．シミュレータは DNS サーバから受信した結果を処理し，シミュレーションを継続する方法を提案する．提案手法は実際の DNS サーバを用いてシミュレーションをすることを可能にし，実際の DNS リフレクタ攻撃データをシミュレーションで得ることが可能だと考えられる．この結果を用いて実際攻撃と比較し，NSS に必要な要素を抽出する．

† 立命館大学大学 Ritsumeikan University
‡ 名古屋工業大学 Nagoya Institute of Technology

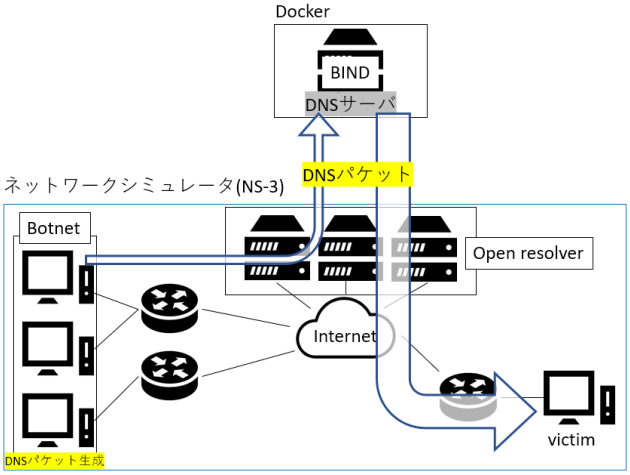

図 1 DNS リフレクタ攻撃シミュレーション環境構成

本稿では，2 章で関連研究について述べ，3 章では提案手法のシステム構成について述べる．4 章ではその実装について述べ，5 章では実装の結果を報告し，6 章で考察し，7 章でまとめる．

2. 関連研究

ns-3[3]は最も一般的に使われるオープンソース離散事象ネットワークシミュレータであり，主に通信の性能評価とプロトコルの開発を主目的とする．本研究の目的を実現するためには，DNS サーバや IP スプーフィングなど課題も多い．

GNS3[4]は，GUI でシミュレーションシナリオを作成でき，仮想環境上でルータを動作させることができる．しかし，ネットワークセキュリティに対するシミュレーションには特化されてない．

NeSSi²[5]はネットワークセキュリティに特化されたシミュレータである．他のシミュレータに比べ忠実にシミュレーションの再現が可能である．特に，特殊の攻撃シナリオを支援しているが，ここで攻撃の特性に合うサーバなどを実現をしていなく，一般的なサーバだけでシミュレーションを行っていることが本手法とは異なる．

NETA[6]は OMNET++で使われるネットワーク攻撃シミュレーションフレームワークである．NETA は本手法の目的のように実際のネットワーク攻撃シミュレーションを目的としている．上位階層の構造とプロトコルの現実的な実装を支援しているが，ユーザの開発を必要としている．さらに，OMNET++でしか使用できない限界がある．

3. システム構成

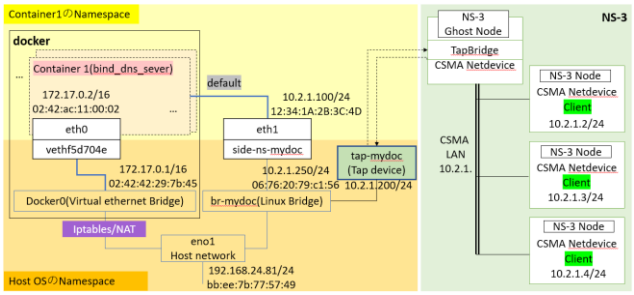

図 2　ns-3 と Docker の通信仕組み

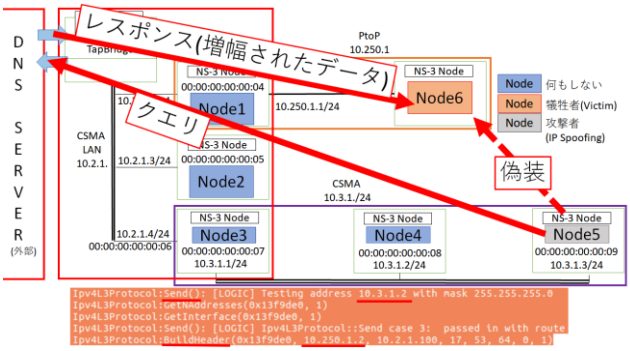

図 3　IP Spoofing 実験環境・ヘッダ Build 結果

図 1 は ns-3 と Docker を用いて構成した DNS リフレクタ攻撃シミュレーション環境である．DNS サーバとしてシミュレータ外の Docker 上で BIND サーバを構築した．Docker を用いたのは，シミュレーション以外の資源消費を最小限にするためである．

ns-3 と Docker 間の通信環境は ns-3 の Tap Bridge と呼ばれる機能を用いて構築する．ns-3 と Docker のネットワーク構成を図 2 に示す．ns-3 は実際のインターネットホストとシミュレータ内部のノードが通信するための Tap Bridge Model を提供する．今回は Tap Bridge Model の UseBridge Mode を通信に利用する．

Linux の Tap Device は Tap Bridge を通過した ns-3 のパケットをイーサネットフレームのまま Linux 仮想 Bridge に送信する．Linux 仮想 Bridge がパケットをペアで構成されている仮想インターフェースに送信すると，namespace で分離されている BIND サーバとの通信ができる．Docker には docker0 という Docker 上のコンテナを接続する仮想 Bridge が存在するが，ns-3 への返信パケットが ns-3 に受信できるようにするため，BIND サーバのデフォルトゲートウェイを新しく作成した仮想インターフェースに設定する．

4. 実装

4.1 ns-3 内の IP Spoofing

DNS リフレクタ攻撃をシミュレーションするためには ns-3 側から送信するクエリパケットに IP Spoofing の実装が必要である．

IP Spoofing は IP の脆弱性を利用した攻撃手法の一つである．送信元 IP アドレスを攻撃対象となる犠牲者ノードの IP アドレスに偽装することで，DNS サーバからの応答を犠牲者ノードに送り付けることができる．

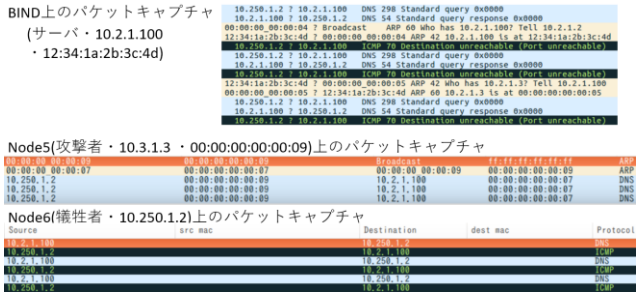

図 4　IP Spoofing 結果

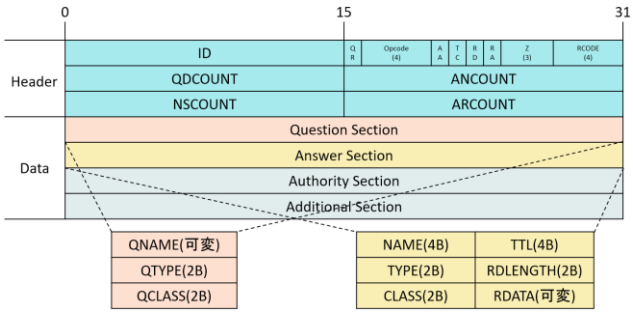

図 5　DNS メッセージフォーマット

IP ヘッダを作る ns-3 のファイルである Ipv4-l3-protocol のコードを変更し、IP Spoofing を可能にした．IP Spoofing の流れは以下の通りである．

- Ipv4-l3-protocol に Attribute として犠牲者コードの IP を表す‘VictimIpv4’アドレスを追加
- シナリオファイルでユーザが特定ノードを犠牲者ノードに設定した場合：設定されたノードの IP アドレスを VictimIpv4 の値に設定
- シナリオファイルで犠牲者ノードの IP アドレスに設定されてない場合：ns-3 では Attribute で ipv4 アドレスを設定すると，IP アドレスの初期設定値が 102.102.102.102 であるため VictimIpv4 は 102.102.102.102 で設定されている
- IsEqual 関数を用いて VictimIpv4 と初期 IP アドレス(102.102.102.102)を比較
  - ➢ VictimIpv4 の値が初期値である場合：IP ヘッダの SourceIP の値を変更しない
  - ➢ VictimIpv4 の値が初期値でない場合：IP ヘッダの SourceIP の値に VictimIpv4 の値を挿入

図 3 を見るとヘッダが Build される時，犠牲者ノードの IP に Spoofing されることが分かる．このように Spoofing された犠牲者ノードの IP が BIND サーバと同じネットワークに属するときと，異なるネットワークに属するときとそれぞれについて動作確認を行い，DNS サーバの応答が犠牲者ノードに送信されることを確認した．その結果が図 4 になる．BIND サーバのパケットキャプチャを見るとちゃんと 10.250.1.2 からクエリを受信し，10.250.1.2 にレスポンスを送信するが，攻撃者である Node5 のパケットキャプチャを見ると実際の IP アドレスが 10.3.1.3 にもかかわらず 10.250.1.2 の IP が表示されることで IP Spoofing しているのが確認できる．犠牲者ノードである Node6 のパケットキャプチャを確認すると送信するパケットが存在せず，BIND サーバである 10.2.1.100 からレスポンスを貰い，それに対して ICMP パケットを返す．

4.2 DNS Amplification の実装

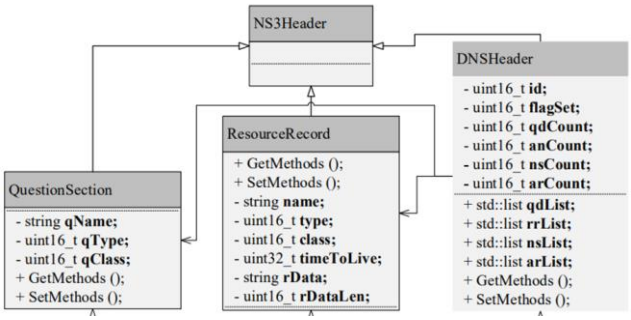

図 6　DNS ヘッダ構造体[文献 7 から引用]

表 1　DNS パケットデータ

	メンバ	設定値
DNS ヘッダ	id	Random
	qr	0
	opcode	0
	aa	0
	tc	0
	rd	1
	z	0
	q_count	1
	ans_count	0
	auth_count	0
	add_count	1
Question Section	Name	ns.hshin.com
	Type	16
	Class	1
Additional Section	Name	NULL
	Type	41
	Class	4096
	TimeToLive	0
	RData	NULL
	RDLength	0

ns-3 のサンプルコードである main-packet-header.cc と文献[7]を参考に DNS ヘッダ実装を進めた．main-packet-header.cc は ns-3 で提供するヘッダクラスを用いて新しいパケットのヘッダを実装したコードであり，文献[7]は DNS パケットを BIND サーバを ns-3 内で実装した研究である．図 5 の DNS メッセージフォーマットを構造体として表現したものが図 6 である．実装した DNS ヘッダを /src/internet/model に含め，アプリケーションレベルでデータを挿入できるようにした．DNS ヘッダには DNS 構造体が定義されていて，入力されたデータを DNS データフォーマットに変換し，Buffer にデータを書き入れたり(Serialization)，Buffer からデータを読み出す(Deserialization)処理をする．

EDNS(0)　RFC6891[8]に定義されている DNS の拡張メカニズムである．トランスポート層プロトコルとして UDP を使用する DNS プロトコルはメッセージ長が 512 バイトに制限されている．そのため 512 バイトオーバのパケットは TCP で伝送するようになるが、そうしたらオーバヘッドが増加してしまう．この問題は DNS に機能を追加する

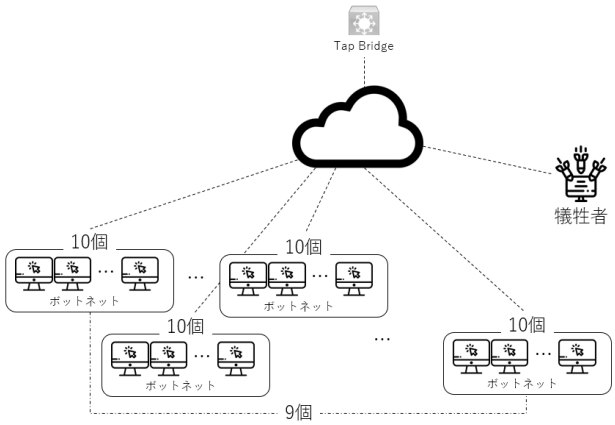

図 7　ns-3 のシナリオ環境

表 2　シミュレーションの Attribute 値

	Attribute	設定値
Channel	p2p Bandwidth(Gbps)	1
	p2p Delay(ms)	0
	csma Bandwidth(Gbps)	1
	csma Delay(ms)	0
Tap Bridge	Mode	UseBridge
	Name	tap-mydoc
Packet	port number	62
	server ip address	10.2.1.100(BIND)
	victim ip address	10.250.1.2
	maxPacketCount	100,000
	Interval(sec)	0.01 と 0.1

(DNSSEC など)のに大きな問題になるため DNS の拡張が提案された．メカニズムとしては DNS ヘッダにフラグの追加ができないため、DNS メッセージの Additional Section に擬似リソースレコード(RR)の形で情報を追加する．EDNS(0)は擬似 RR Type として OPT(41)だけを導入する．EDNS(0)の問題点は大きく 2 つが挙げられる．一つ目は，一部のファイアウォールでは最大 DNS メッセージ長が 512 バイトと想定されているため、ブロックされる可能性があること．二つ目はクエリパケットと比べ非常に大きな応答パケットを簡単に作成できるため DNS リフレクタ攻撃に悪用されることが可能であること．本稿では，二つ目の応答パケットのデータ増幅問題を利用し，効果的な DNS リフレクタ攻撃を実現する．EDNS(0)のデータは文献[8]を参考に Additional Section 擬似リソースレコードとして挿入した．挿入した DNS ヘッダ，クエリ，EDNS(0)のデータは表 1 に示す．

4.3 DNS サーバからのデータ増幅レスポンス用意

BIND サーバの Named.conf，正引きゾーンファイルと逆引きゾーンファイルの設定を行った．Named.conf にはループバックと ns-3 と繋がっているアドレス範囲(10.2.1.0/24)を設定し，正引きゾーンファイルと逆引きゾーンファイルのパスを指定した．正引きゾーンファイルには自身をネームサーバとして指定し，A レコードを ns-3 内の同じネットワークノード 3 個を指定した．そして，増幅攻撃用 TXT レコードに UDP Payload Size 最大サイズ 4096 バイトを満たす

ように合計 3819 文字を記入した．逆引きファイルには正引きゾーンファイルを同じくネームサーバのドメインと PTR レコードを ns-3 内の同じネットワークノード 3 個分指定した．

4.4 ns-3 のシミュレーションシナリオ

ns-3 には DDoS 攻撃の動作をするシナリオファイルを作成した．その環境は図 7 で，シミュレーション Attribute は表 2 の通りである．まず，シミュレーション環境としては外と通信するための Tap Bridge と異なるネットワークに犠牲者ノード 1 個が存在し，攻撃するための異なる 9 個のネットワークで各 10 個のボットが合計 90 個存在する．予想されるシミュレーション内容は以下の通りである．

- ボットは 3 秒ごとに 1 個ずつ増えながら犠牲者 IP に偽装して DNS サーバにパケットを送信
- クエリを受信した DNS サーバは増幅されている TXT レコードレスポンスを犠牲者 IP に返信
- トラフィックの限界時点になるとスループットが増えず，一定値を維持

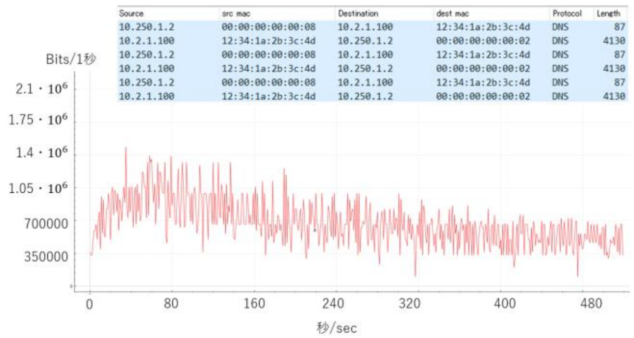

図 8　DNS レスポンスパケットのスループット（100ms）

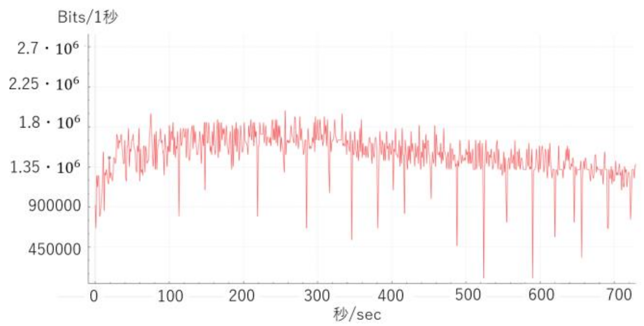

図 9　DNS レスポンスパケットのスループット（10ms）

5. 結果

ns-3 上で設定できる csma チャンネルの bandwidth は 1Gbps に設定し，パケット送信インターバルは 0.1 秒(100ms)と 0.01 秒(10ms)の二つの場合を想定しシミュレーションを行った．図 2 の構成の中，仮想 Bridge である br-mydoc で Wireshark を使用したスループットの観測結果を図 8，図 9 に表す．

パケットキャプチャを見ると 1 つの送信パケットに対して約 4kByte のレスポンスが返ってくるのが分かる．それに比例し，二つのグラフとも 0 秒からスループットもだんだん上がるのが確認できる．しかし，100ms の場合 960Kbps, 10ms の場は 1.6Mbps に到達するとそれ以上上がらなく逆に落ちるという結果が見れた．落ちている中でも 1280Kbps, 960Kbps，640Kbps，320Kbps などの値でグラフが維持されることも分かる．そこで，予想したスループットの結果には満たされないことを確認した．

6. 考察

このような結果で考えられることとしては，現在 ns-3 のシミュレーションで使用している csma チャンネルの衝突のよる性能低下が挙げられる．csma はバス型ネットワークトポロジーであるためノード数が多いと衝突が起きやすく，すべてのノードに影響を与える．そのためパケットを送信するノードが一定値以上になると衝突が起き，DNS サーバに至るノード数自体が減ると考えられる．しかし，現在 ns-3 には NetDevice モジュールとして p2p，csma，wifi を提供している．この三つ以外の NetDevice を使用するとしたら新しく NetDevice モジュールを作成する必要がある．

7. おわりに

今回は ns-3 と BIND サーバを用いて実際の DNS リフレクタ攻撃を再現しエミュレーションを行った．実際の DNS リフレクタ攻撃のように動作し，最初は想定した通りノード数が増えるほどスループットが増加するが，一定スループットの値まで上がると逆に落ちることが観測された．今後は実験結果として想定した結果が導出されなかったためそれに対する原因検討と解決，そして実際の DNS リフレクタ攻撃と比較し，シミュレーションに影響を与える要素を明らかにする．

謝辞

本研究は JSPS 科研費 JP18K18045 の助成を受けたものです．

参考文献

[1] 国立研究開発法人 情報通信研究機構,サイバーセキュリティ研究所 サイバーセキュリティ研究室, “NICTER 観測レポート 2018” (2019)
[2] 瀧本栄二, “ネットワークセキュリティシミュレータに関する検討”, 信学技報, MoNA2018-14, Vol. 118, No. 170, pp. 47-51 (2018).
[3] ns-3,6 MAY 2019, https://www.nsnam.org/ (参照 16 JUNE 2019)
[4] GNS3,16 JUNE 2019, https://www.gns3.com/ (参照 16 JUNE 2019)
[5] Stephan Schmidt, Rainer Bye, Joël Chinnow, Karsten Bsufka, Ahmet Camtepe, Sahin Albayrak, “Application-level Simulation for Network Security”, Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops}, Simutools '08, pp. 33:1--33:10 (2008).
[6] Leovigildo Sánchez Casado; Rafael A. Rodríguez-Gómez; Roberto Magán-Carrión; Gabriel Maciá-Fernández, “NETA: Evaluating the effects of NETwork Attacks. MANETs as a case study”, Advances in Security of Information and Communication Networks, (SecNet 2013), Communications in Computer and Information Science, Vol. 381, pp. 1-10 (2013).
[7] Janaka Wijekoon, Hiroaki Nishi, “Implement Domain Name System (DNS) on network simulator-3”, SIMUTOOLS'16, pp. 56-65 (2016).
[8] João Damas, Michael Graff, Paul A. Vixie, “Extension Mechanisms for DNS (EDNS(0))”, RFC 6891 (2013).