ARP Cache Poisoning 攻撃に対する即時防壁

07H068 寺岡良真

目次

概要

情報ネットワーク社会において、もはや情報セキュリティが重要であることは衆知の事実である。攻撃の技術も日々進歩し、それによって知的財産や金銭的な問題も多発している。本研究では、LAN 内においてクリティカルな攻撃である ARP Cache Poisoning と呼ばれる手法についての検証、及び即時的な対策を講じた。ARP Cache Poisoning は MITM (中間者攻撃) に属する攻撃手法である。これは ARP(Address Resolution Protocol) の仕様を突く攻撃であるので、現状で 多くの OS では攻撃を防ぐことができない。また、ネットワークの下位層で起こるため、攻撃自体が非常に気づきにくいことも挙げられる。代替え手段として事前に cache を静的に設定する対策が考えられるが、ネットワークの規模や OS の再起動後毎に設定する必要があるため実用的でない。この攻撃自体は他の攻撃への踏み台要素であることが多いので、次のさらなる攻撃による情報の漏洩や改竄なども懸念される。

そこで攻撃を回避できないならば、まず ARP Cache Poisoning 発生自体の早急な検出が必要であると考えた。ARP Cache Poisoning は攻撃発動時にネットワーク上に大量の ARP パケットが発生し、連続して同じ情報を持つ ARP パケットをコンピュータが受信する特徴を確認した。これにより、大量の同じ ARP パケットを取得した際に攻撃が発生していると検出できる。しかし、これでは攻撃の検出だけであって攻撃自体を止めることはできない。次の二次的な攻撃を阻止する、または遅らせる必要がある。ARP Cache Poisoning の攻撃を有効にさせないためには、正しい ARP cache を OS に静的に設定することが挙げられる。攻撃の被害に合うと即座に ARP cache が書き換えられるため、先に正しい ARP cache のバックアップを定期的にとっておくことで対処した。そして攻撃発生を検出したと同時に、バックアップから静的に OS に設定するよう防壁を提案した。即時的な対策がとれることから、ARP Cache Poisoning を受けても被害を拡大することを阻止できる。また、警告を発するので、管理者が攻撃者のピックアップや、ログをとるための時間稼ぎとなる利点になると考えた。

1 序論

現在、ネットワーク上に存在するコンピュータは危険にさらされている。その中でも、ARP Cache Poisoning と呼ばれる攻撃が確認されている。ARP Cache Poisoning とは UNIX 系、MS Windows、Mac OS、問わず多くの OS において不可避な攻撃である。これはアプリケーションのバグなどの脆弱性をついているわけではなく、Ethernet [*1] の仕様に基づく攻撃であるからである。そのため OS 自体に対策を施すためのパッチやアップデートは期待できない。この攻撃はコンピュータ間の通信に攻撃者が割り込み、通信相手のコンピュータを偽って正しい通信として認識させる。攻撃者が対象間に割り込んでいるため、コンピュータ間の通信すべてに関与することができる。これにより情報の漏洩や改竄、また次なる攻撃への踏み台にされてしまう危険が懸念される。LAN 内に攻撃者を侵入させない物理的な措置も考えられるが、物理的な制約やソーシャルエンジニアリングを回避することは非常にコストがかかってしまう。他の攻撃と併用すればインターネット越しに攻撃を成功できることも確認されているので、コストに見合った対策とは言い難い。

そこで本研究では物理的な措置ではなく、非常に小さなスクリプトを使った ARP Cache Poisoning の対策を提案し、テスト環境においてその有効性を検証した。

[*1] コンピュータネットワーク規格で、Ethernet は IEEE 802.3 委員会によって標準化された。

2 現状での脅威

2.1 ARP について

Address Resolution Protocol (アドレス解決プロトコル)。既存の Ethernet 環境においてネットワーク層の IP アドレスだけでは通信を行うことが出来ない。これは IP アドレスがあくまで論理的なアドレスであり、物理的なアドレスではないためである。そこで通信を可能にするために、さらに下位のデータリンク層の物理的なハードウェアアドレスである MAC アドレスを取得する必要がある。そこで ARP は前述の IP アドレスと MAC アドレスの対応付けを行う。対応付けされた各ホストの ARP テーブルはキャッシュとして OS が保持している。キャッシュは動的なものと静的なものが混在し、OS のアーキテクチャに基づいて格納される。

2.2 MITM について

中間者攻撃 (Man-In-The-Middle attack)、またはバケツリレー攻撃 (bucket-brigade attack)。コンピュータ間に攻撃者が割り込むことで、両者が行っている通信に介入する攻撃手法。攻撃者は通信を行うコンピュータに、自分が正しい相手であるかのように振る舞うことで攻撃が成立する。攻撃者が正しい通信相手を偽っているため、被害を受けているコンピュータからは攻撃者の存在を感知し難い。またコンピュータ間に割り込むため、通信内容の盗聴や情報の改竄、奪取が可能となる。

2.2.1 ARP cache pisoning とは

前述の ARP にはセキュリティ上の致命的な問題が 2 つ存在する。1 つは受信した ARP パケットが本当に正しい送信者から送信されたパケットか判断することができない。つまり、ARP は全ての ARP パケットは必ず正しい送信者が送信したとするので、攻撃者が正しい送信者と偽装していた場合も見破ることができない。そして 2 つ目が、受け取った ARP パケットの内容を検閲せずに OS がキャッシュとして保持するので、不正な ARP パケットもそのまま受理してしまう可能性がある。これはプロトコルの仕様の段階で ISMS [*2]でいわれる C.I.A [*3]が失われている状態といえる。ARP Cache Poisoning は以上の ARP の仕様を的確に悪用した攻撃手法である。

- 割り込みたい通信をしているホストに毒 ARP パケットを投げて汚染させる。
- 通信間から攻撃者の望む攻撃ツールを待機させておく。
- 動的キャッシュなので定期的に ARP 毒で汚染させておく。
- 他のアタック手法と併用して様々な攻撃を行う。

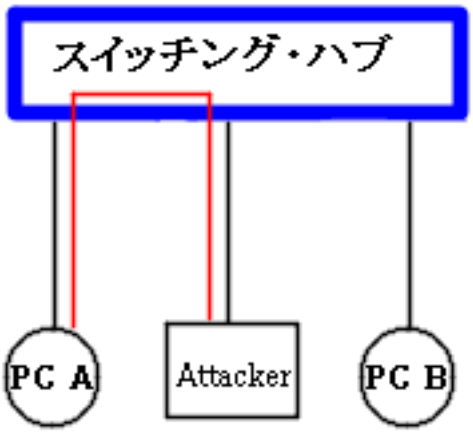

図 1　ARP Cahce Poisoning 手順 1 Attacker は PC A、PC B が利用しているスイッチングハブに接続し、PC A に対して ARP 毒をユニキャストする。ユニキャストを赤線で示す。

ARP cache pisoning の利点は基本的に低いレイヤーでの攻撃なので、通常の検知方法ではほとんど対象に悟られにくい攻撃である。また、被害者間の通信に割り込んでいるので情報の改竄や奪取が容易である。これにより、場合によっては SSH や SSL も意味をなさない。また、他の攻撃技術との親和性が高く、様々な攻撃と組み合わせることが出来る。IDS/IPS からの検知にも耐性が大きいこともいえる。逆に欠点としては、あくまで ARP の届く範囲での攻撃なので技術的な面ではそれ以上には被害の範囲は広がらない (もちろん他の攻撃手法を駆使することで可能になる)。定期的に対象に毒を汚染しなくてはいけないことである。

[*2] Information Security Management System
[*3] 機密性 (Confidential)、完全性 (Integrity)、可用性 (Availability)

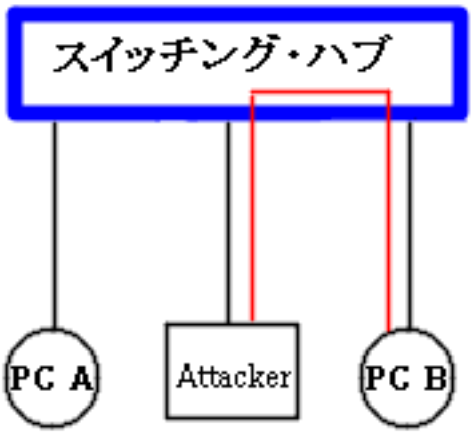

図 2　ARP Cache Poisoning 手順 2 Attacker は PC B に対して ARP 毒をユニキャストする。ユニキャストを赤線で示す。

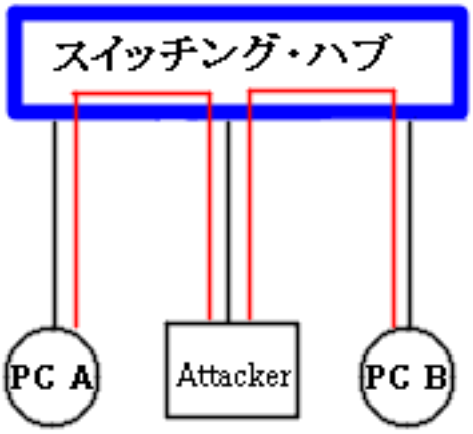

図 3　ARP Cache Poisoning 成功 PC A を PC B の通信は相手に気づかれることなく Attacker を経由することになる。PC A と PC B の通信経路を赤線で示す。

2.2.2 あまり有効でない ARP Cache Poisoning 対策

基本的に有効な対策をとることができないとされている。代替え手段として、LAN 内のすべてのコンピュータに静的にキャッシュを設定する方法がある。しかし、コンピュータの数の問題や環境変化の対応ごとに設定し直すため実用的ではないとされる。

3 本研究で提案する ARP Cache Poisoning 対策

第 2.2.1 節で述べたように、この攻撃自体はほぼ不可避である。しかし、確認した攻撃ツールはいずれも短時間に連続して同じ ARP 毒を投げ続ける動作が伺える。保安上の理由により詳細なツール名は記述しないが、このパターンを解析することで攻撃の検出は可能であると考えた。いずれにせよ攻撃自体は成功してしまうので、正常な通信の継続と攻撃者特定などの時間稼ぎが必要な点を含め、管理者への早期連絡を可能にするものとして対策を行うことにした。

問題としては、上位 Cracker などは非常にスマートに ARP 毒を汚染させてくるので、この対策では攻撃の誤検出が発生する可能性が存在する。よって、対象となる仮想的な攻撃者は Script Kiddy などの低級アタッカーとみなす。

3.1 攻撃の検出

ARP Cache Poisoning の特徴として、攻撃時に普通では考えられない量の ARP が発生することが挙げられる。本実験の検証の際、実際に攻撃に使われるツールを用いたところ、同じ情報を持った ARP パケットの発生を確認した。そこで本研究で開発したスクリプトには、同一の情報を持った ARP を連続して 3 つ以上受け取ると攻撃が発生したとみなした。この設定した値は変更可能である。なお、攻撃が発生した場合はコンソール画面に警告を発する。

図 4　攻撃の検出 Warning の ARP Cache Poisoning に対し、提案システムの Warning を出す。

3.2 一時防壁

防壁と記述しているが Fire wall のようなものではない。ARP 毒はパケットを受けとるとすぐさま cache に反映してしまうので、定期的に ARP cahce のバックアップを作成を行う。なお ARP の動的な cache を保持する時間は OS により異なり、MS Windows の場合は 2 分、Linux では 15 分、ルータは 60 分となっている。本研究では Linux を採用しているのでバックアップの時間はおよそ 25 分間隔としているが、もちろん OS ごとに設定を変更することが可能である。攻撃の発生を検出した際に即座にバックアップの ARP cache を OS に静的に設定する。Red hat 系 Linux は静的に設定しても動的 cache に汚染されてしまう仕様であるため、攻撃が続く限り何度も静的キャッシュで上書きすることで対策とした。この場合ネットワークトラブルが発生し通信不能になってしまう恐れがあるが、攻撃も失敗に終わるので情報改竄などの二次災害の回避は可能とみなした。また、攻撃の検出は成功しているので管理者が早期に攻撃を関知できる。

4 考察

本研究で提案する対策によって、LAN 内の ARP cahe poisoning の有効性は格段に低下した。依然、経験豊富な Cracker によるスマート攻撃は可能であるが、攻撃の検出を取りこぼすような false negative は発生しなかった。

本研究の利点としては、ARP Cache Poisoning という検出が非常に困難な攻撃から、発生直後に即時対応できることである。また、ruby スクリプトであることから、他の OS への導入に負担が少ないことが挙げられる。

本研究の欠点としては、あくまで一時的な時間稼ぎにすぎない点である。LAN 内の防御に必要なコンピュータに搭載する必要がある。上位 cracker によるスマートな攻撃などと、正当な　ARP パケットとの誤検知の可能性も示唆される。これに対する対策はいまのところ非常に難しい。

5 まとめ

本研究では Script Kiddie レベルの攻撃に一時的に防御策を講じることを目的としてシステムの開発を行った。LAN 内で検出の難しい ARP Cache Poisoning の被害を受けても、いち早く関知者は発見することができる。また、一時的な防壁を展開することによって、コンピュータ通信の情報の漏洩や改竄の即時的な防御が可能である。以上の点から ARP Cache Poisoning に対する一時的な防壁という目標は達成できた。

経験の豊富な Cracker はよりスマートな ARP Cache Poisoning を行い、正当なパケットと判別し難い攻撃を行ってくる。今後は、これに対する誤検知や攻撃検出精度の向上を目指すアプローチが必要である。

謝辞

本研究を進めていく上で、大垣 斉講師から御指導及び御協力を戴きました。合同会社セキュリティ・プロフェッショナルズ・ネットワーク代表社員 Eiji James Yoshida 氏は本研究を行うきっかけを戴きました。また、研究室のメイルリストのメンバの方々には御助言を賜りました。ここに深く感謝の意を表します。

参考文献

[1] RFC826, An Ethernet Address Resolution Protocol

[2] RFC903, A Reverse Address Resolution Protocol

[3] ARP Cache Poisoning mecahnism and measures

[4] Ruby/Pcap extension library

[5] W. リチャード スティーヴンス (著), W.Richard Stevens (原著), 橘 康雄 (翻訳), 井上 尚司 (翻訳). 詳解 TCP/IP〈Vol.1〉プロトコル

付録 A　Source code

A.1　攻撃に対する即時的な防壁スクリプト

ARP-TYPE は Script kiddie による ARP Cache Poisoning に対する即時的な防壁であるとともに、攻撃の存在を管理者へ通告するプログラムである。

— ARP-TYPE.rb begin—

```
#! /usr/bin/ruby

# Ruby on R0zette :)

require 'pcap'

def usage
    print <<-EOM
====================
     arp-type
====================
Usage   : arp-type.rb -i [nic]
Version : 0.2.6
Auther  : R0zette (r0zette.k@gmail.com)
    EOM
end

def warning(en)
    print <<-EOM

         WARNING
============================
    [On Marishiei Sowaka]
The big enemy is approaching
    at full throttle.
According to the data. It is
 identified as "#{en}"
============================
        NO REFUGE
    EOM
end

def dump_pkt(nic)
    snif = Pcap::Capture.open_live(nic)
    filt    = Pcap::Filter.new('arp', snif)
```

```
    snif.setfilter(filt)

    snif.each_packet{ |pkt|
        data = pkt.raw_data.unpack("H*")
        pkt_analyz(data)
    }
    snif.close
    exit
end


def pkt_analyz(d)
     $fst = d[0]
     if $scd = $fst
         $count += 1
     else
         $count = 0
     end

     if $count == 3
         smac = $fst[45..55]
         warning(smac)
         fixed_cache
         $count = 0
     end
     $scd = $fst
end

def make_ethers
    refile = open("/proc/net/arp")
        data = refile.read
        ip   = data.scan(/\d+\..*\.\d+/s)
        arp  = data.scan(/\w+:.*:\w+/s)
    refile.close

    # Write "ethers" temporary file.

    wrfile = File.open("ethers", "w")
        arp.each_with_index{ |x,i|
            wrfile.print ip[i]," ",arp[i],"\n"
        }
    wrfile.close
end
```

```
def fixed_cache
    system("arp -f ethers")
    p "Set to static ARP cache!!"
end


# --- Main loochin ---

$fst = ""
$scd = ""
$count = 0

if ARGV[0] != '-i'
        usage
        exit
else
    if ARGV[1].nil?
        usage
        exit
    end

    cache_t = Thread.new do
        loop do
            make_ethers
            sleep 1500
        end
    end
    dump_pkt(ARGV[1])
    exit
end
```

— ARP-TYPE.rb end —