=== ch10.docx

[chap]10章　セキュリティのためのパケット解析

本書の大半はパケット解析を利用したネットワークのトラブルシューティングに割かれていますが、現実にはかなりのパケット解析がセキュリティ目的で行われています。これは不正侵入者を防ぐためにネットワークトラフィックを見直すアナリスト、あるいは侵入されたホストでのマルウェア感染の範囲を突き止めるフォレンジック調査官の仕事だったりします。セキュリティのためのパケット解析は大きなテーマであり、本が１冊書けるほどです。ここではセキュリティに的を絞ったパケット解析のさわりだけを紹介します。

この章ではセキュリティ関係者の立場で、ネットワークレベルでのシステム侵害をさまざまな角度から見ていきます。まずネットワーク偵察、悪意あるトラフィックのリダイレクト、システムエクスプロイテーションなどについて説明します。次に侵入アナリストとなって、侵入検知システム（IDS）からの警告をもとに、トラフィックを細かく分析します。本章を読めば、セキュリティ関連を担当していなくても、ネットワークセキュリティの本質が理解できるでしょう。

偵察

攻撃者が最初に行うのは、標的とするシステムの徹底的なリサーチです。この段階を一般に「フットプリンティング」といい、標的とする企業のWebサイトやGoogleなど、公開されているさまざまなリソースを利用して行われます。リサーチが完了すると、攻撃者は開いたポートや実行されているサービスを見つけるため、標的のIPアドレス（またはドメイン名）のスキャンを開始します。

攻撃者はスキャンによって、標的が稼働していて、アクセスできるかどうかを判断します。たとえば銀行強盗が、メインストリート123番にある大手銀行での強盗を計画していたとしましょう。強盗は念入りな強盗計画を立てたものの、現地へ行ってみると、銀行はバインストリート555番へ引越していました。もっと悪いケースだと、昼間の営業時間中に徒歩で銀行に押し入って、金庫から盗むつもりでいたのに、銀行が休業日だったとしたらどうでしょう。標的が稼働していて、アクセスできるかどうかを確認するのが、最初に越えなければならないハードルなのです。

スキャンによるもうひとつの重要な成果は、標的のどのポートが待ち受けしているかが分かることです。銀行強盗のたとえに戻って、強盗が銀行施設の見取り図の知識がまったくないままに、銀行へ押し入ろうとしたらどうでしょう。セキュリティの弱点を知らないわけですから、侵入することはできません。

この項では、ホスト、その開いたポート、またネットワークの脆弱性を識別するのに使われる、一般的なスキャン手法についていくつか説明します。

注　これまででは、送信者と受信者、あるいはクライアントとサーバの接続についてのみ触れてきました。本章では攻撃者または被害者の側の通信について説明していきます。

SYNスキャン

システムに対して最初に行われるのがTCP SYNスキャンで、これはステルススキャンまたはハーフオープンスキャンとも呼ばれます。SYNスキャンがもっともよく使われるのには理由があります。

・非常に早く信頼性が高い。

・TCPスタックの実装に関わらず、すべてのプラットフォームで正確に機能する。

・他のスキャン手法よりもノイズが少ない。

TCP SYNスキャンは、標的のどのポートが開いているかを判断するのに、3ウェイハンドシェイクプロセスを使います。攻撃者はポートと正常な通信を確立するふりをして、被害者のポートにTCP SYNパケットを送ります。被害者がこのパケットを受け取ると、図10-1のようなことが起こります。

図10-1　TCP SYNスキャンによって起こる可能性のあること

攻撃者　被害者　80番ポートが開いている

攻撃者　被害者　80番ポートが閉じている

攻撃者　被害者　80番ポートにフィルタがかかっている

被害者のマシンのサービスがポートで待ち受けをしていてSYNパケットを受けとると、攻撃者にTCP SYN/ACKパケットを応答として送ります。すると攻撃者はそのポートが開いていて、サービスが待ち受けしていることを知ります。通常であれば、ハンドシェイクを完了させるために最終のTCP ACKパケットが送られますが、この場合攻撃者はこれ以上ホストと通信する気がないため、TCPハンドシェイクを完了させません。

スキャンしたポートでサービスが待ち受けしていない場合、攻撃者はSYN/ACKパケットを受け取りません。被害者のOSの設定によっては、攻撃者はポートが閉じていることを意味するRSTパケットを応答として受け取ります。あるいは何の応答もない場合もあります。これはファイアウォールやホスト自身によって、ポートにフィルタがかかっているという意味です。あるいは送信の途中で、応答が消失しただけかも知れません。一般にはポートが閉じているということですが、決定的ではありません。

ファイルsynscan.pcapは、NMAPツールを利用したSYNスキャンのよい例です。NMAPは、Fyodorが開発した、堅牢なネットワークスキャンアプリケーションで、どんなスキャンでも実行することができます。NMAPは<http://www.nmap.com/download.html>から無料で入手可能です。

サンプルのキャプチャには約2000個のパケットが入っていて、このスキャンが適度な規模であることを示しています。スキャンの範囲を確認する最良の方法のひとつが、[Conversations]ウィンドウを見ることです（図10-2）。ここには攻撃者（172.16.0.8）と被害者（63.13.134.52）間のIPv4の対話ひとつしかありません①。またこの2つのホスト間に、TCPの対話が1994あるのに気づくでしょう②。

図10-2　さまざまなTCP通信が行われていることを示す[Conversations]ウィンドウ

スキャンは非常に早く行われるため、それぞれのSYNパケットの応答を見つけるのに、キャプチャファイルをスクロールするのはあまりよい方法とはいえません。パケットへの応答を受け取る前に、さらにパケットが送られているかも知れないからです。幸いなのは、適切なトラフィックを見つける手助けとなるフィルタが作成できることです。

SYNスキャンでのフィルタの使用

フィルタの一例として、被害者の443番ポート（HTTPS）に送られた最初のパケット、SYNパケットを取り上げます。このパケットに対する応答を見るため、443番ポートを出入りするすべてのトラフィックを表示するフィルタを作成します。迅速に行う方法は次の通りです。

1．キャプチャファイルの最初のパケットを選択します。

2．パケット詳細のペインのTCPヘッダを広げます。

3．[Destination Port]フィールドを右クリックし、[Prepare as Filter]を選択、[Selected]をクリックします。

4．これで443番ポートへ向かうすべてのパケットのフィルタが、フィルタダイアログに置かれます。443番ポートからのパケットもフィルタしたいので、フィルタダイアログの一番上をクリックし、dst部分を消去します。

すると攻撃者から被害者へ送られる2個のTCP SYNパケットがフィルタされました（図10-3）。

図10-3　SYNパケットでの2回の接続確立の試み

どちらのパケットにも応答はないので、被害者であるホストかファイアウォールなどによってフィルタされたか、あるいはポートが閉じている可能性があります。443番ポートに対するスキャンだけでは判断できません。

同じ手法をほかのパケットでも試して、違う結果が出るかどうかを見てみましょう。まずフィルタの横の[Clear]ボタンをクリックし、先ほど作成したフィルタを削除します。次にリストから9番目のパケットを選びます。これはポート53番、一般にDNSに関連するポートへ送られたSYNパケットです。先ほどと同様に、このポートに向かうパケットのフィルタを作り、dstを消去して、ポート53番を通過するすべてのTCPに適用させます。このフィルタを適用したら、5つのパケットをチェックしましょう（図10-4）。

図10-4　ポートが開いていることを示している5個のパケット

最初のパケットは、キャプチャの初めに選択したSYNパケットです。2番目は被害者からの応答で、3ウェイハンドシェイクを確立したときに戻ってくるべきTCP SYN/ACKパケットです。通常であれば、次のパケットは最初にSYNを送ったホストからのACKであるはずです。しかしここでは、攻撃者は接続を完了させないので、応答を送りません。その結果、被害者はSYN/ACKを3回ほど再送します。53番ポートでの接続を試みるとSYN/ACK応答が戻ってきたので、このポートでサービスが待ち受けしていると考えていいでしょう。

このプロセスを13番目のパケットでもう一度繰り返してみましょう。これは113番ポート、つまりIRC識別や認証に使われるIdentプロトコルと関連するポートへ送られた、SYNパケットです。このポートに同じタイプのフィルタを適用すると、図10-5のように4つのパケットが表示されます。

図10-5　SYNのあとにRSTが送信、ポートが閉じていることを示している

最初のSYNパケットのあと、すぐにRSTが被害者から送信されています。これはこのポートで接続を許可しておらず、サービスが稼働していないことを意味しています。

ポートの開閉を見極める

SYNスキャンに対する応答の違いを理解したら、今度はどのポートが開いていて、どれが閉じているのかを判断する迅速な方法を探します。今度の答えも[Conversations]ウィンドウにあります。このウィンドウでは、パケット番号によってTCP対話をソートし、Packetsカラムを2回クリックして数値が高い順に並べることができます（図10-6）。

図10-6　[Conversations]ウィンドウで開いているポートを探す

3つのポートをスキャンした結果、それぞれの対話に5つのパケットが表示されました①。これら5つのパケットが最初のSYN、それに応答するSYN/ACK、被害者から再送されたSYN/ACKを表しているので、53番、80番、22番のポートが開いていることが分かります。

その他の5つのポートでは、2つのパケットしか表示されていません②。１つめは最初のSYN、そして2番目はRSTです。つまり113番、25番、31337番、113番、70番は閉じています。

残りのエントリには1つしかパケットが存在しないので、被害者のホストが最初のSYNに応答しなかったことになります。したがって残りのポートは閉じていると考えられますが、確かではありません。

OSフィンガープリント

攻撃者は標的のOSを知るために相当な努力をします。OSが分かれば、適切な攻撃が仕掛けられるからです。またOSを特定することで、重要なファイルやディレクトリがどこにあるかが分かるので、システムにアクセスしやすくなります。

OSフィンガープリントは、実際にそのシステムにアクセスすることなく、システム上で実行されているOSを特定するのに用いられるテクニックの総称です。OSフィンガープリントには、パッシブとアクティブの2種類があります。

パッシブフィンガープリント

パッシブフィンガープリントでは、標的から送られたパケット内の特定のフィールドを調査することによって、OSを判断します。この方法がパッシブ（受動的）とされるのは、標的が送信するパケットをチェックするだけで、自らはパケットを送信しないためです。ステルスに実行できるため、攻撃者にとっては理想的な方法です。

そうはいっても、送られてきたパケットを調べるだけで、どうやって標的のOSを特定できるのでしょうか。実はこれは非常に簡単で、RFCで定義されたプロトコルの仕様のおかげで可能になっているのです。TCP、UDP、IPヘッダにはいくつもフィールドがありますが、デフォルト値が設定されていません。つまりTCP/IPスタックを実装する場合、各OSはこれらのフィールドのデフォルト値を定義する必要があります。表10-1は一般的なフィールドと、さまざまなOSでのデフォルト値を示しています。

表10-1　一般的なパッシブフィンガープリントの値

プロトコルヘッダ　IP　IP　TCP　TCP　TCP

フィールド　最初のTTL　フラグメント禁止フラグ　最大セグメントサイズ　ウィンドウサイズ　SackOK

デフォルト値

OS

ファイルpassiveosfingerprinting.pcapに含まれている2つのパケットがよい例です。どちらも80番ポートに送られたTCP SYNパケットですが、違うホストから送信されています。このパケットに含まれる値を表10-1に照合するだけで、それぞれのホストのOSが特定できるはずです。各パケットの詳細は図10-7に示しました。

図10-7　パケットからOSが特定できる

表10-1を参照して、パケット別に関連するフィールドの詳細（表10-2）を作りました。

表10-2　パケット別のOSフィンガープリント

プロトコルヘッダ　フィールド　パケット1の値　パケット2の値

IP　最初のTTL

IP　フラグメント禁止フラグ　セット　セット

TCP　最大セグメントサイズ　1440バイト　1460バイト

TCP　ウィンドウサイズ　64240バイト　2920バイト

TCP　SackOK　セット　セット

これらの値によって、パケット1はWindows搭載機から、パケット2はLinux搭載機から送られたと結論づけることができます。

表10-1の一般的なパッシブフィンガープリントのフィールドのリストは、完全なものではありません。間違いも多いので、値に狂いが生じる場合もあるでしょう。そのため、パッシブOSフィンガープリントから得られた結果に全面的に頼ることはできません。

注　OSフィンガープリント手法に使えるツールのひとつがp0fです。このツールはパケットキャプチャで関連フィールドを解析し、疑わしいOSをアウトプットします。p0fのようなツールを使うと、OSだけでなく、そのバージョンやパッチまでも特定できます。p0fは<http://lcamtuf.coredump.cx/p0f.shtml>からダウンロードできます。

アクティブフィンガープリント

受動的にトラフィックを監視するだけでは結果が得られない場合、より直接的なアプローチが必要になります。これをアクティブフィンガープリントと呼びます。攻撃者はOSを特定する応答を引き出すため、特別に作ったパケットを被害者に送信します。この手法では被害者と直接やり取りを行うのでステルスではありませんが、かなり効果はあります。

ファイルactiveosfingerprinting.pcapには、Nmapスキャンユーティリティを使った、アクティブOSフィンガープリントスキャンの例が含まれています。ファイル内のパケットは、OSを特定する応答を引き出すよう設計されたプローブを、Nmapが送った結果によるものです。Nmapはこれらプローブへの応答を記録してフィンガープリントを作成し、データベースの値と比較してOSを特定します。

注　NmapがアクティブにOSを特定するために用いる手法はかなり複雑です。NmapによるアクティブなOSフィンガープリントがどのように実行されているかをさらに学ぶには、Nmapの作者であるGordon “Fyodor” LyonによるNmapガイド「Nmap Network Scanning」を参照してください。

エクスプロイテーション

攻撃者はみな、エクスプロイテーションを目的としています。リサーチを行い、標的を偵察し、脆弱性を見つけ、標的にアクセスするためのエクスプロイトを準備します。この章の残りの部分では、Microsoftの脆弱性のエクスプロイト、ARPキャッシュポイゾニングによるトラフィックのリダイレクト、データ引き出しを実行するリモートアクセス型トロイの木馬を含む、さまざまなエクスプロイテーション手法を見ていきましょう。

Operation Aurora

2010年1月、Operation Auroraは、当時まだ知られていなかったInternet Explorerの脆弱性を突きました。攻撃者はこの脆弱性を悪用し、Googleやその他企業のマシンを、ルートレベルで遠隔操作したのです。

ユーザーがInternet ExplorerでWebサイトを訪れるだけで、こうした悪意あるコードが実行され、攻撃者は管理者権限を使ってユーザーのマシンにアクセスできるようになります。これには悪意あるサイトへのリンクを含むメールを送信するスピアフィッシングの手口が使われました。スピアフィッシングのメールは信用できる相手から送信されているように見せかけているため、成功率が高いのです。

Auroraは、標的とされたユーザーがスピアフィッシングのメールのリンクをクリックしたときから始まっています。ファイルaurora.pcapにファイルが入っています。

キャプチャは、被害者（192.168.100.206）と攻撃者（192.168.100.202）との3ウェイハンドシェイクで始まっています。最初の接続は80番ポートで行われているので、HTTPトラフィックと考えてよいでしょう。この推測は、4番目のパケットである、/infoのHTTP GETリクエスト①によって確認できます（図10-8）。

図10-8　被害者が/infoのGETリクエストを送信

攻撃者のマシンはGETリクエストの受信を確認し、6番目のパケットで応答コード302（このアドレスは一時的に移動しています）と報告します。このステータスコードは一般に、別のページにリダイレクトする場合に用いられます。ステータスコード302①とともに、[Location]フィールドがロケーション/info?rFfWELUjLJHpPを指定します（図10-9）。

図10-9　このパケットでクライアントのブラウザがリダイレクトされる

応答コードHTTP 302のパケットを受け取ると、クライアントは7番目のパケットで、/info?rFfWELUjLJHpP URLにもうひとつのGETリクエストを送り、8番目のパケットでACKを受け取ります。ACKに続くいくつかのパケットは、攻撃者から被害者へと転送されたデータです。このデータをよく見るため、ストリームのなかの9番目のパケットを右クリックして、[Follow TCP Stream]を選択します。このストリームのアウトプットには、最初のGETリクエスト、302リダイレクト、2番目のGETリクエストが表示されています（図10-10）。

このあとからにわかに状況がおかしくなってきます。攻撃者は非常に奇妙なコンテンツでGETリクエストに応答しているのです。そのコンテンツの最初の部分を図10-11に示しました。

コンテンツは<script>タグに囲まれた、一連のランダムな数字と文字のように見えます①。高水準スクリプト言語を使っていることを示すためにHTML内で使われるのが<script>タグで、このタグで囲まれた部分には通常さまざまなスクリプトステートメントが見られます。しかしこのでたらめな文字と数字は、検出を回避するために暗号化されていることを示しています。ここではこれがエクスプロイトだと分かっているので、テキストのあいまいな部分にはパディングや、脆弱性をエクスプロイトするためのシェルコードが含まれていると考えていいでしょう。

図10-10　クライアントに転送されるデータストリーム

図10-11　<script>タグで囲まれたコンテンツは暗号化されている

攻撃者から送信されたコンテンツの2番目の部分を図10-12に示します。暗号化されたテキストのあとに、ようやく読むことができるテキストを見つけました。プログラミングの知識がなくても、このテキストがいくつかの変数をもとにストリングを変換したもののように見えます。これが</script>タグで閉じられる前のテキストの最後の部分です。

図10-12　サーバから送信されたコンテンツのこの部分には、読めるテキストと疑わしいアイフレームが含まれている

攻撃者がクライアントに送信したデータの最後の部分は2つに分かれています。最初のセクションは<span id="vhQYFCtoDnOzUOuxAflDSzVMIHYhjJojAOCHNZtQdlxSPFUeEthCGdRtiIY">①です。2番目のセクションは<span></span>タグにはさまれている<iframesrc="/infowTVeeGDYJWNfsrdrvXiYApnuPoCMjRrSZuKtbVgwuZCXwxKjtEclbPuJPPctcflhsttMRrSyxl.gif" onload="WisgEgTNEfaONekEqaMyAUALLMYW(event)" />②です。繰り返しますが、このコンテンツは不自然に長い、解読不能なランダムな文字列であり、わざとわかりにくくしている可能性がある点からも、悪意ある活動のしるしと考えられます。

<span>タグに囲まれたコードの部分は「アイフレーム」といい、攻撃者がHTMLページにコンテンツを埋め込むときに使うよくある手法です。<iframe>タグはユーザーが検出できないインラインフレームを作成します。ここでの<iframe>タグは奇妙な名前のGIFファイルを参照しています。被害者のブラウザがこのファイルの参照を閲覧すると、21番目のパケットでGETリクエストを送り①、するとすぐにこのGIFが送られます②（図10-13）。このGIFはおそらく、すでに被害者のマシンにダウンロードされているエクスプロイトコードをトリガするのに利用されると思われます。

図10-13　被害者はアイフレームで指定されたGIFを要求、ダウンロードする

被害者が攻撃者に4321番ポートで接続を開始するとき、25番目のパケットで奇妙なことが起きています。この2番目の通信ストリームをパケット詳細のペインで見てもほとんど情報がないので、再度TCPストリームを見て、やり取りされているデータをよく調べてみましょう。図10-14は[Follow TCP Stream]ウィンドウのアウトプットです。

図10-14　攻撃者がコマンドシェルでやり取りしている

このWindowsコマンドシェルが見えたら、即刻警戒が必要です①。このシェルは被害者からサーバへ送られたもので、攻撃者のエクスプロイトが成功し、ペイロードがドロップされたという意味だからです。エクスプロイトが立ち上がると、クライアントはコマンドシェルを攻撃者に戻します。このキャプチャでは、攻撃者がdirコマンドを入力し②、被害者のマシンのディレクトリを見ていることすら分かります③。

コマンドシェルにアクセスした攻撃者は、被害者のマシンに対し管理者権限が得られるので、好き放題できるようになります。たった1度の、わずか数秒のクリックで、被害者のマシンの管理権限はすべて攻撃者に渡ってしまうのです。

このようなエクスプロイトは、ネットワークIDSに検出されるのを防ぐため、やり取りされる際は分からないように暗号化されています。そのためこのエクスプロイトについての事前の知識やエクスプロイトコードのサンプルがないと、さらなる解析をしない限り、被害者のシステムで何が起きているかを正確に把握するのは困難です。幸いにもこのパケットキャプチャには、悪意あるコードの明らかな兆候、つまり<script>タグで囲まれた難読化されたテキスト、奇妙なアイフレーム、そしてプレインテキストで書かれたコマンドシェルがありました。

Auroraエクスプロイトがどのように攻撃するかをまとめてみましょう。

・被害者は信頼できると見せかけたメールを攻撃者から受け取り、そのなかのリンクをクリックすると、攻撃者の悪意あるサイトへGETリクエストが送信されます。

・攻撃者のWebサーバが被害者に302リダイレクトを送信、被害者のブラウザはリダイレクトされた先のURLに、自動的にGETリクエストを送信します。

・攻撃者のWebサーバは、エクスプロイトと、悪意あるGIFイメージへのリンクを含んだアイフレームを含む奇妙なJavaScriptコードの入ったWebページを、クライアントに表示します。

・被害者は悪意あるイメージにGETリクエストを送信し、これをサーバからダウンロードします。

・先に送信されたJavaScriptコードが悪意あるGIFによって解読され、被害者のマシン上でコードが実行、Internet Explorerの脆弱性がエクスプロイトされます。

・脆弱性がエクスプロイトされると、コードに隠されたペイロードが実行され、4321番ポートで被害者と攻撃者間での新たなセッションが開始されます。

・ペイロードからコマンドシェルが生成、攻撃者へ戻され、攻撃者がコマンドシェルを操れるようになります。

防ぐ側の視点からいうと、このキャプチャファイルからIDSのシグネチャを作成すれば、今後この攻撃が起こるのを防げるかも知れません。たとえば、<script>タグで囲まれたテキストの最後にあるプレインテキスト部分のように、難読化されていない部分をフィルタすることが可能です。あるいはURLに「info」が含まれるサイトへ、302でリダイレクトされたすべてのHTTPトラフィックについて、シグネチャを記述することも考えられます。こうしたシグネチャを現場で使うにはさらに調整する必要がありますが、悪くないアイディアです。

注　悪意あるトラフィックのサンプルをもとにシグネチャを作成するのは、未知の脅威からネットワークを守ろうとする人にとって重要なステップです。ここで説明したようなキャプチャは、シグネチャを作成するスキルを磨くのに最適です。侵入検知と攻撃シグネチャについてさらに知りたければ、Snortプロジェクト<http://www.snort.org/>を参照してください。

ARPキャッシュポイゾニング

ARPキャッシュポイゾニングについては、2章で説明しました。これはネットワークエンジニアにとって、効果的かつ有益なツールとなります。しかしながら、悪意を持って使用すると、中間者（MITM）攻撃として致命的なものにもなりうるのです。

中間者攻撃では、攻撃者は通信の割り込みまたは変更を行うために、2つのホスト間のトラフィックをリダイレクトします。中間者攻撃には、セッションハイジャック、DNSスプーフィング、SSLハイジャックを含む、さまざまな形式があります。

ARPキャッシュポイゾニングでは、特別に作られたARPパケットが、2つのホストがお互いに通信しあっていると思い込ませます。しかし実際には、中間者としてパケットを転送している第3者と通信しているのです。

ファイルarppoison.pcapには、ARPキャッシュポイゾニングの例が含まれています。ファイルを開くと、まず被害者である172.16.0.107がGoogleを閲覧し、検索しているのが分かります。この検索により、DNSクエリが混じったHTTPトラフィックが生じています。

ARPキャッシュポイゾニングはレイヤ2で使われるテクニックなので、パケット一覧のペインでパケットを調べてみましたが、不正行為を見つけるのは難しそうです。パケット一覧のペインにいくつかカラムを追加して、わかりやすくしてみましょう。

1．[Edit]から[Preferences]を選択します。

2．[Preferences]ウィンドウの左側の[Columns]をクリックします。

3．[Add]をクリックします。

4．Source MACと入力し、[Enter]を押します。

5．[Field type]のドロップダウンリストで、[Hw src addr (resolved)]を選択します。

6．追加したばかりのエントリをクリックし、[Source]カラムのすぐあとにくるようドラッグします。

7．[Add]をクリックします。

8．Dest MACと入力し、[Enter]を押します。

9．[Field type]のドロップダウンリストで、[Hw dest addr (resolved)]を選択します。

10．追加したばかりのエントリをクリックし、[Destination]カラムのすぐあとにくるようドラッグします。

11．[OK]をクリックします。

このステップを完了すると、図10-15のような画面になるはずです。これでパケットの送信元と宛先のMACアドレス示す2つのカラムが追加できました。

図10-15　送信元と宛先のMACアドレスを示す新たなカラムが追加されたカラム設定画面

MACアドレスの名前解決がオンになっていれば、通信している通信機器がDellとCiscoのものだと分かるはずです。ここは非常に重要なので覚えておきましょう。なぜかというとキャプチャをスクロールしたとき、54番目のパケットで、Dellのホスト（被害者）と新たに入ってきたHPのホスト（攻撃者）との間に、変わったARPトラフィックが発生しているからです（図10-16）。

図10-16　Dell機器とHP機器間の奇妙なARPトラフィック

先に進む前に、この通信に関わっているエンドポイントを見ておきましょう。表10-3にまとめました。

表10-3　監視しているエンドポイント

役割　メーカー　IPアドレス　MACアドレス

被害者　Dell

ルータ　Cisco

攻撃者　HP　不明

では何がこのトラフィックをおかしくしているのでしょうか。6章のARPについての説明を思い出してほしいのですが、ARPパケットには主に2つのタイプ、つまりリクエストとレスポンスがあります。リクエストパケットは、特定のIPアドレスに結びついたMACアドレスを持つマシンを見つけるため、ネットワーク上のすべてのホストにブロードキャストとして送られます。するとリクエストを送信した通信機器に応答する形で、マシンからパケットが送信されます。こうしたことから考えると、この通信には奇妙な点がいくつかあります。

ひとつには、54番目のパケットは、MACアドレス00:25:b3:bf:91:eeの攻撃者から、MACアドレス00:21:70:c0:56:f0の被害者に直接送信されたARPリクエストです①。この種のリクエストは、ネットワーク上のすべてのホストにブロードキャストされるべきなのに、被害者を直接ターゲットにしています。またこのパケットは攻撃者から送信されたもので、ARPヘッダには攻撃者のMACアドレスが含まれているにも関わらず、IPアドレスはルータのものとなっています。

このパケットのあとには、MACアドレス情報を含んだ被害者からのレスポンスが攻撃者に送られています②。恐ろしいことが起きているのは56番目のパケットです。攻撃者が送ったこのARPレスポンスは、IPアドレス172.16.0.1は、MACアドレス00:25:b3:bf:91:eeだと伝えているのです③。問題なのは、172.16.0.1は00:25:b3:bf:91:eeではなく、00:26:0b:31:07:33だということです。172.16.0.1のルータがパケットキャプチャの最初のほうで被害者とやり取りしていたので、この事実が分かっています。ARPプロトコルは安全ではないため（未承諾のアップデートでもARPテーブルに反映させる）、被害者はルータに行くべきトラフィックを、攻撃者に送ってしまいます。

注　このパケットキャプチャは被害者のマシンからのものなので、全体像は見られません。実際に攻撃を仕掛けるには、攻撃者を被害者だとルータに思わせるため、一連のパケットをルータに送信しなければなりません。しかしこれらのパケットを見るには、ルータ（あるいは攻撃者）からのパケットキャプチャが必要です。

両方をうまくだますと、図10-17のように、被害者とルータ間の通信が攻撃者へと流れます。

図10-17　ARPキャッシュポイゾニング（中間者攻撃）

ARPスプーフィング設定（被害者からの視点）

ルータ（Cisco）

攻撃者（HP）

被害者（Dell）

172.16.0.107が172.16.0.1だと偽る

172.16.0.107は00:21:70:c0:56:f0にある

172.16.0.1は00:25:b3:bf:91:eeにある

ARPスプーフィングの結果（攻撃者がトラフィックに割り込む）

ルータ

攻撃者

被害者

57番目のパケットが、攻撃の成功を確認しています。このパケットを、怪しげなARPトラフィック前に送ったパケット（40番目など）と比べると（図10-18参照）、リモートサーバ（Google）のIPアドレスは同じですが①、ターゲットのMACアドレスが変わっています②。MACアドレスが変わったことから、現在トラフィックはルータを通過する前に、攻撃者を経由していると分かります。

この攻撃は本当に目立たないので、検出が非常に難しくなります。こうした攻撃専用に設定されたIDSの助けを借りるか、ARPテーブルエントリの急な変更を検出するよう設計されたソフトウェアが必要です。解析しているネットワーク上のパケットをキャプチャするためにARPキャッシュポイゾニングを利用することが多くなるでしょうが、諸刃の刃だということを知っておくべきです。

図10-18　ターゲットのMACアドレスの変更は攻撃の成功を示す

リモートアクセス型トロイの木馬

ここまでは、キャプチャを調べる前に、何が起きているのかが分かっているセキュリティ問題を見てきました。攻撃がどのように行われるかを学ぶにはよい方法ですが、あまり現実的ではありません。現実社会のシナリオでは、ネットワーク防御の仕事を担う人々は、ネットワークを行き来するすべてのパケットを調べたりはしません。その代わりに、ある種のIDSを使って、ネットワークトラフィックに異常があれば警告が受けられるようにし、あらかじめ定義した攻撃シグネチャをもとに、さらに調査が行えるようにします。

次の例では、現実世界の解析のように、簡単な警告から始めます。このケースではIDSは次のような警告を出します。

[\*\*] [1:132456789:2] CyberEYE RAT Session Establishment [\*\*][Classification: A Network Trojan was detected] [Priority: 1]07/18-12:45:04.656854 172.16.0.111:4433 -> 172.16.0.114:6641TCP TTL:128 TOS:0x0 ID:6526 IpLen:20 DgmLen:54 DF\*\*\*AP\*\*\* Seq: 0x53BAEB5E Ack: 0x18874922 Win: 0xFAF0 TcpLen: 20

次のステップではこの警告を引き起こしたシグネチャルールを見ます。

alert tcp any any -> $HOME\_NET any (msg:"CyberEYE RAT Session Establishment";content:"|41 4E 41 42 49 4C 47 49 7C|"; classtype:trojan-activity;sid:132456789; rev:2;)

このルールは、16進数のコンテンツに41 4E 41 42 49 4C 47 49 7Cが含まれるパケットが内部ネットワークに侵入してきたら、警告を発するよう設定されています。このコンテンツは人間が読めるASCIIだと「ANA BILGI」と変換されます。検出されると警告が発せられ、CyberEYEリモートアクセス型トロイの木馬（RAT）が存在する可能性を知らせます。RATは被害者のコンピュータ上で秘密裏に実行され、攻撃者へと接続する悪意あるプログラムで、攻撃者は被害者のマシンを遠隔操作することができます。

注　CyberEYEはRAT実行ファイルを作成し、感染したホストを操るための、トルコ生まれの有名なツールです。皮肉にも、見つけたら警告を発するように設定した「ANA BILGI」とは、トルコ語で「基本情報」という意味です。

今度はファイルratinfected.pcapの警告に関連するトラフィックを見てみましょう。このSnort警告は、通常は警告をトリガした単独のパケットのみをキャプチャしますが、ここではホスト間の通信シーケンス全体が関わっています。次のようにSnortルールで定めた16進数ストリングを検索します。

1．[Edit]から[Find Packet]を選択します。

2．[Hex Value]ラジオボタンを選びます。

3．テキストの部分に41 4E 41 42 49 4C 47 49 7Cの値を入力します。

4．[Find]をクリックします。

図10-19に示したように、4番目のパケットのデータ部分で上記のストリングがまず見つかるはずです①。

図10-19　4番目のパケットでまずSnort警告のストリングが見つかる

[Edit]から[Find Next]の選択を何度か繰り返すと、5、10、32、156、280、405、531、652番目のパケットでもこのストリングが見つかります。このキャプチャのすべての通信が攻撃者（172.16.0.111）と被害者（172.16.0.114）との間のものですが、ストリングの一部のインスタンスは別の対話で起こっているようです。4番目と5番目のパケットは4433番と6641番のポートを使っていますが、ほかのインスタンスのほとんどが4433番ポートとほかのランダムに選択されたエフェメラルポートで起きています。[Conversations]ウィンドウのTCPタブを見れば、これら複数の対話の存在が確認できます（図10-20）。

図10-20　攻撃者と被害者との間の3つの対話

色分けすることによって、違う対話を見た目で分類することができます。

1．パケット一覧のペインの上にあるフィルタダイアログで、フィルタ(tcp.flags.syn ==1) && (tcp.flags.ack == 0)を入力、[Apply]をクリックします。これでトラフィックの各対話の最初のSYNパケットが選択できます。

2．最初のパケットを右クリック、[Colorize Conversation]を選択します。

3．[TCP]を選択し、色を選びます。

4．残りのSYNパケットについても同じプロセスを繰り返し、それぞれについて違う色を選びます。

5．終了したら[Clear]をクリックし、フィルタを削除します。

対話を色分けすると、互いがどう関連しているかが見て分かるようになるので、2つのホスト間の通信プロセスを追跡しやすくなります。2つのホストが通信を始めるのが最初の対話（6641番／4433番ポート）なので、ここから始めるのがいいでしょう。対話内のパケットのどれかを右クリックし、[Follow TCP Stream]を選択して、転送されるデータを見ます（図10-21）。

まず攻撃者から被害者に、テキストストリングANABILGI|556が送信されているのが分かります①。すると被害者は、コンピュータ名（CSANDERS-6F7F77）、使用しているOS(Windows XP Service Pack 3)などを含む基本的なシステム情報を応答として送り②、攻撃者に同じストリングBAGLIMI?を送信します③。攻撃者から送られる通信はストリングCAPSCREEN60④のみで、これは6回現れています。

攻撃者が戻したCAPSCREEN60ストリングが気になるので、どこへつながっているのかちょっと見てみましょう。それにはまた検索ダイアログを使い、[String]オプションを指定して、パケット内でこのテキストストリングを検索します。

図10-21　最初の対話から面白い結果が出た

この検索を実行すると、27番目のパケットのストリングで最初のインスタンスが見つかります。この情報が興味深いのは、ストリングが攻撃者からクライアントに送られるとすぐ、クライアントがパケットの受け取りを確認して、29番目のパケットで新しい対話が始まっていることです。

この新しい対話のTCPストリームアウトプットを追跡すると（図10-22）、見慣れたストリングANABILGI|556が目に入り、そのあとにストリングSH|556、最後にCAPSCREEN|C:\WINDOWS\jpgevhook.dat|84972が続いています①。CAPSCREENのあとにファイルパスが指定されていて、そのあとに解読不能なテキストがあるのに気づくでしょう。ここで面白いのは、解読不能なテキストがストリングJFIFの先頭に追加されていることで②、これはGoogle検索をかけるとJPGファイルの先頭に見つかるものでした。

この時点では、攻撃者はこのJPGイメージを転送するために対話を開始したと判断していいでしょう。しかしさらに重要なのは、このトラフィックからコマンド構造が見えることです。CAPSCREENは攻撃者がJPGイメージを転送するために起動したコマンドのようです。実際CAPSCREENコマンドが送信されると、いつも結果は同じです。これを検証するには、各対話のストリームを見るか、次のようにWiresharkのIOグラフ機能を使います。

1．[Statistics]から[IO Graphs]を選択します。

2．フィルタtcp.stream eq 2、tcp.stream eq 3、 tcp.stream eq 4、tcp.stream eq 5、tcp.stream eq 6を、5つのフィルタダイアログにそれぞれ挿入します。

3．[Graph 1][Graph 2][Graph 3][Graph 4][Graph 5]のボタンをクリックし、指定したフィルタのデータポイントを有効にします。

4．Y軸のスケールを[Byte/Tick]に変更します。

図10-23がそのグラフです。

図10-22　攻撃者がJPGファイルのリクエストを開始しようとしている

図10-23　同じ活動が繰り返されようとしていることを示すグラフ

このグラフによれば、各対話に同じ量のデータが含まれていて、同じ長さのようです。これでこの活動が数回繰り返されていると判断できます。

転送されたJPGイメージのコンテンツについては想像がつくでしょうが、これらJPGファイルの中身が実際に見られるかどうか試してみましょう。WiresharkからJPGデータを抽出するには、次のステップを実行します。

1．まず図10-22の前の文章で説明したように、パケットのTCPストリームを追跡します。

2．通信は分離されているので、被害者から攻撃者へ送られたデータストリームのみが見えるはずです。これを行うには[Entire Conversation (85033バイト)]とあるドロップダウンの横の矢印を選択します。172.16.0.114:6643 --> 172.16.0.111:4433 (85020バイト)となるよう、トラフィック（矢印）の向きに注意します。

3．[Save As]ボタンでデータを保存します。拡張子を.jpgとするのを忘れないように。

イメージファイルを開こうとしても開きません。もう1段階残っているからです。8章でFTPトラフィックからファイルを抽出したのとは異なり、このトラフィックは実際のデータにコンテンツを追加しています。この場合、TCPストリームの最初の2行はトロイの木馬のコマンドシーケンスの一部で、JPGを構成しているデータではありません（図10-24）。ストリームを保存すると、この外部データも保存されます。その結果、JPGファイルヘッダを探すファイルビューワは、探しているものとは一致しないコンテンツを見ることとなるため、画像が開かないのです。

図10-24　トロイの木馬が追加した外部データが、ファイルが開くのを妨げている

この問題を治すのは簡単なプロセスですが、バイナリエディタを使ったちょっとした操作が必要です。このプロセスをファイルカービングといいます。図10-25では、WinHexを使ってJPGファイルの最初の数バイトをハイライトしています。バイナリエディタを使ってこれらのバイトを削除し、画像ファイルを保存しましょう。

図10-25　JPGファイルからバイトを取り除く

不要なデータを削除すれば、ファイルが開くはずです。トロイの木馬が被害者のデスクトップのスクリーンキャプチャを乗っ取り、攻撃者へと戻していたことがこれではっきりしました（図10-26）。

図10-26　転送されたJPGは被害者のコンピュータのスクリーンキャプチャだった

これらの通信シーケンスが完了すると、通常のTCPティアダウンによって通信が終了します。

このシナリオは、IDSによる警告をもとにトラフィックを解析するときに、侵入アナリストがたどるプロセスの一例です。

・警告と警告を引き起こしたシグネチャを調べます。

・シグネチャが実際にトラフィックにあることを確認します。

・トラフィックを調べ、侵入されたマシンで攻撃者が何を行ったかを見つけます。

・被害者からさらに重要な情報が漏れないよう、封じ込めを開始します。

まとめ

セキュリティ関連のシナリオでのパケットキャプチャの分割、一般的な攻撃の解析、IDSによる警告への応答についてだけで、1冊の本が書けます。この章では一般的なスキャニングとエニュメレーションのタイプ、中間者攻撃、システムのエクスプロイトについての2つの例、またマシンが乗っ取られた場合に何が起こるかについて説明しました。