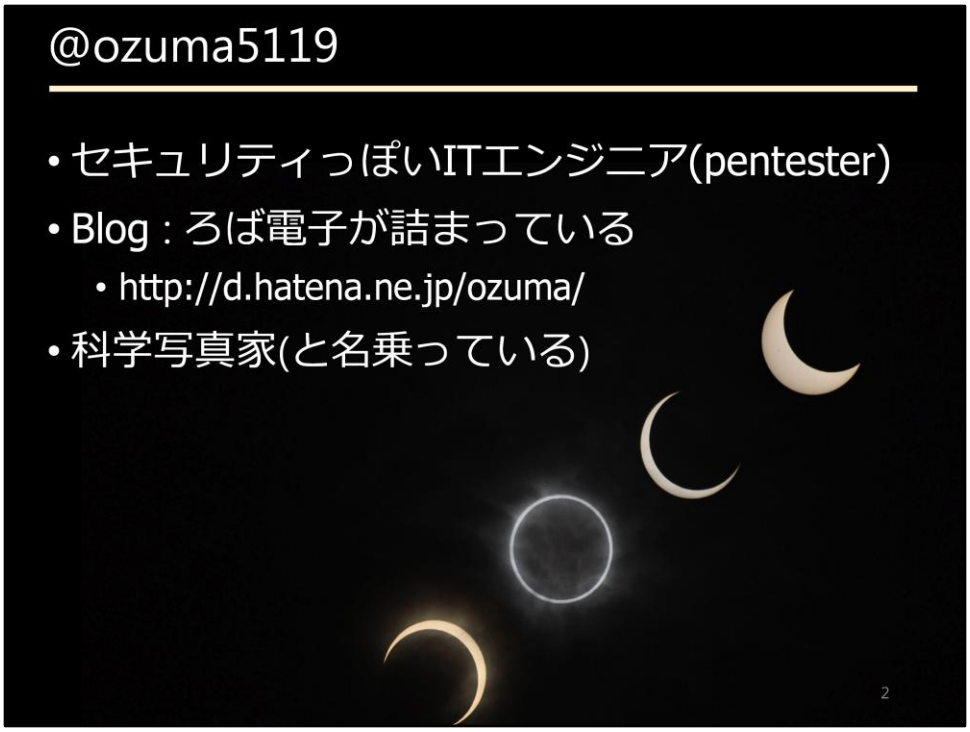
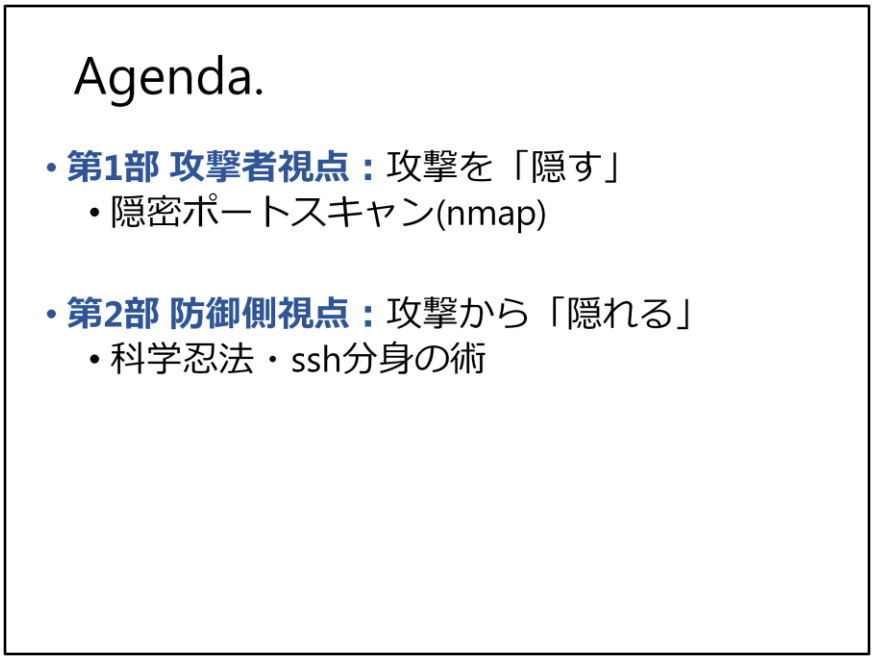
攻撃を「隠す」、攻撃から「隠れる」

と称して発表します。すみだセキュリティ勉強会の主催のozuma5119です。



すみだセキュリティ勉強会を主催しています、ozuma5119と申します。
ふだんは、比較的固めの会社でセキュリティエンジニアをしています。ブログはこちら。

科学写真家というのは、「理科の教科書に載ってるような写真」を撮る人たちです。
こちらは副業ということで、小学生向けの教材の写真とか撮って、ときどき本に載ったりします。



Agendaですが、まず、今回はポートスキャンとnmapに絞って話をします。
「隠す」と一口に言っても、ファイルの見えない領域に隠すとか、パケットやプロトコルのいろんな「隙間」に隠すとか、暗号化で機密情報を隠すとか深堀りすると色々あるのですが……キリが無いので思い切り限定することにしちゃいました。期待していた方、すみません。

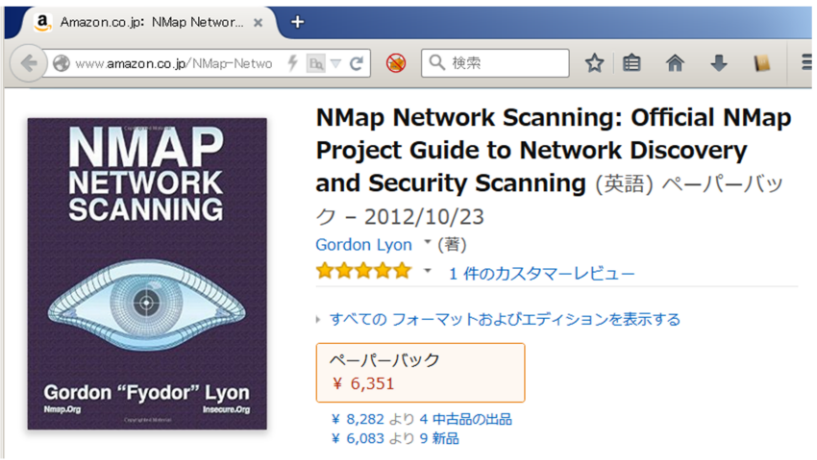
色々ネタを練ってみたのですが、結局ポートスキャンとnmapの話だけになっちゃいました。それ以外の話はいずれ機会があればどこかで。

発表は二部構成です。まず第一部、攻撃を隠すということで、隠密ポートスキャン。どうやってポートスキャンを隠すか？相手に気づかれないようにしてどうやってポートスキャンするか？ってことです。
次に第二部、攻撃から隠れるということで、ポートスキャンからどうやって防御するか？ポートスキャンを食らっても大丈夫なためにはどうすればいいか？
ここでは、大量のsshを起動しているように見せる「科学忍法・ssh分身の術」をメインに紹介します。

というわけでさっそく第一部、攻撃を隠すということで、隠密ポートスキャン。

今回のメインはnmapですので、まずその話を軽くします。ポートスキャンという言葉は、さすがにだいじょうぶでしょうか。対象ホストのTCP/UDPの開放ポートを調べるためのツールです。

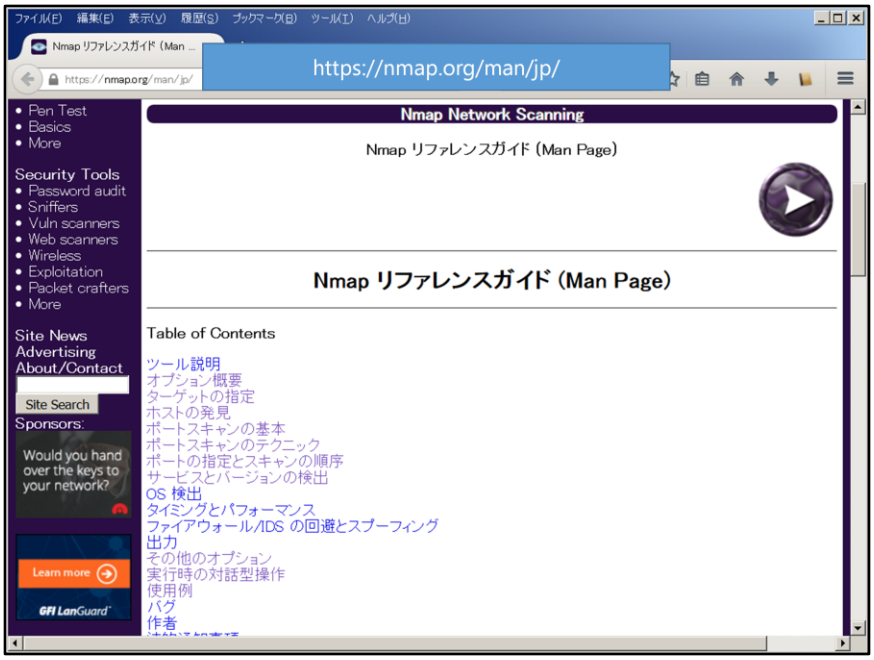
ポートスキャナ - nmap



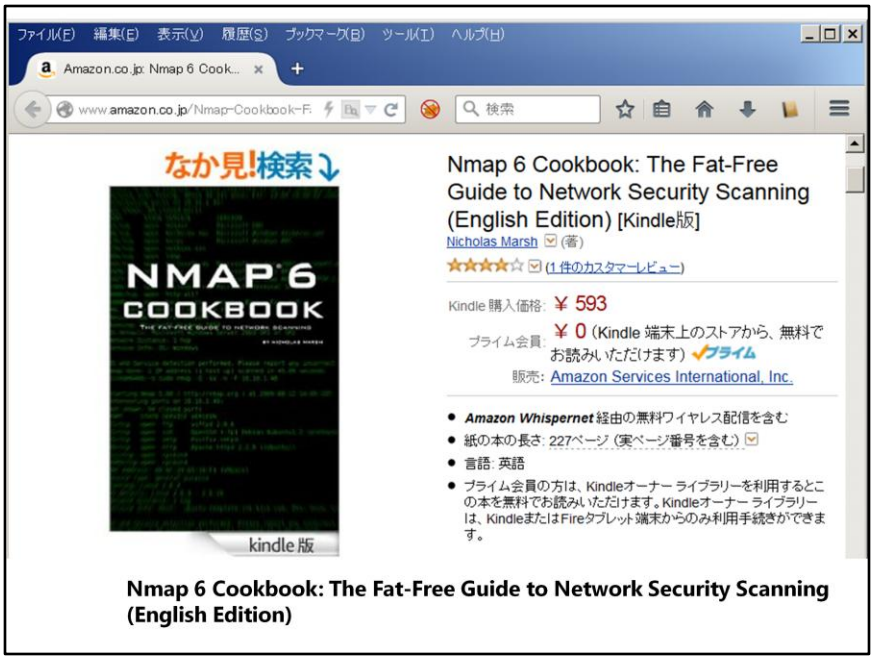
この発表ではポートスキャナとして、有名なnmapを使います。「nmapとは何か?」からはさすがにやりませんので適当にググって調べてみてください。おそらく、世界で一番有名かつよく使われているポートスキャナです。

nmapの書籍は色々出ていますが、一番有名なのはこの公式ガイド「NMAP NETWORK SCANNING」、通称・目ン玉本です。
http://www.amazon.co.jp/dp/0979958717/
nmapの開発者であるGordon “Fyodor” Lyon氏　が書いた本です。2008年なのでちょっと昔の本ですが、過剰なほど細かく書かれており、何回読んでも新しい知識が得られます。

でもこれは400ページ越えでとっても分厚いし、英語が苦手な方もいるでしょう。実は公式Webページには和訳されたマニュアルがあるので、まずそれを読むといいです。
https://nmap.org/man/jp/



公式ページの日本語リファレンスガイド。
https://nmap.org/man/jp/
これ読むだけでも結構な知識が得られます。

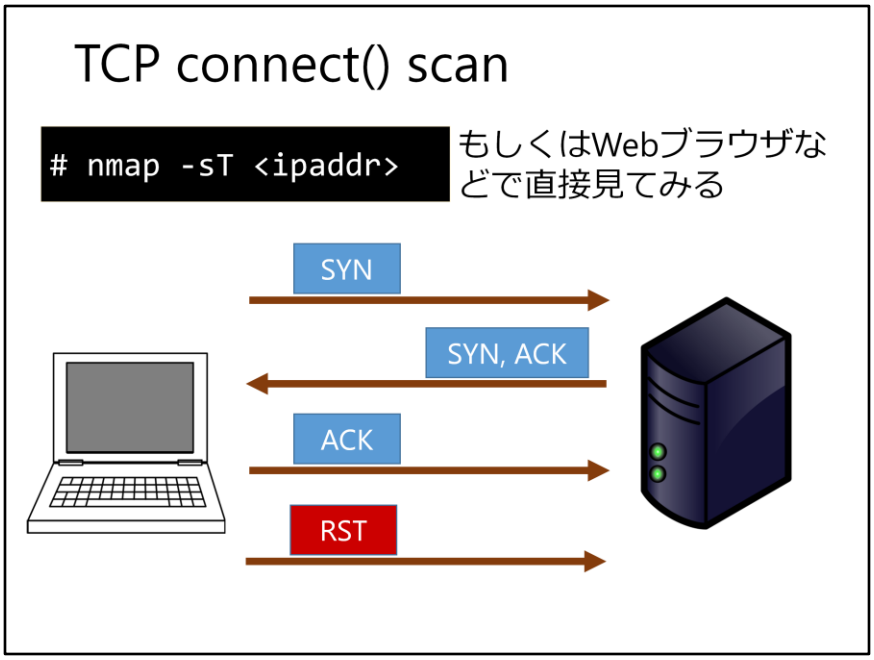


ちなみに、最近出ているこのNmap 6 Cookbookは読みやすくておすすめです。

Nmap 6 Cookbook: The Fat-Free Guide to Network Security Scanning (English Edition) [Kindle版]
Nicholas Marsh (著)
http://www.amazon.co.jp/dp/B00T3P4TA0/

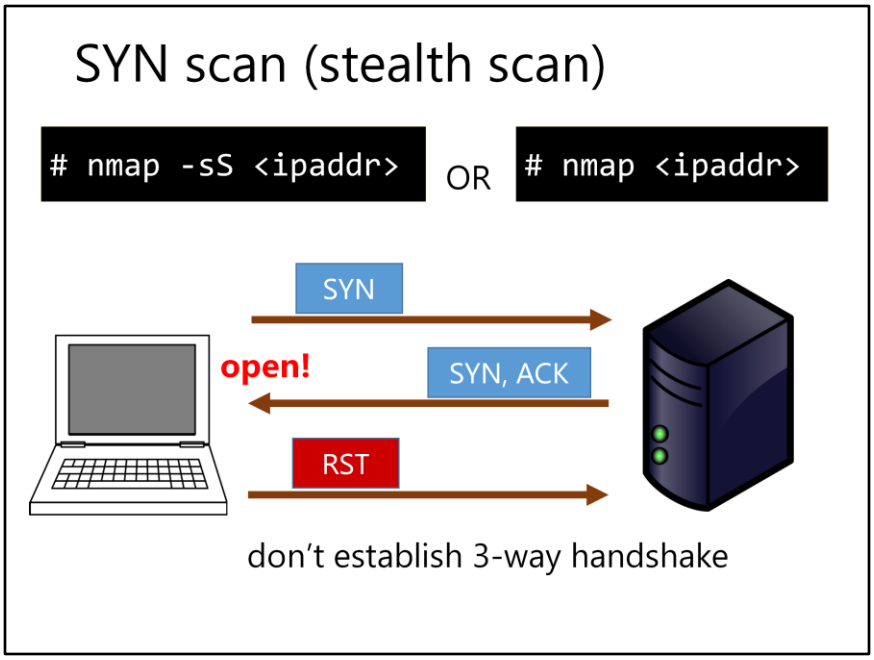
230ページほどと手軽ですし、Kindle版は1000円を切っていてえらい安いです。2,3時間くらいで読めますから、書籍がいい人はまずこれを手に取るといいです。

さて、隠密スキャンと言えば、まずは定番のSYNスキャン。ステルススキャンとも呼ばれます。
何を今さらと思う人も多いでしょうが、初心者の方もいるでしょうからここから復習しましょう。



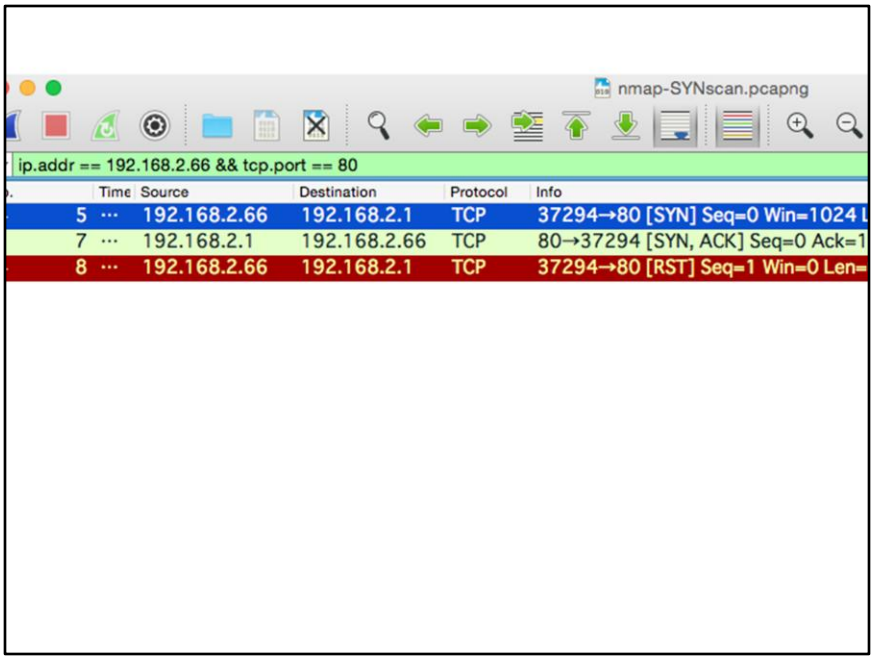
SYNスキャンの前に、もっとも原始的な手法であるTCP connect()スキャンから解説します。

ポートが開放しているかどうかを調べる際、もっともシンプルな方法は「実際に繋いでみる」でしょう。nmapでは、-sTというオプションでこのスキャンが行えます。あるいは対象がWebなら、ブラウザで見てみるというのももちろんアリです。
TCP connect()スキャンでは、対象と3WAYハンドシェイクを成立させます。そのため多くのアプリでは接続のログが残り、検出・追跡されやすいスキャンです。速度もあまり早くありません。



一方、こちらのSYNスキャンは、高速かつ検出されにくいことから、ポートスキャンにおいてもっともよく使われます。
nmapでは-sSとオプション指定するか、あるいはオプション無しのデフォルトでこのSYNスキャンとなります。(ただしユーザの権限によっては生ソケットを扱えずにこのようなスキャンができない場合があり、その場合はオプション無しのスキャンがTCP connectスキャンになります。nmapを使う場合はroot権限であることがほとんどのケースで必要です)。

対象ホストにSYNパケットを送った際、ACKが返れば、その時点でポートはOpenであると分かります。つまり、ポートの開放を判断するには3WAYハンドシェイクは必要ありませんから、ACKが返ったらさっさとRSTを投げて3WAYハンドシェイクせずに切っちまえ、というのがSYNスキャンの考え方です。
また一方、開放されていないポートにSYNパケットを送ると、ホストの設定により若干異なりますが多くの場合はRSTが返ります。これでOpen|Closeが判定できるわけです。



nmapで実際にSYNスキャンしながらパケットキャプチャした様子は、こんな感じですね。
192.168.2.66から、192.168.2.1のポート80にSYNスキャンしています。
SYNに対してACKが返ったので、Openと判断してすぐにRSTを投げつけて、3WAYハンドシェイクが成立しないようにしています。

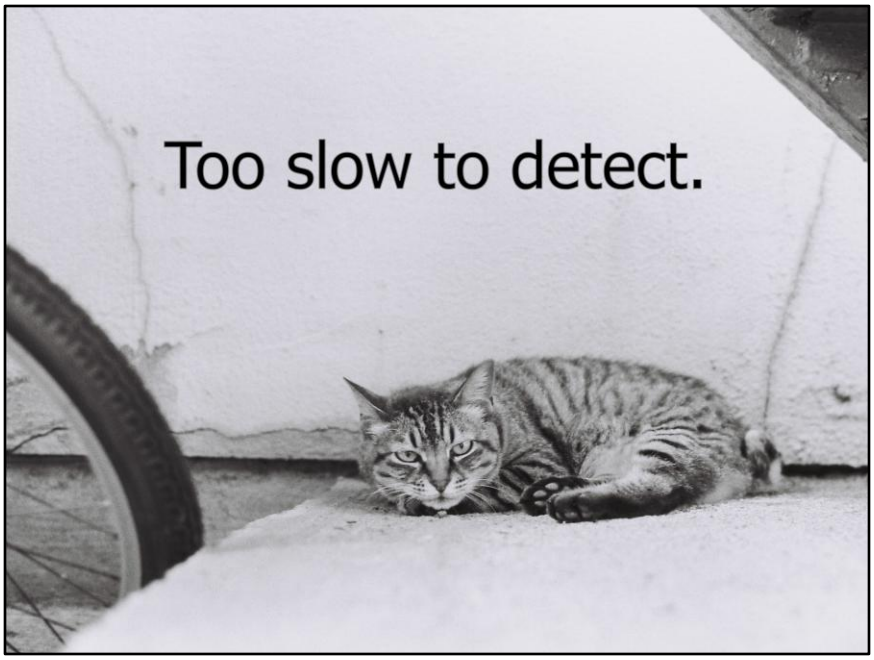
なお、さっきSYNスキャンを検出されにくいと言いましたが、現在においてはあまりに一般的なポートスキャンの手法であるため、SYNスキャンを検出できないネットワークセキュリティ製品ってのは、まぁ、まずありません。

SYNスキャンは、何を「隠して」いるのか

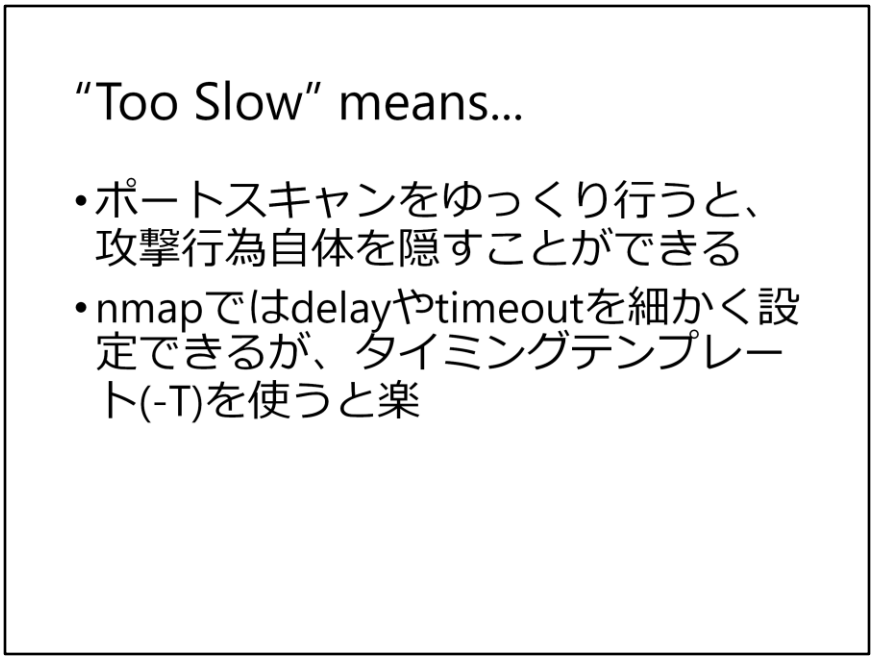
- 通常の手順(3WAYハンドシェイク)を介さないで、
  - **ログに残るのを隠す**
  - 「プロトコル通りに動いている」ことを前提とした検知から攻撃を隠す
- 行為自体を隠しているわけではない

SYNスキャンは通常の手順(3WAYハンドシェイク)を介さないで、
- ログに残るのを隠す
-「プロトコル通りに動いている」ことを前提とした検知から攻撃を隠す
わけです。ポートスキャンの行為自体を隠しているわけではない。

何が言いたいかというと……スキャン行為自体を隠したい場合、もう一工夫欲しいなーってこと。そのことを次に話します。

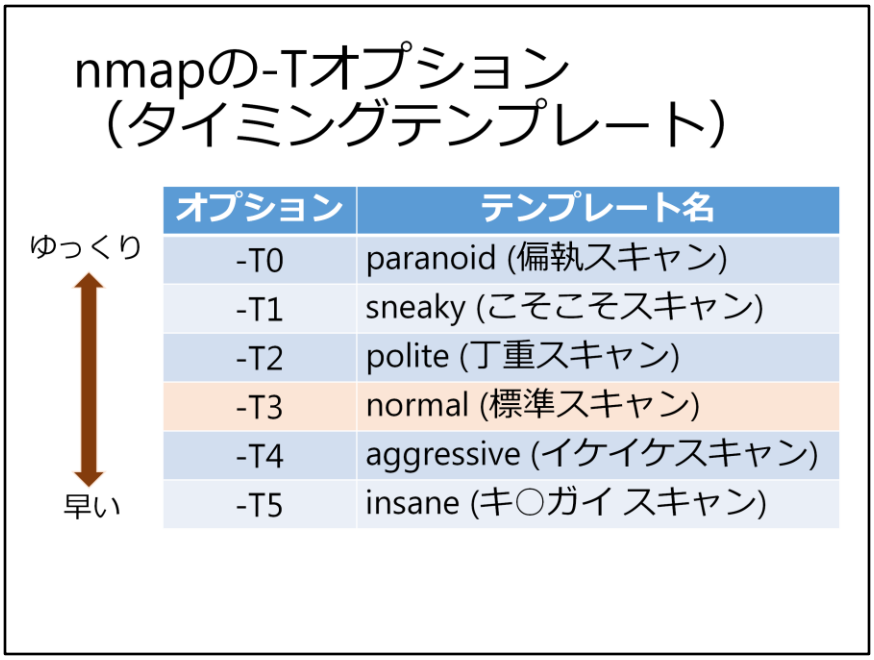


ー工夫加えるうち、一番簡単なのは「ゆっくりやる」ということです。



あとで述べますが、ポートスキャンをゆっくり行うと、事実上、攻撃行為自体を隠すことができます。

では、nmapでポートスキャンをゆっくりやるにはどうすればいいか?
実はnmapでは、並列数やタイムアウト値、delayなどを実に細かく設定できるのですが、一つ一つの値をチューニングしていくのはとても大変です。
そこで、最初から用意されているタイミングテンプレートを-Tオプションで使うと楽です。

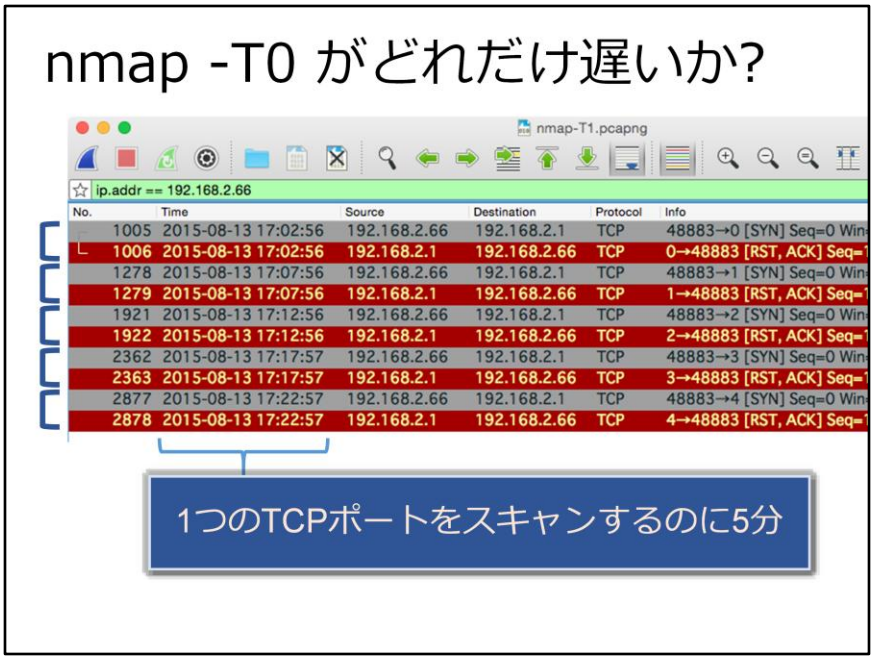


nmapの-Tオプションでは、6つのテンプレートが用意されています。-Tの後ろに数字を指定しますが、数字が小さいほどゆっくり、大きいほど早くなります。
テンプレートには名前が付いており、0が一番遅いparanoidスキャン、5が一番早いinsaneスキャンです。何も指定しない場合は、-T3を指定したものとみなされます。

-T0 : paranoid (偏執スキャン)
-T1 : sneaky (こそこそスキャン)
-T2 : polite (丁重スキャン)
-T3 : normal (標準スキャン)
-T4 : aggressive (イケイケスキャン)
-T5 : insane (キ○ガイ スキャン)
※上記の日本語訳は私の勝手な意訳なので、よそで言っても通じません。

実用上、ほとんどの場合は-T3でいいのですが、敏感なファイアウォールがいてSYNスキャンに干渉されているなーという場合、-T2でスキャンすると上手くいくことがあります。

なお、-T0は本当にメチャクチャ遅いので、実用性は全くありません。実例を見てみましょう。



これが、-T0をつけてポートスキャンをした際のパケットキャプチャです。ここではTCPの0番から4番までスキャンしていますが……。
見ての通り、なんとひとつのポートをスキャンするのに5分間隔を開けています。ということは、1000ポートをスキャンするだけでも5000分……つまり83時間、約3日半かかることになります。

でも、スキャンされる側はどうでしょう。これほどゆっくりスキャンされては、相手もスキャンされていると気が付くことはおそらく無いでしょう。

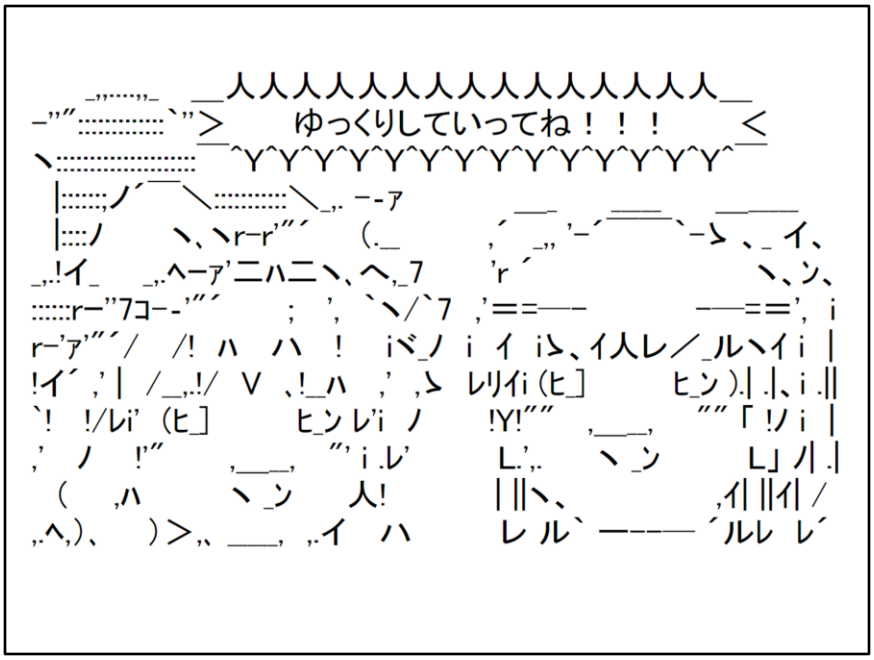
Threshold (閾値)

- 侵入検知システムでは、Threshold以上の事象を「検知」としてアラート
  - これはインターネットに限った話ではない。敏感すぎる防犯センサなど…
- そしてThreshold以下の事象は、
  「存在しない」

多くの侵入検知システムでは、Threshold(閾値)以上の事象を「検知」としてアラートをあげます。
これは、インターネット上には攻撃トラヒックがあふれているので、ちょっとした1パケット一つ一つに全部アラートをあげていてはひっきりなしになってしまうからです。

このような仕組みはインターネットに限った話ではありませんね。敏感すぎる防犯センサなどは、1回警告が出ただけでは誰も信用せず、何度も続けて鳴って初めて「何か異常かな?」と思われる、なんてのはよくあることです。

こうしてみると、Threshold以下の事象は、実質的に「存在しない」と見なせるわけです。これにより、ポートスキャンの行為自体を隠すことができます。

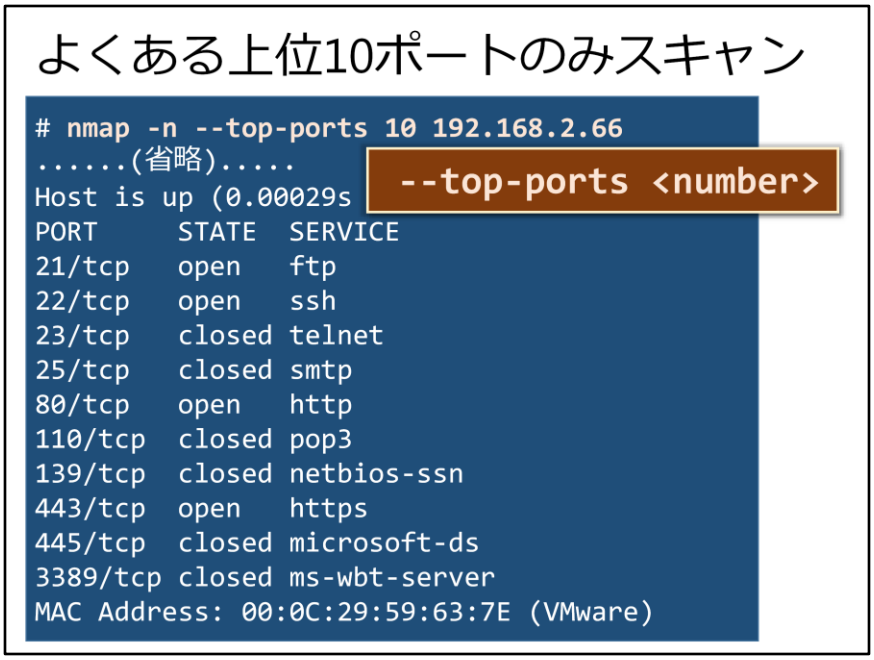


こうして、ポートスキャンをゆっくりやることでスキャン自体を隠すことができます。
-T0スキャンは非現実的と思うかもしれませんが、攻撃者はスキャナをしかけて3,4日放っておくだけで良い。本気で攻撃したい対象なら、そのくらいの手間はかけるでしょう。

攻撃側が有利と言われる理由のひとつは、こんなところにあります。

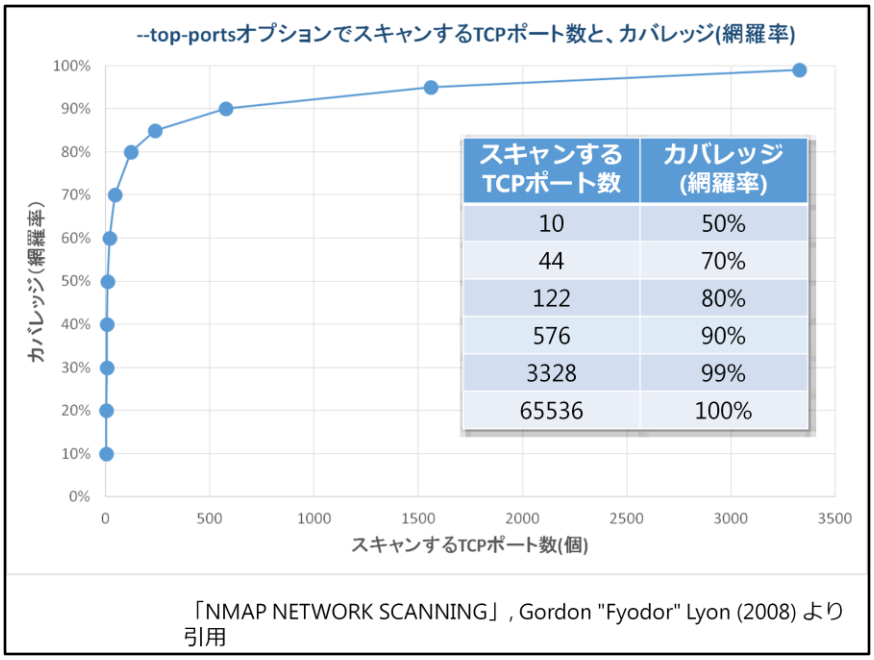
遅くするだけでは脳が無いので、もうちょっとクレバーなアプローチも取りたい。
ここでは、全ポートをスキャンする必要は無く、ほんの一部のポートをスキャンするだけ十分だよということを示します。

当たり前ですが、65535個のポートスキャンをするより、ほんの100ポートだけスキャンする方が、見つかりにくい。



ポートを絞ってスキャンする場合には、「一般的によく開いているポート」を狙うと効率的です。
nmapにはそのために便利なオプションが用意されており、それがこの --top-portsというオプションです。引数に数値を指定することで、上位nポートのポートスキャンできます。ここでは、よくある上位10ポートのスキャンをしていますね。

ちなみに、このランキングは以下の通りです。
1位 80/tcp (http)
2位 23/tcp (telnet)
3位 443/tcp (https)
4位 21/tcp (ftp)
5位 22/tcp (ssh)
6位 25/tcp (smtp)
7位 3389/tcp (rdp)
8位 110/tcp (pop3)
9位 445/tcp (smb)
10位 139/tcp (netbios)

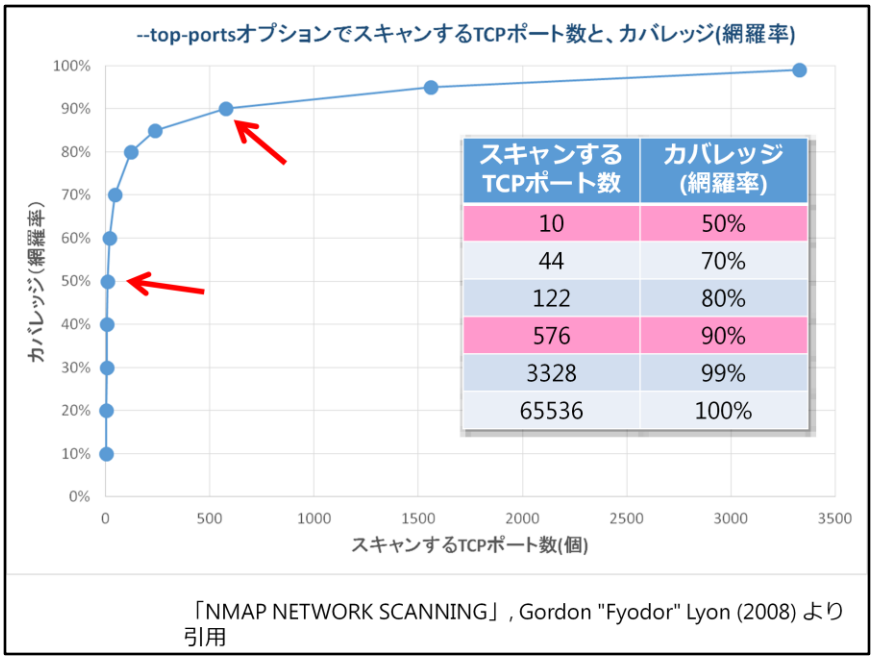


「NMAP NETWORK SCANNING」, Gordon "Fyodor" Lyon (2008) より引用

では、上位何ポートくらいスキャンすれば十分だと言えるでしょうか。

これは、Nmapのガイドブック（目ん玉本）に載っている、いかに効率的にスキャンをするか？ というsectionに登場するデータでグラフを描いたものです。
この書籍は2008年のものなので現在は細かい数字は違ってくるでしょうが、言いたい結論は変わらないはずです。

このグラフは、横軸がtop-portsオプションでスキャンするTCPのポート数（個数）、縦軸がカバレッジです。（Nmap本ではEffectivenessという単語を使っていますが、ここではカバレッジとしました）。
ここでいうカバレッジは網羅率、つまり対象ホスト上で開放している全てのポートがスキャンできたときを100%としたときの、実際にスキャンされるポートの割合です。
多くのホストをスキャンして、ポートごとの「開放率」とでも言うものを統計的に出しておけば、このようにカバレッジも計算できるわけです。



「NMAP NETWORK SCANNING」, Gordon "Fyodor" Lyon (2008) より引用

たとえば、50%のカバレッジを出すには、10個のポートをスキャンするだけで良いわけです。これは想像よりもはるかに高い値ではないでしょうか。
90%のカバレッジを出すにも、500個ちょっとのポートをスキャンすれば十分です。このように、よく使われるポートは非常に「偏り」がありますから、必要最小限のポートのみに絞ってスキャンしましょう。探査行為が少なければ少ないほど、攻撃が検知される可能性も低くなります。

また、99%のカバレッジを出すには3000個ちょっとのポートをスキャンすればいいのに、そこからさらにカバレッジ100%を目指すには、当然のことながら全TCPポート(0-65535)の65536個をスキャンしなければいけません。これより、全ポートのスキャンはいかに非効率的であるかが分かると思います。--top-portsオプションを積極的に使って、最小限のスキャンで攻撃しましょう(するなよ)。

さらなる隠密スキャン

- キーワードだけ紹介
  - TCP Idle Scan (-sI)
  - FTPバウンススキャン (-b) [古典]
  - デコイ(囮) (-D)
  - MACアドレス偽装 (--spoof-mac)

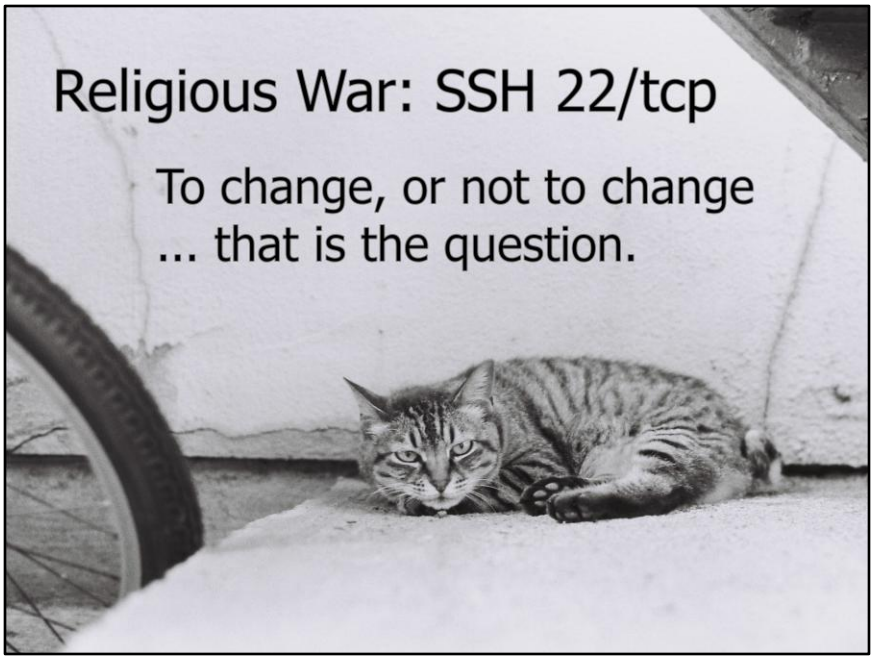
他にも隠密スキャンに使えそうな技はあるんですが、時間も無いのでキーワードだけあげておきます。
興味のある方は各自調べてみてください。

・TCP Idle Scan (-sI)
・FTPバウンススキャン (-b) [古典]
・デコイ(囮) (-D)
・MACアドレス偽装 (--spoof-mac)

では、続いて第2部～。

今度は防御側の話。ポートスキャンから隠れる方法を色々考察します。
題して、「科学忍法・ssh分身の術」。

ちなみに科学忍法という言葉、誰でも知ってる元ネタを……と思って使ったのですが、
この会場の年齢的に実は半分以上の人が知らない気がしています。
が、まぁこのまま進めます。(ガッチャマン)。



話が拡散しそうなので、今回の防御側のテーマはsshのポートをどうやって隠すか?に絞ってお話します。
SSHのポートを22/tcpから変えるべきかどうか?ってのはもはや宗教戦争じゃないですか。今日の勉強会で、その議論にようやく終止符を打って見せますよ!

sshd shouldn't use 22/tcp?

- **変えるべき派**
  - 攻撃されにくい。攻撃対象として選定されにくい（正当派）
  - 攻撃ログが劇的に減るからやった方が良い（ピンポンダッシュ嫌だよ派）
  - 多くのドキュメントで変えることが推奨されているから、変えたほうがいい(流され派)

まず、sshdのポートを22/tcpから変えるべき派の主張を並べてみました。
・攻撃されにくい。攻撃対象として選定されにくい（正当派）
・攻撃ログが劇的に減るからやった方が良い（ピンポンダッシュ嫌だよ派）
・多くのドキュメントで変えることが推奨されているから、変えたほうがいい(流され派)

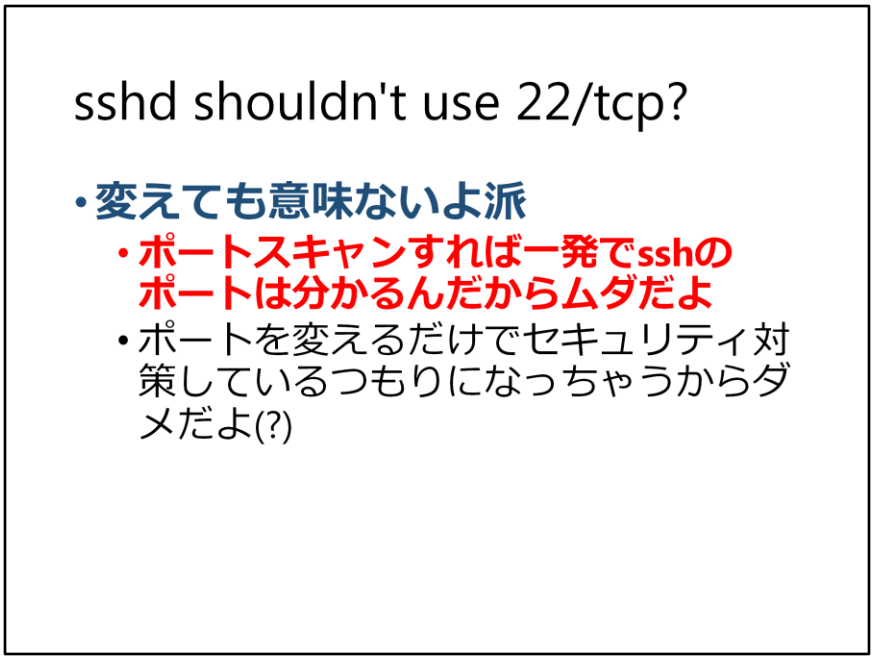
私も、ポート番号を変えるなんてすぐできるし、それでピンポンダッシュが減るんだからゴタゴタ言ってないでやればいいじゃん、派です。

sshd shouldn't use 22/tcp?

- **変えても意味ないよ派**
  - ポートスキャンすれば一発でsshのポートは分かるんだからムダだよ
  - ポートを変えるだけでセキュリティ対策しているつもりになっちゃうからダメだよ(?)

変えても意味ないよ派の主張もまとめてみました。
・ポートスキャンすれば一発でsshのポートは分かるんだからムダだよ
・ポートを変えるだけでセキュリティ対策しているつもりになっちゃうからダメだよ(?)

なんてことがよく言われていますよね。



で、意味ないよ派の主張のキモはここかなと思います。「ポートスキャンをすれば一発で分かるんだから意味ない」。
でも、それ本当に一発で分かる？

ポートスキャンをしても一発で分からないようにすれば、私の勝ち、ということですか。
なお、ここで宗教戦争をはじめるといくら時間があっても足りないので、ポートを変えるべきかどうかのツッコミはお願いだから今日はしないでください。。。

ポートスキャンをしにくくなる、一つ目の方法は簡単です。
プリンタポート……具体的には9100/tcpにするだけで隠れることができるんです。

```
# nmap -n -sV -p1-65535 192.168.2.66
....(snip)....
PORT     STATE SERVICE    VERSION
9097/tcp open   ssh       OpenSSH 5.3 (protoc
```

↓見えてないぞ!!!!!

```
# nmap -n -sV -p1-65535 192.168.2.66
....(snip)....
PORT     STATE SERVICE    VERSION
9100/tcp open  jetdirect? Excluded from versi
```

論より証拠ということで、nmapで実際にスキャンした様子です。
ここではバージョン情報を取得する-sVスキャンをしていますが、見ての通り、
9100/tcpでsshdを起動している方はVERSIONが「Excluded from version scan」
と表示されており、情報取得ができていません。

これでは攻撃者も、ここにOpenSSHがいるとは分かりませんね。

Nmap Reference Guide

--allports (Don't exclude any ports from version detection)

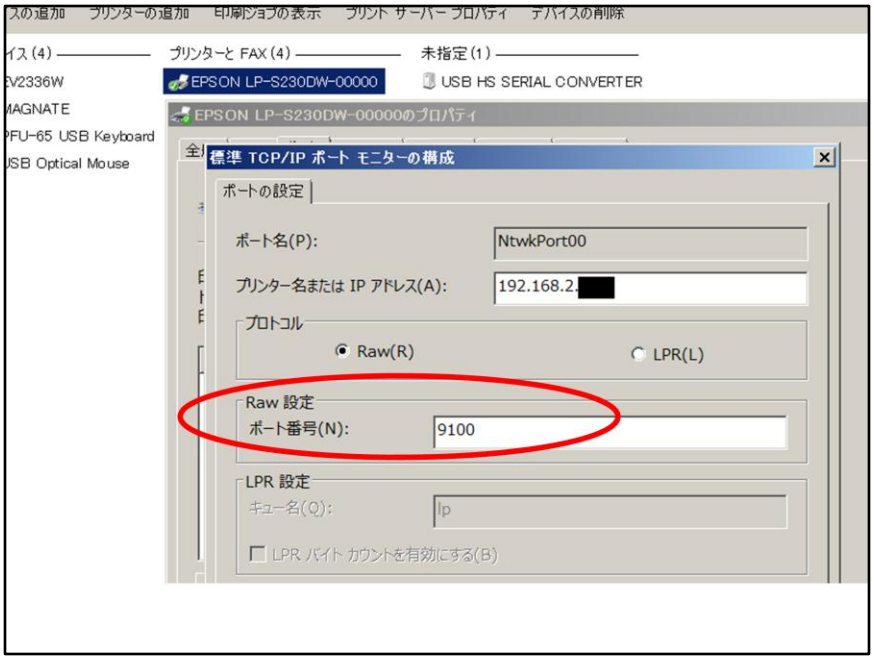
By default, Nmap **version detection skips TCP port 9100** because some printers simply print anything sent to that port, ...(snip)....

https://nmap.org/book/man-version-detection.html

なぜこんなことが起こるのか……これはnmapのWebページ、オプションの解説のところに書いてあります。
9100/tcpはプリンタが使うポートで、一部の機種ではここに送られたデータをそのまま印刷するという機能を持ちます。すると、バージョン情報取得のためのスキャンをすると大量の印刷をしてしまうおそれがあるため、デフォルトではこのポートに対しては-sVが効かないようになっています。すなわち、SSHが動いているかどうかも分かりません。

なおドキュメントには9100/tcpのみと書いてありますが、実際は9100/tcpから9107/tcpの8つのポートがこの挙動を示します。

先ほどのマニュアルに書かれているように、実際に--allportsオプションを付加したのがこちらです。
今度は正しくVERSIONが取得できて、「OpenSSH 5.3 (protocol 2.0)」と表示されていることが分かります。

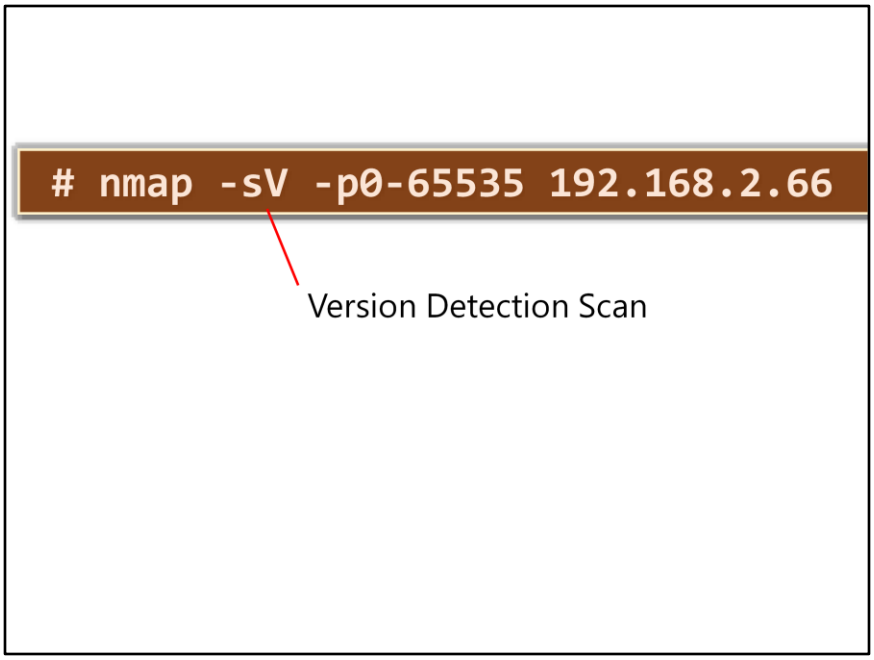


この9100/tcpでのプリンタポートは、過去の遺物ではありません。
これはWindows 7のコントロールパネルから、うちで使っているEPSONプリンタの設定を見ているところですが、ポートの設定で「Raw設定」というのを選ぶとこの9100/tcpが出てきます。

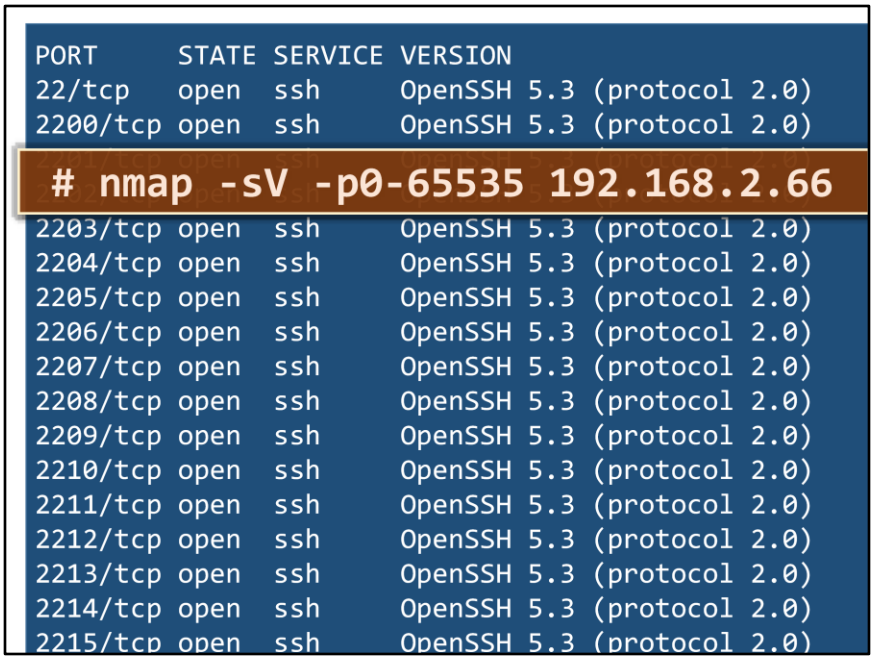
- sshdは9100-9107/tcpへ隠すと見つかりにくい（気がするよ）

このように、9100から9107までの8つのポートは、nmapでバージョン検出がしにくいので、比較的「見つかりにくい」ポートです。
ただ、このポートならではの問題があるかもしれないので、実際に9100/tcpでsshdを起動して運用するまではやったこと無いです。でも、いまどき9100/tcpで印刷する人いないだろうし、検討してみる価値はあると思います。

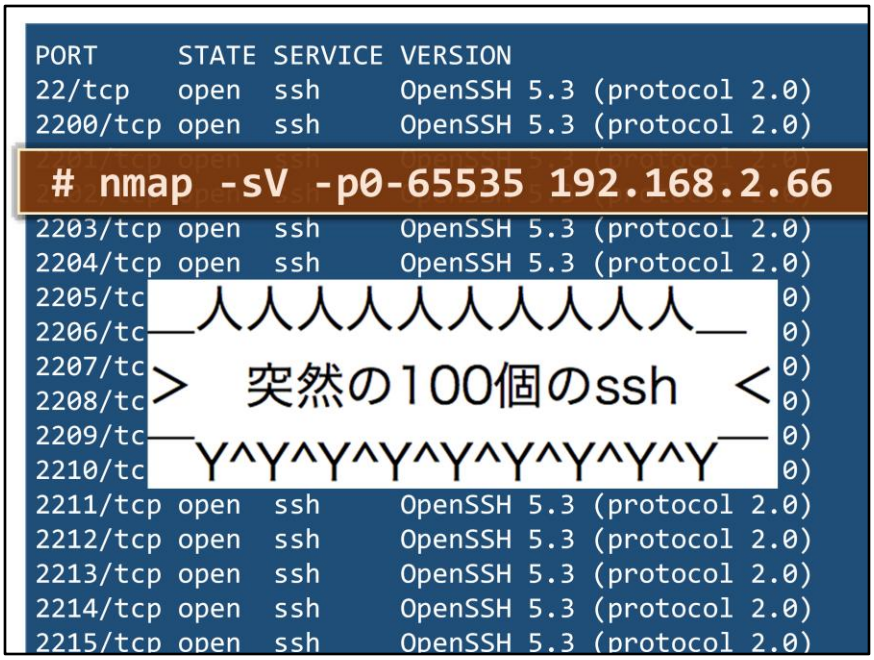
続いて、ようやく本日のメインテーマにやってきました。
これより、科学忍法・ssh分身の術をご覧いただきます。



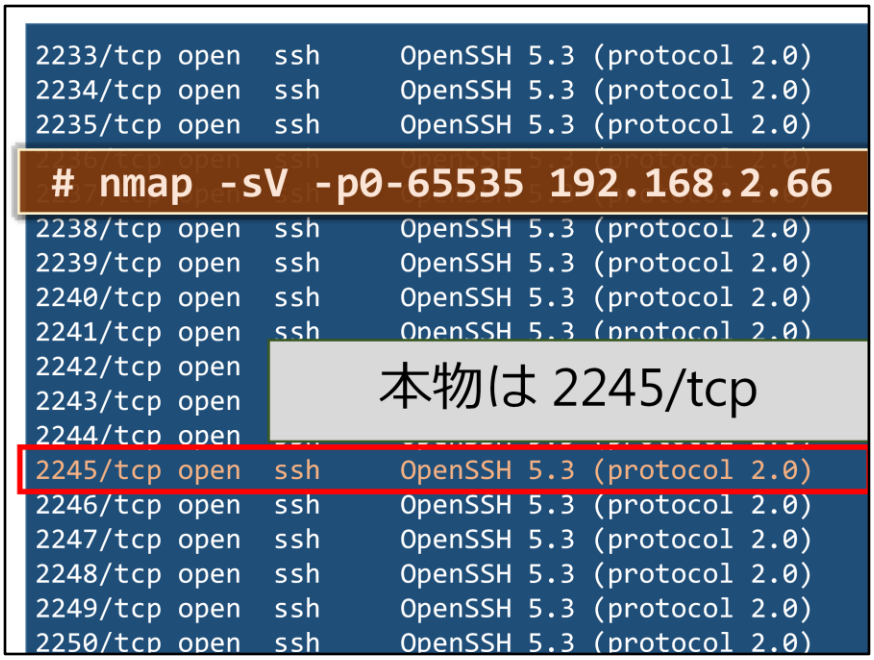
攻撃者が、0から65535までのフルポートスキャンをかけてきたとしましょう。
-sVオプションでバージョン情報取得も行い、sshのポートを探す気マンマンです。おそらく、「sshのポート番号なんて変えても、ポートスキャンすれば一発で分かるんだから意味ないよ」としたり顔で主張していた、あいつによる仕業に違いありません。



ところがこのサーバ、ポートスキャンすると大量のポートが開いています。
大量に「OpenSSH 5.3 (protocol 2.0)」が見えていますね。
22/sshと、2200から2299までの99個、全部がOpenSSHに見えます。



ポートスキャンをしかけてきた攻撃者は、突然の100個のsshにびっくり仰天するに違いありません。
これぞ、科学忍法・ssh分身の術!



実はこのサーバ、ホンモノのsshdは2245/tcpで起動しています。
でもnmapによるバージョンスキャンだけでは、これが本物だと判断することは、(たぶん)不可能です。攻撃者もどのポートを辞書攻撃すればいいか、迷ってしまうことでしょう。

というわけで、sshのポート番号を変えることについて、
「ポートスキャンすれば一発でsshのポートは分かるんだからムダだよ」
という主張を崩すことに成功しました。お前、それサバンナでも同じこと言えるの？という感じです。

これで、sshのポート番号は22/tcpのデフォルトから変えた方がいい、という説を強化できました。

Decoy (囮)

- "Decoys are **fake** military equipment that are intended to **deceive** the enemy."
  - Wikipedia [Decoy] より引用

このような防御手法は、サイバーセキュリティの世界ではあまり使われませんが、軍事行動ではよく見られます。
これはデコイ（Decoy；囮）を用いた手法と分類できます。

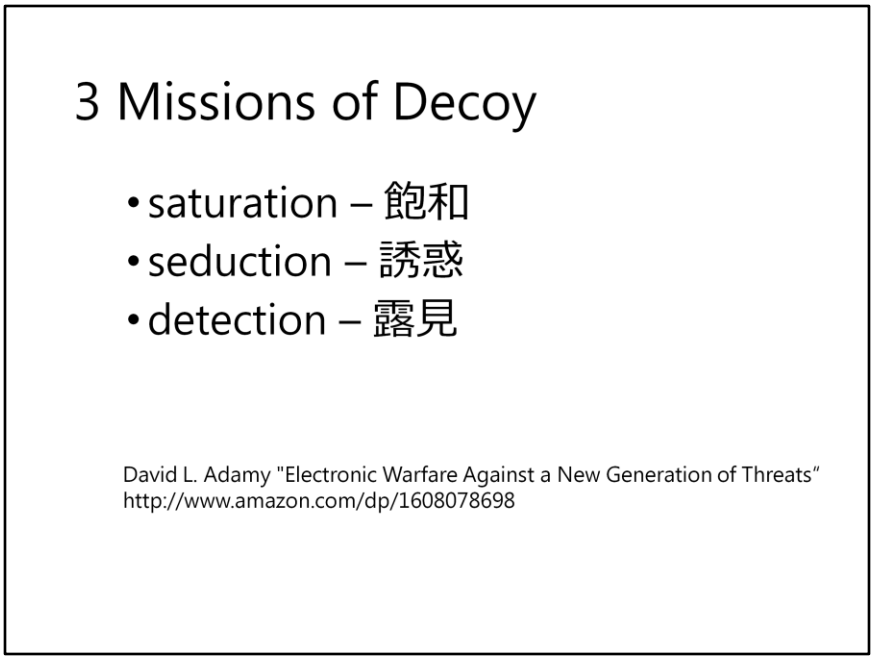
デコイとは何か。Wikipediaから引用しましたが、軍事用語で、敵を欺くための、fakeのことを指します。

(画像は著作権フリーのものをWikipediaより引用)

兵器のデコイは、古代から現代まで広く例があります。実際、第二次世界大戦でもエル・アラメインの戦いでハリボテ戦車が使われました。ハリボテ戦車は、敵の索敵の錯覚や、空爆のダミー目標にさせることができるなど、いいことだらけです。北朝鮮もハリボテ戦車を好むことが知られています(?)。

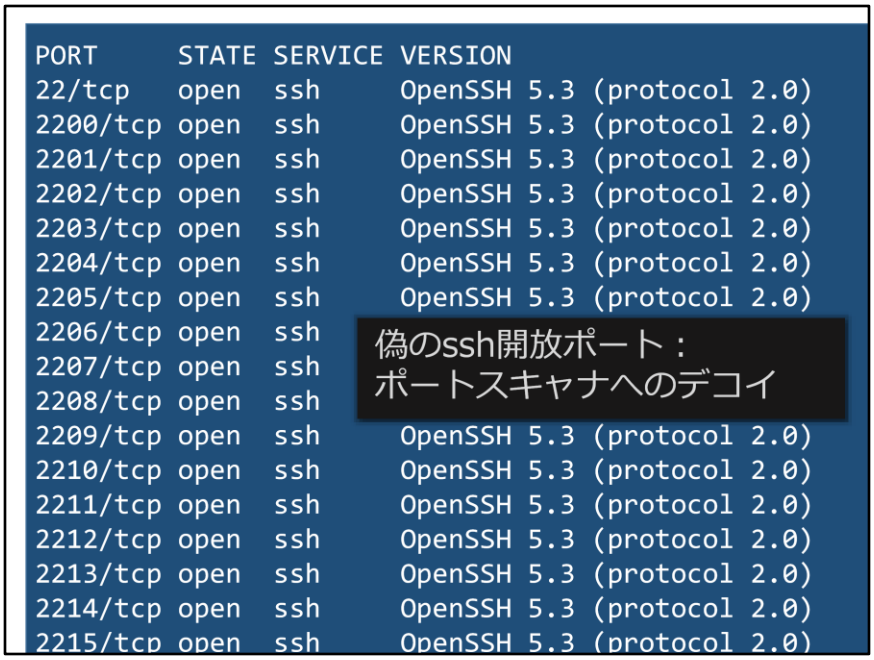
この写真は、C-130という輸送機(日本の自衛隊も持っています)です。何か爆弾をまいて攻撃しているように見えますが、実はこれ、攻撃じゃなくて防御・回避行動です。ここでは、フレアを展開しています。

赤外線誘導ミサイルにロックオンされたら、このフレアを展開します。すると、誘導装置がこれらのデコイに向かうので、輸送機本体が撃墜されるのを防ぐ。。。というワケです。なお、フレアはレーダー誘導には効果が無いため、同時にチャフという金属リボンのようなものを展開することもあります。
それと、最近のミサイルはCCDカメラで画像認識しながら追尾するので、特に空対空ミサイルではフレアやチャフの効果は薄れているそうです。まぁ私は軍事オタクじゃないのでその辺の細かい話は割愛します。

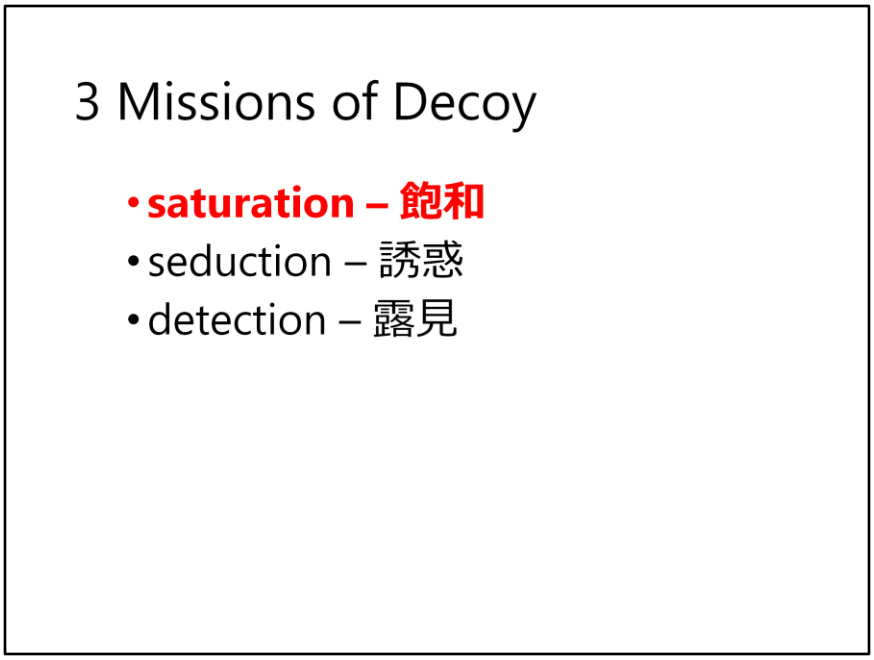


これは軍事関連の本から持ってきた分類ですが、デコイには3つのmissionがあります。
saturation：飽和。例えばレーダー上に大量のターゲットを出現させることで敵の処理能力を超えさせる。
seduction：誘惑。敵の攻撃を、ダミーであるデコイに向かわせる。典型的な「おとり」の例です。
detection：露見。これは、アニメでよく見る風景で言うと……敵がどこにいるのか全く分からない状態で、物を投げると敵がそこに銃弾を浴びせます。その弾が飛んできた位置から、敵の潜伏箇所を特定する。これは近現代だけでなく、古代でも弓矢による攻撃へ、デコイで同じ対策が行われていたことでしょう。

David L. Adamy "Electronic Warfare Against a New Generation of Threats“
http://www.amazon.com/dp/1608078698



先ほど大量に見えていたOpenSSHの開放ポートですが、これらはダミーですから、
実質デコイと言えるでしょう。
つまり、ポートスキャンへ対抗するためのデコイです。



このssh分身の術で狙っているのは、デコイのsaturationによる効果です。
結局、見えている全ポートについて、全部辞書攻撃をすれば、実質1ポートしか開いていないときと同じ攻撃はできることにはなります。しかし100個もポートが開いてれば、攻撃者のリソースをかなり割いてしまうことは確かでしょう。諦めてくれる可能性も高いかもしれません。

反論：防御になっていない?

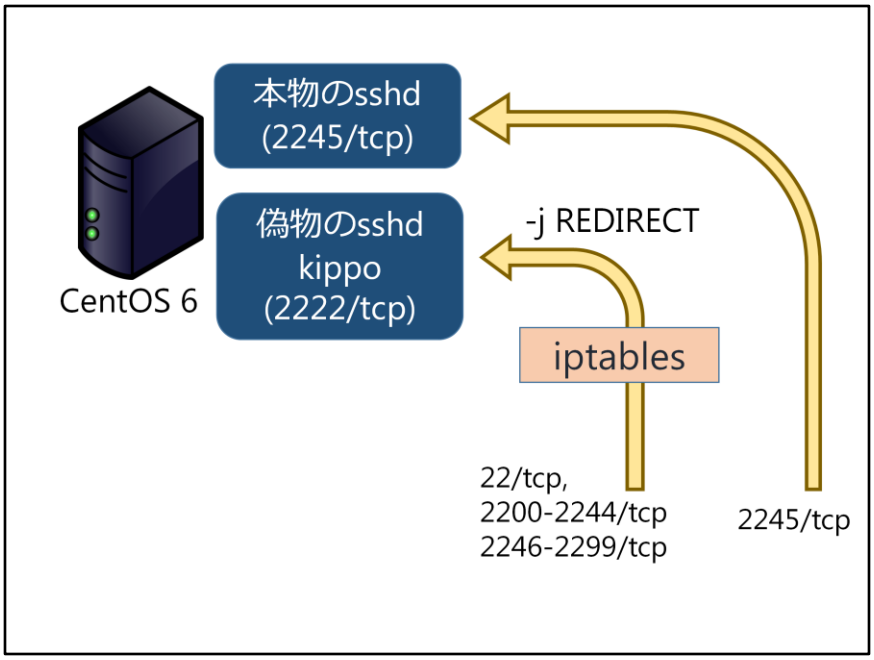
- 遅延装置は、鍵をかけたドアと同じ
  - 破られない錠前は無いが、侵入を遅らせることはできる
- 十分な遅延効果があれば、攻撃を検知してブロックするまでの時間も十分に取ることができる

なお、予想される反論について、先に予防線を張っておきます。

そもそもこれ、防御になってないんじゃないの? という意見があるかもしれません。回答としては、防御装置としての一定の効果はあるはずです。

ここで作ったデコイは、いわば遅延装置です。これは、鍵をかけたドアと同じです。鍵というのは、セキュリティでは遅延装置として考えられます。つまり、破られない錠前は無いが、侵入を遅らせることはできるわけです。
十分な遅延効果があれば、攻撃を検知してブロックするまでの時間も十分に取ることができます。また、現実的に何年もかかってしまうレベルならば、それは実質的に防御になっているとみなして良いでしょう。

ということで、それでは実際に科学忍法・ssh分身の術を皆さんも作っていただきたいと思います。
簡単なのでここで解説します、ガッチャマンのつくりかた。



中身を開かすと、シンプルかつ簡単な構成です。
ここではまず、本物のsshdを2245/tcpで動かしています。
一方、ニセモノのsshdを2222/tcpで動かしています。

そして22/tcpと、2200/tcpから2299/tcpまで(2245/tcpを除く)の通信は、iptablesのREDIRECTを使って2222/tcpへリダイレクトしています。タネを知れば、「なーんだ」って感じですね。

ニセモノのsshdは、なんとなくsshdっぽく振る舞えばなんでもいいんですが、ここでは使い慣れているkippoを使っています。これはsshハニーポットで、sshサーバとして振る舞って様々なデータ取得が可能なものです。今回は、単なるダミーsshdとして利用しています。

kippoについて詳しくは、以前にblogに書いたことがあるのでそちらを参照してください。
> sshハニーポットをkippoで作ってみる
> http://d.hatena.ne.jp/ozuma/20130829/1377703104

なお今回はkippoを使いましたが、例えばホンモノのOpenSSHをポートを変えてもう一つ起動し、そちらは絶対にログインできない設定にしておく、とかでも良いと思います。

本物のsshd(2245/tcp)以外は、偽物のsshd(kippo; 2222/tcp)へ

```
# iptables -A PREROUTING -t nat -i eth0 -p tcp --dport 2200:2244 -j REDIRECT --to-ports 2222
# iptables -A PREROUTING -t nat -i eth0 -p tcp --dport 2246:2299 -j REDIRECT --to-ports 2222
```

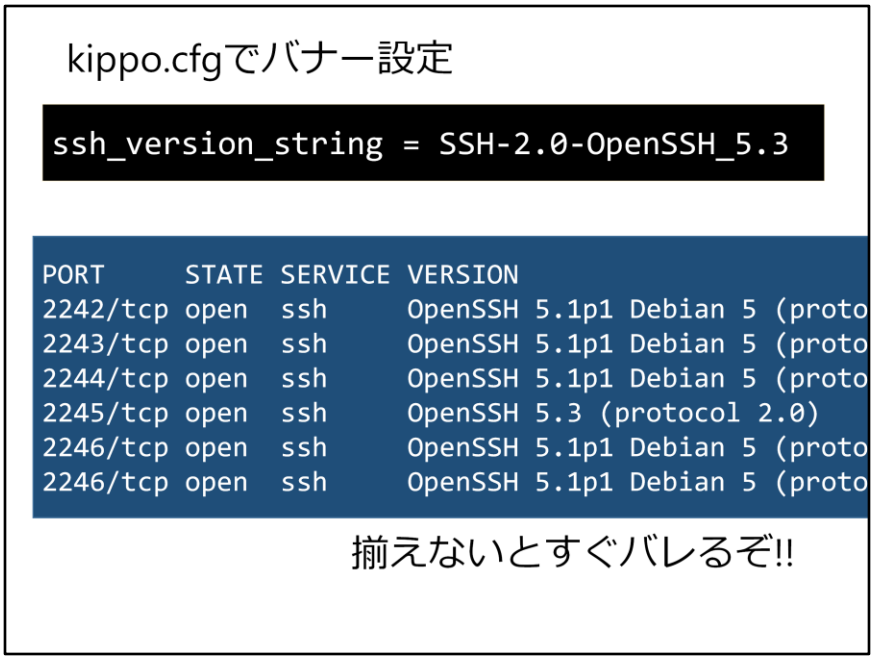
- ループは発生しないため、2222/tcpの特別扱いは不要。
  ※2222/tcpを2222/tcpへ-j REDIRECTしてもだいじょうぶ

ポートをリダイレクトするには、iptablesの、natテーブルに書いてやればいいわけですね。

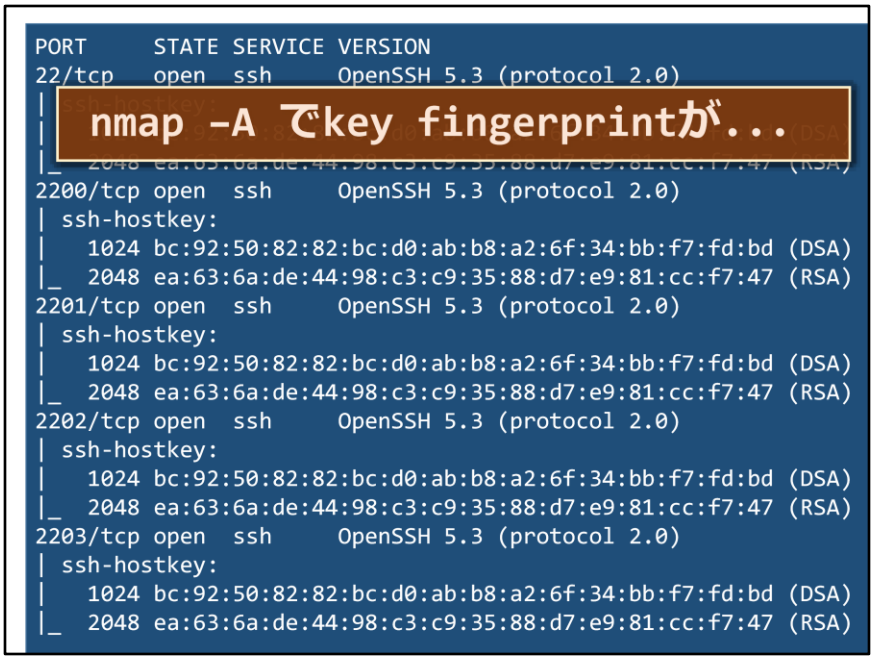
```
# iptables -A PREROUTING -t nat -i eth0 -p tcp --dport 2200:2244 -j REDIRECT --to-ports 2222
# iptables -A PREROUTING -t nat -i eth0 -p tcp --dport 2246:2299 -j REDIRECT --to-ports 2222
```

iptablesの基礎知識は皆さん大丈夫でしょうか。これは、「natテーブル」の「PREROUTINGチェイン」で、「REDIRECTターゲット」を使う、という言い方をします。--dportなどでポート範囲を指定するには、コロンを使います。そこでここでは、2245/tcpをのけて書いています。

ちなみにiptablesのREDIRECTでループは発生しないので、2222/tcpを別扱いして除外する必要はありません。
natテーブルのPREROUTINGチェインが2度以上評価されることは無いので。

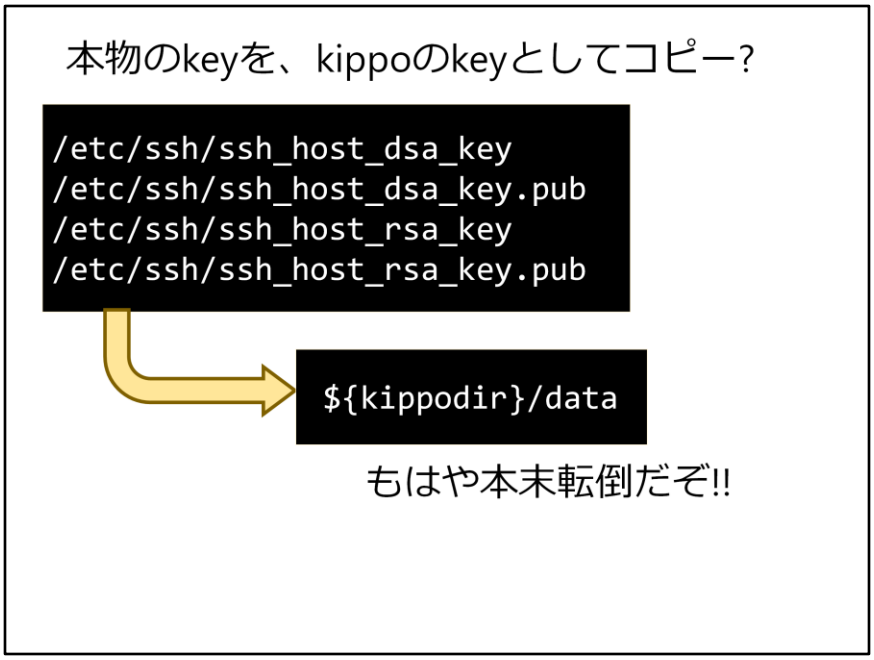


また、kippoのデフォルトではDebianのsshのバナー表示をするので、CentOS 6のOpenSSHとは大きく異なるバナーが表示されます。
これでは本物のバナーと違うのですぐバレてしまいます、kippo.cfgの`ssh_version_string`で適宜設定しましょう。



また、もう一つ問題があります。

nmapでは-Aスキャン(サービススキャン)をすることで、-sVよりも細かい情報を取ることができます。これやるとSSHではkey fingerprintが表示されるのですが、これがホンモノの2245/tcpだけ違う、という事態が発生します。バレちゃいます、どうしましょう。そこまで欺く必要も無いし、それは諦める……というのもまぁ一つの手ではありますが。



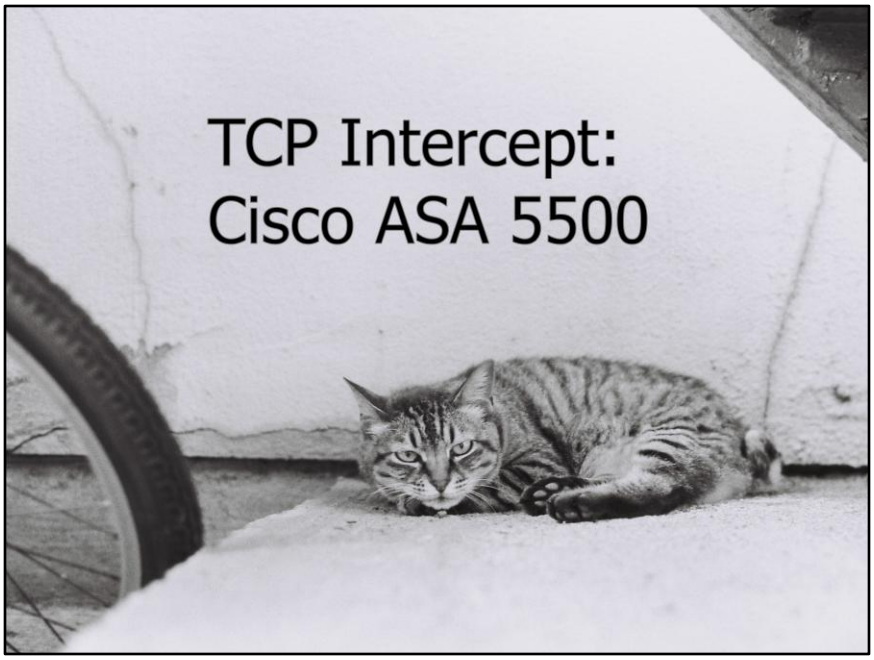
もはや本末転倒な気がしてきましたが、kippoでは任意の鍵ファイルを食わせることができます。
そのためここに挙げたように、/etc/ssh/ssh_host_dsa_keyなどをkippoのdataディレクトリ配下にコピーしてくれれば、key fingerprintまで同じにすることができます。

まぁ、逆にセキュリティ的にどうなよ感が強すぎるので、ここまでやることはおすすめしません。

確かに区別が付かなくなった

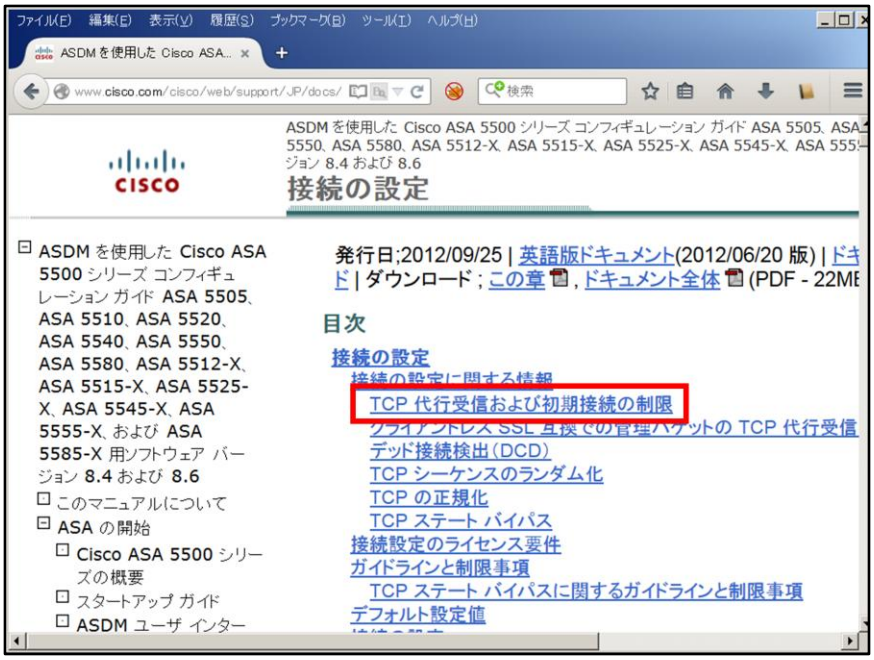
```
# nmap -n -A -p0-65535 192.168.2.66
.....(省略).....
PORT     STATE SERVICE VERSION
.....(省略).....
2222/tcp open  ssh     OpenSSH 5.3 (protocol 2.0)
| ssh-hostkey:
|   1024 bc:92:50:82:82:bc:d0:ab:b8:a2:6f:34:bb:f7:fc
|_  2048 ea:63:6a:de:44:98:c3:c9:35:88:d7:e9:81:cc:f7
2245/tcp open  ssh     OpenSSH 5.3 (protocol 2.0)
| ssh-hostkey:
|   1024 bc:92:50:82:82:bc:d0:ab:b8:a2:6f:34:bb:f7:fc
|_  2048 ea:63:6a:de:44:98:c3:c9:35:88:d7:e9:81:cc:f7
```

これは、ホンモノのsshdの鍵をkippoに移植したものです。
確かにkey fingerprintが一致して、もはや区別がつかなくなりました。でもおすすめしません。

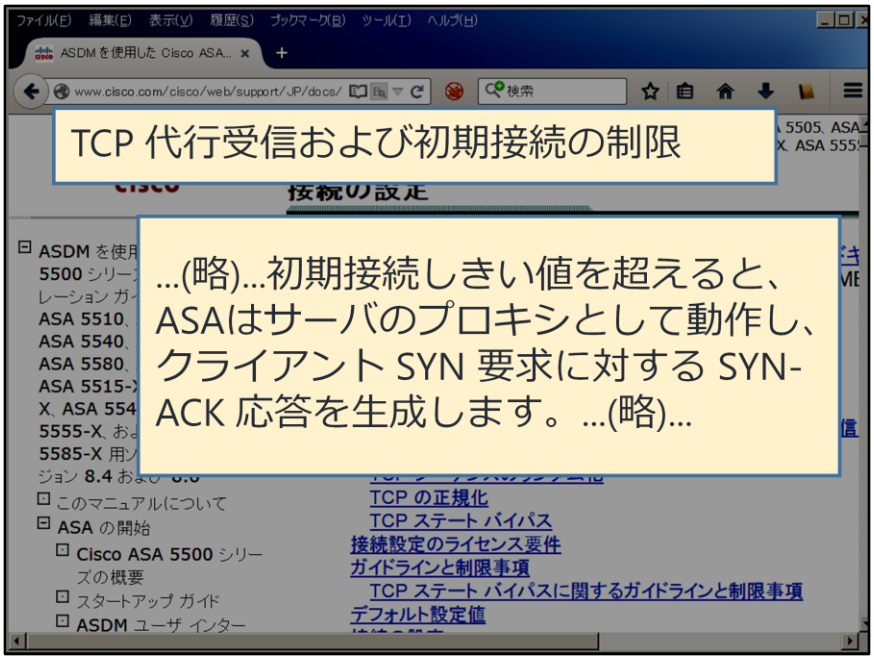


さて、分身の術はopenに見せかけるという手法でしたが、実は既にこのような防御処方はファイアウォールで一般的です。
Cisco ASAなどで設定できるやつですね。

あんまり特定ベンダの宣伝みたいになるのもアレですから、サラリとだけ解説します。



Ciscoでは、TCP代行受信という呼び方をしています（英語では、TCP Intercept）。



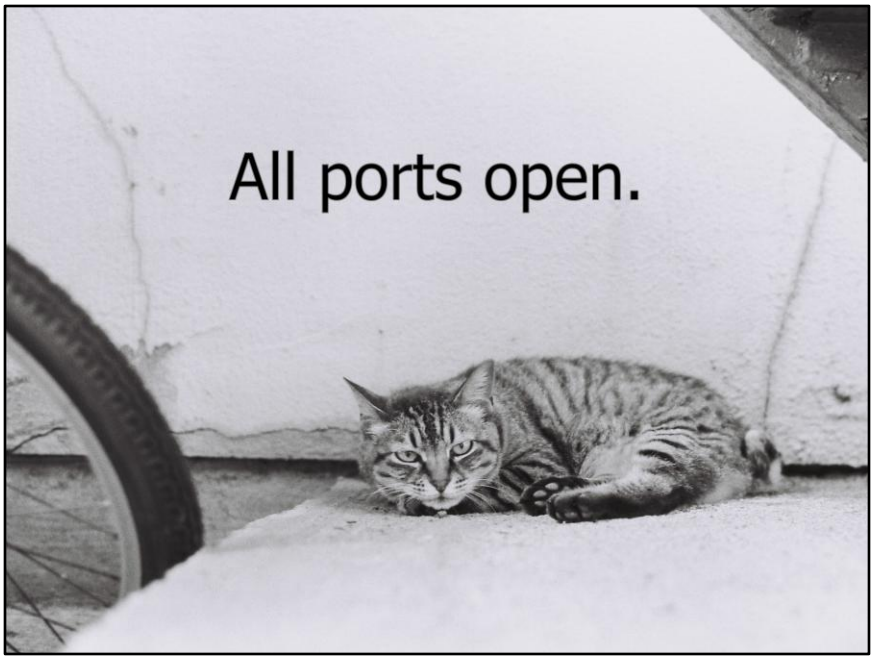
これは、ファイアウォールの裏にいるサーバに変わって、ACKを返してくれるという機能です。
つまり、SYNスキャンされると、サーバの前のCisco ASAがどんどんACKを勝手に返してくれるので、攻撃者から見ると全部のポートがOpenに見える、という挙動を示します。これは攻撃者にはなかなか厄介です。私も悩まされたもんです。

これ、要するにOpenポートを大量に見せてしまう、ということでssh分身の術と根本的な考え方は同じだと思います。

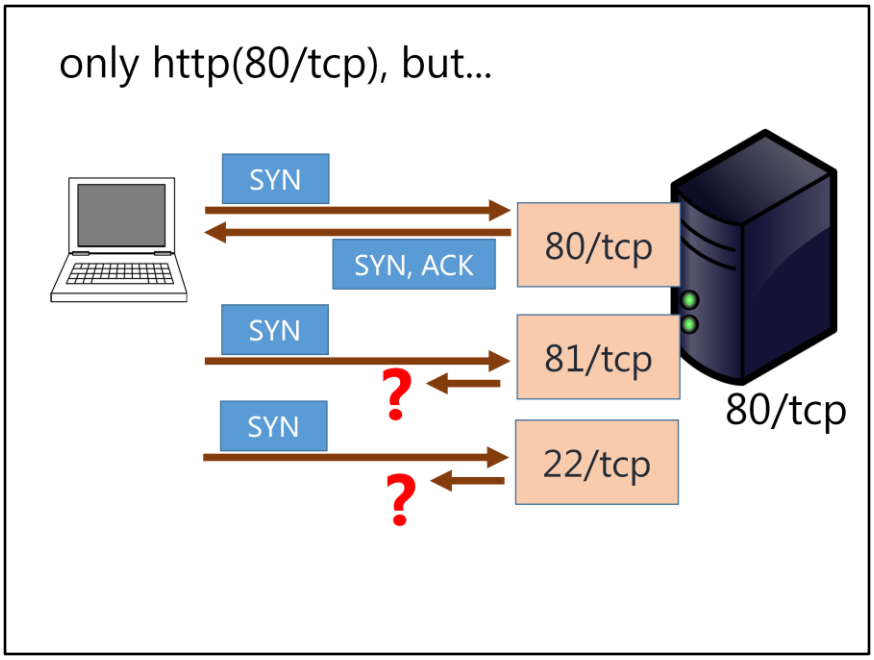
逆に全ポートfilteredに見せる

- iptablesのlimit, hashlimitモジュールを使うことで対策可能(説明省略)
- 参考) yasulib memo:
  フルポートスキャンから開放ポートを隠す方法
  - http://d.hatena.ne.jp/yasulib/20150302/1425282464

なお、ファイアウォール的なことはCiscoのクソ高いものを買わなくても、どのご家庭にもあるiptablesでも頑張ればできます。TCP代行受信とは逆に、ポートスキャンを検知してfilterするには、iptablesのlimit, hashlimitモジュールを使って書いてやればよい。
これはyasulibさんのブログ記事、「フルポートスキャンから開放ポートを隠す方法」が参考になります。
http://d.hatena.ne.jp/yasulib/20150302/1425282464

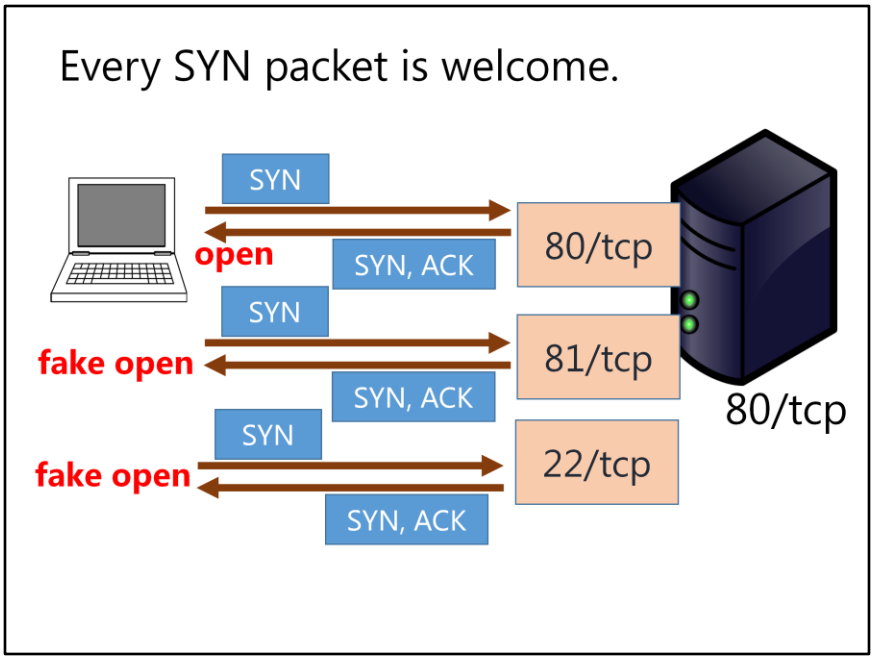


さて、最後にちょっとした主張です。
そもそもWebサービス等においては、TCPポートは全部Openに見せかけちゃえばいいのでは? という意見です。



少し考えを進めると、別にSYNスキャンに対する応答じゃ無くても。。。
例えば、80/tcpのみを外部に公開しているWebサーバがあるとします。一般の利用者は80/tcpにしか来ないはずですから、それ以外のポートに接続した際の状態がどうなっていようが、一般利用者には何も関係ないはずです。

この場合、80/tcpにくれば当然ACKを返しますが、じゃぁ22/tcpに来たら? 81/tcpに来たら? 普通ならRSTを返しますが。。。



このような時、全ポートのSYNに対してACKを返し、openであるかのように振る舞っても別に問題ないわけです。
繰り返しますが、利用者は80/tcpにしかアクセスしませんから、それ以外のポートが偽応答(本当はcloseしているのにopenしているかのように振る舞う)をいくら返しても、それで困るのはポートスキャンする人だけです。

このように、「SYNスキャンを受けたとき」とかいちいち限定せず、なんでもかんでも常に全部openのように振る舞っちまえ、というのは個人的に面白いなと思っているアイディアです。

で、最後に宣伝です。

6月に、友人と本を出しました。

三宅英明、大角祐介「新しいLinuxの教科書」、出版はSBクリエイティブです。
http://www.amazon.co.jp/dp/4797380942/

この勉強会に来ているような方には簡単すぎる内容かもしれませんが、本を買っていただけると、この会の運用資金にも充てることができます！
ぜひ買ってください!! 買ったあと不要なら、後輩にあげるなどしましょう。ブックオフには売らないで!