

Covert Channel

～偽装通信とその見破り方へのアプローチ～

宮本 久仁男 (a.k.a wakatono)

wakatono@todo.gr.jp

湯けむり偽装通信

ポロリもあるよ

この資料のライセンス

- この作品は、クリエイティブ・コモンズの帰属 - 非営利 - 同一条件許諾ライセンスの下でライセンスされています。この使用許諾条件を見るには、
<http://creativecommons.org/licenses/by-nc-sa/2.1/jp/>をチェックするか、クリエイティブ・コモンズに郵便にてお問い合わせください。住所は: 559 Nathan Abbott Way, Stanford, California 94305, USA です。

本日の趣旨

- Covert Channelを開設する人の心理を交えつつ
- どういうタイプのものがある
- どうやったらある程度アタリがつけられるか
- そんなことを解説できるといいなあ...
 - 定義はいろいろあるだろうけど、実用的な理解をどうするか？というあたりを主眼にしてるつもりです
- 決して、**悪用しないで**くださいませorz

Agenda

- Covert Channelの定義ってなんだろ？
- なぜ通信路を隠したがるか
- と・こ・ろ・で...暗号化通信にはどんなのがあるの？
- なぜ通信内容を隠したがるか
- いろいろなツール～通信を隠すためには？
- あらゆる提案～どれがホントかな？
- 見破り方～決定版はないが、アタリはつけられる？
- 行き過ぎると～ウィルスやワームと同じ手法
- そこまでやるならば～やっぱ構成管理かな
- むすび～不毛な争いにしかならない...orz

Covert Channel の定義って？

- Covert Channel の定義@NCSC

<http://www.radium.ncsc.mil/tpep/library/rainbow/NCSC-TG-030.html#2.0>

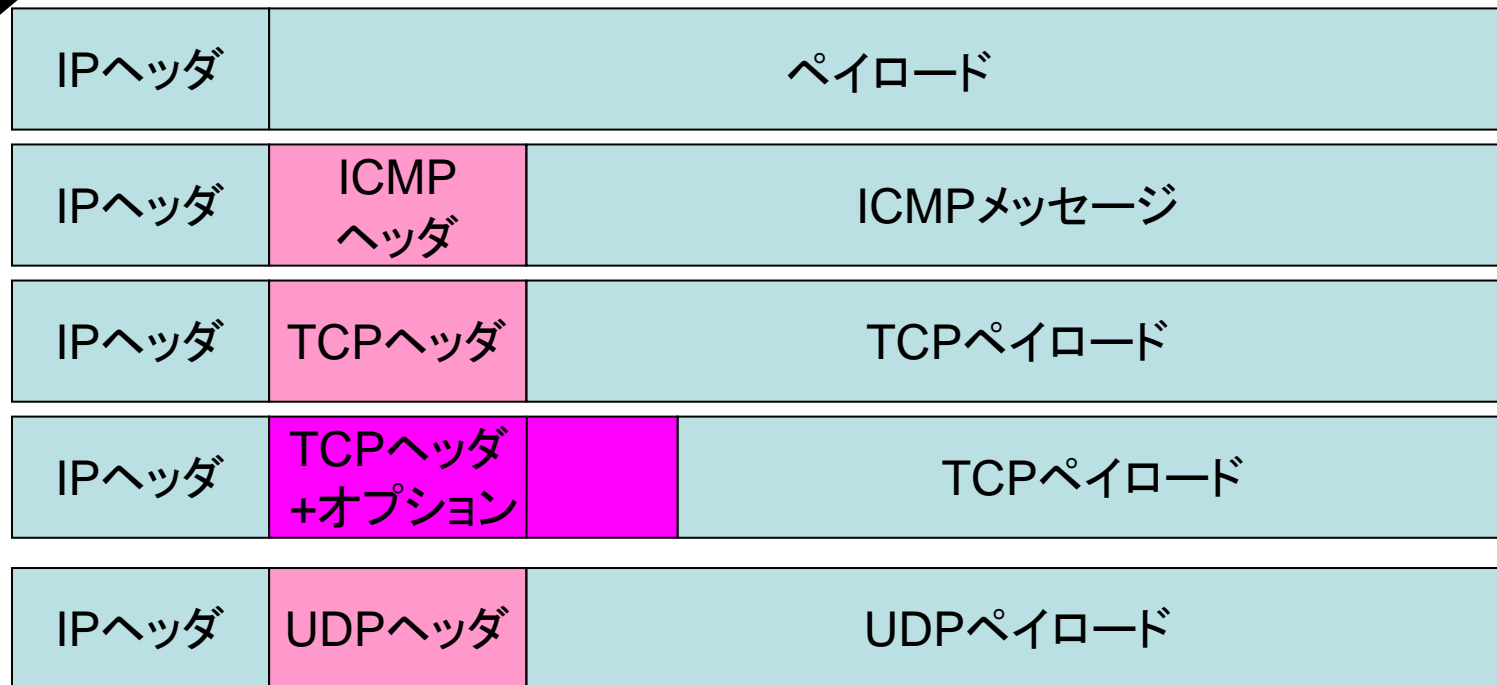
1. 情報を転送しているように見えない
2. リソースのステータスが入るべき値に任意の値を入れて送っている
3. リソースの配置ポリシーと、管理実装の結果によって、偽装されている／されていないと決められる
4. 情報の転送にあたり、通常はデータオブジェクトとして見えない（扱われない）部分を用いる

- ...どれがいい？
ぶっちゃけわからなきゃOKなのかなあ...
- キーワードは「偽装」

偽装のアプローチ

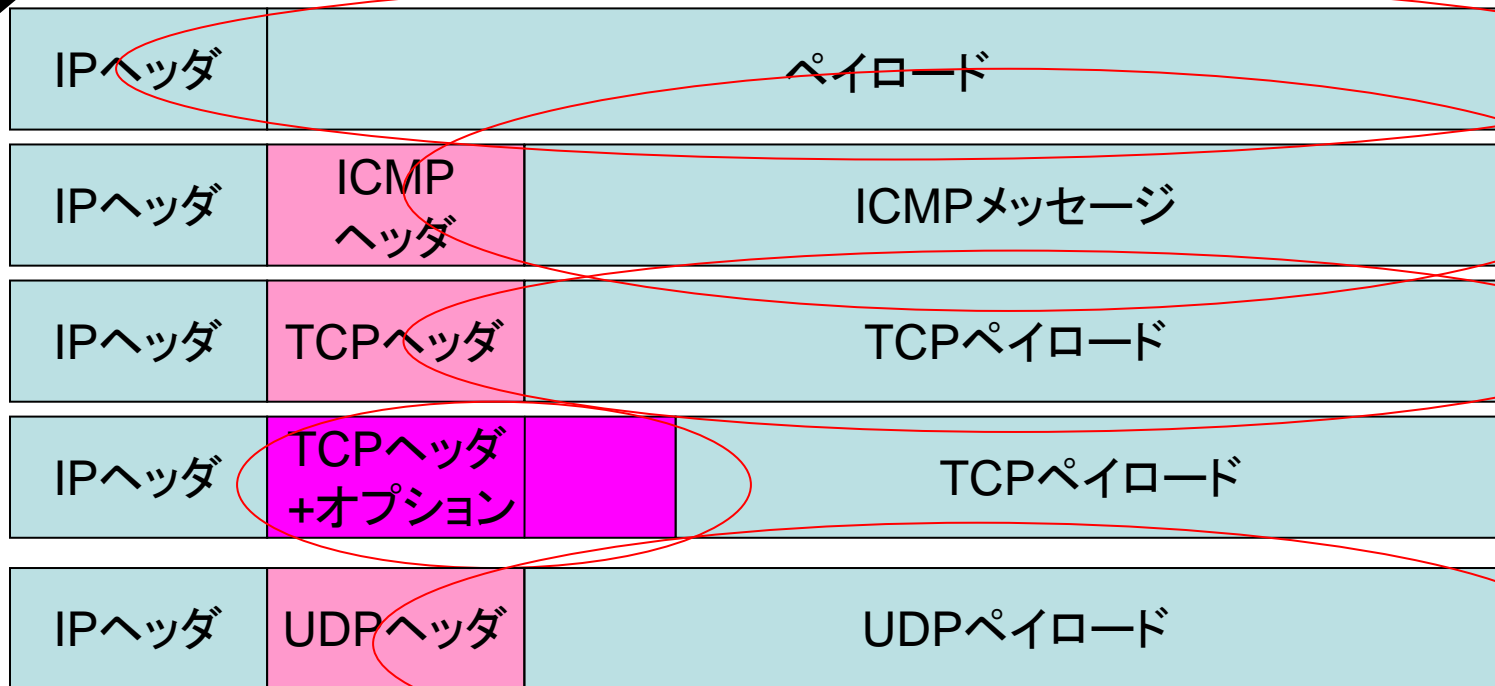
- 本来意図しないところにデータを隠す
 - 今回の解説のメイン
- ピアが通信するタイミングを符号化する
 - 今回は解説しませんが...
 - あるタイミングであるパターンを持ったパケットが流れる→1 / 流れない→0、とか...
 - HTの脆弱性の論拠

TCP/IPで使うパケットの構成



- あなたなら、どこにデータを入れる？

Covert Channelで使われる領域



- あなたなら、どこにデータを入れる？

なぜ通信を隠したがるの？

- そりゃあなた...
- 通信してることがバレたらとめられるからに決まってるじゃないですか
- それ以前に「普通には外部と通信できない」

いろいろなツール～通信を隠すには？

- TCPペイロードに隠す
 - GNU httptunnel
 - HTTP Tunnel@JUMPERZ.NET
 - mproxy
- ICMPペイロードに隠す
 - ICMPShell
 - PingTunnel
- TCPヘッダに隠す
 - covert_tcp.c
- UDPペイロードに隠す
 - nxts

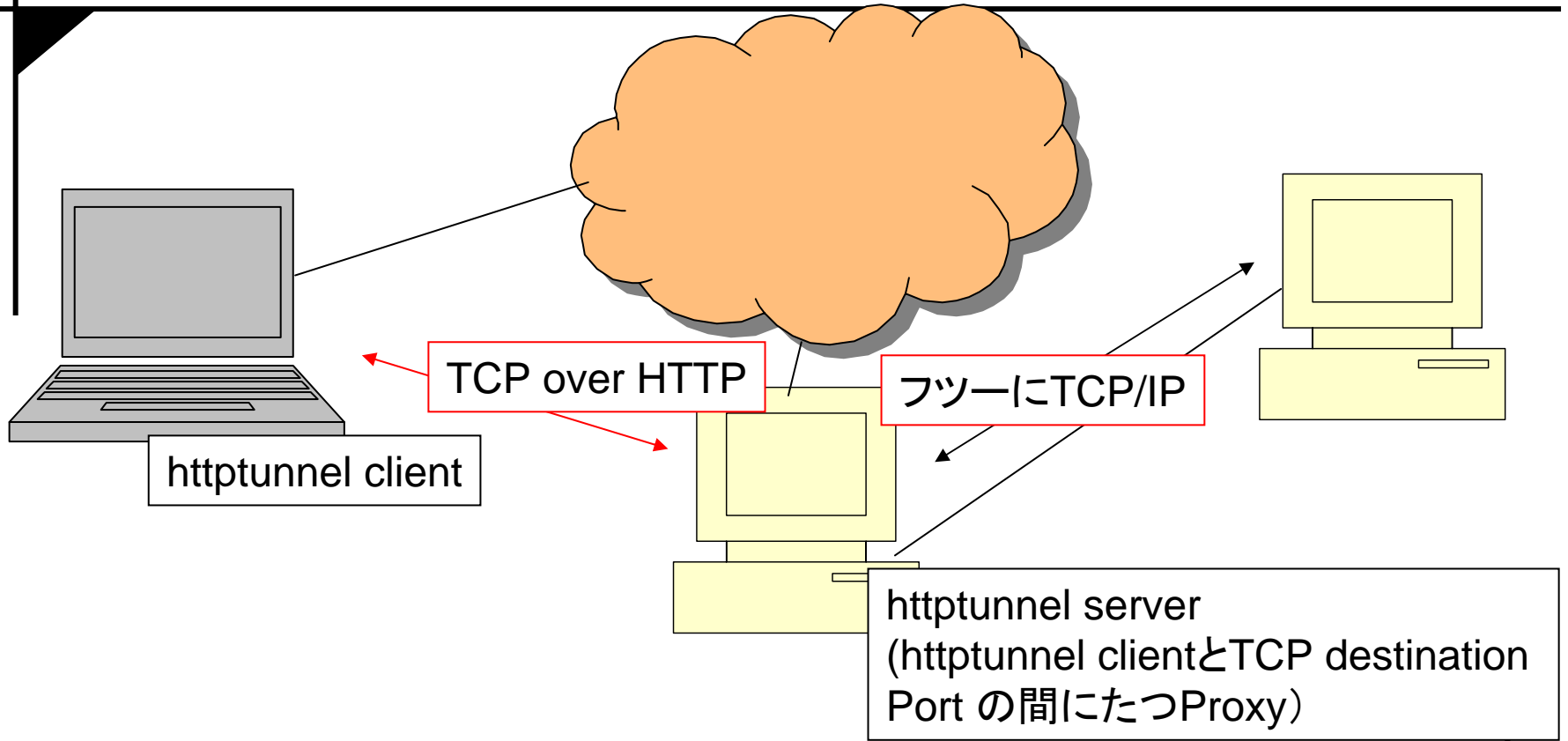
TCPペイロードに隠す(1/2)

- GNU httptunnel
 - Cで書かれたトンネルプログラム
 - HTTPトラフィックに、別の通信内容に乗せる
 - hts(サーバ)とhtc(クライアント)がHTTPプロトコルで通信
 - 1セッションしか張れない
- HTTP Tunnel@JUMPERZ.NET
 - Javaで書かれたトンネルプログラム
 - HTTPトラフィックに、別の通信内容に乗せる
 - REYE(サーバ)とLEYE(クライアント)がHTTPプロトコルで通信
 - 複数セッション可能

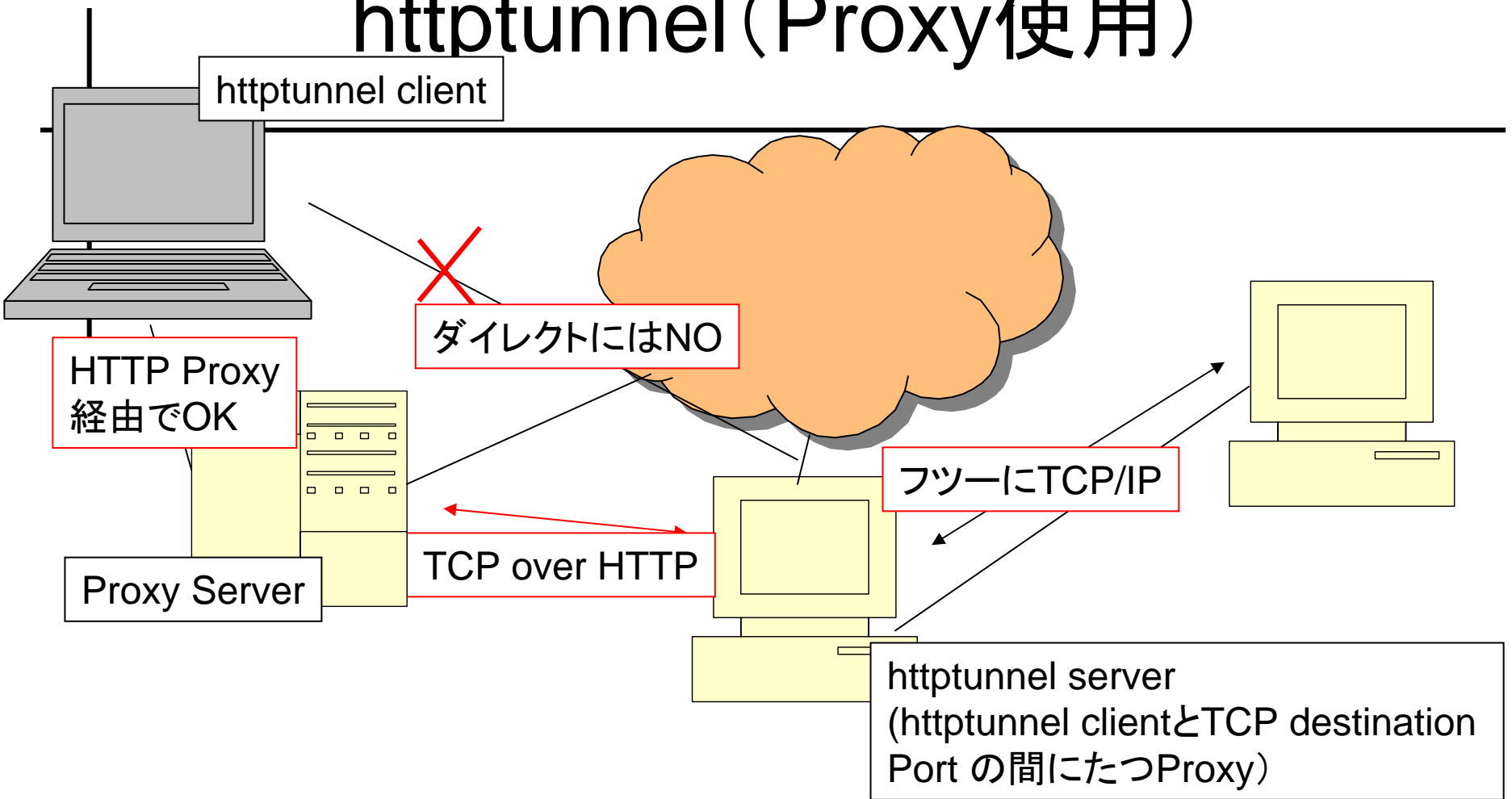
TCPペイロードに隠す(2/2)

- HTTP・Tunnel
 - 商用HTTPトンネル
- mproxy
 - SMTP(?)トンネル
 - SMTPというか、メールでペイロードをやりとり
 - 激重(らしい)

httptunnelの例



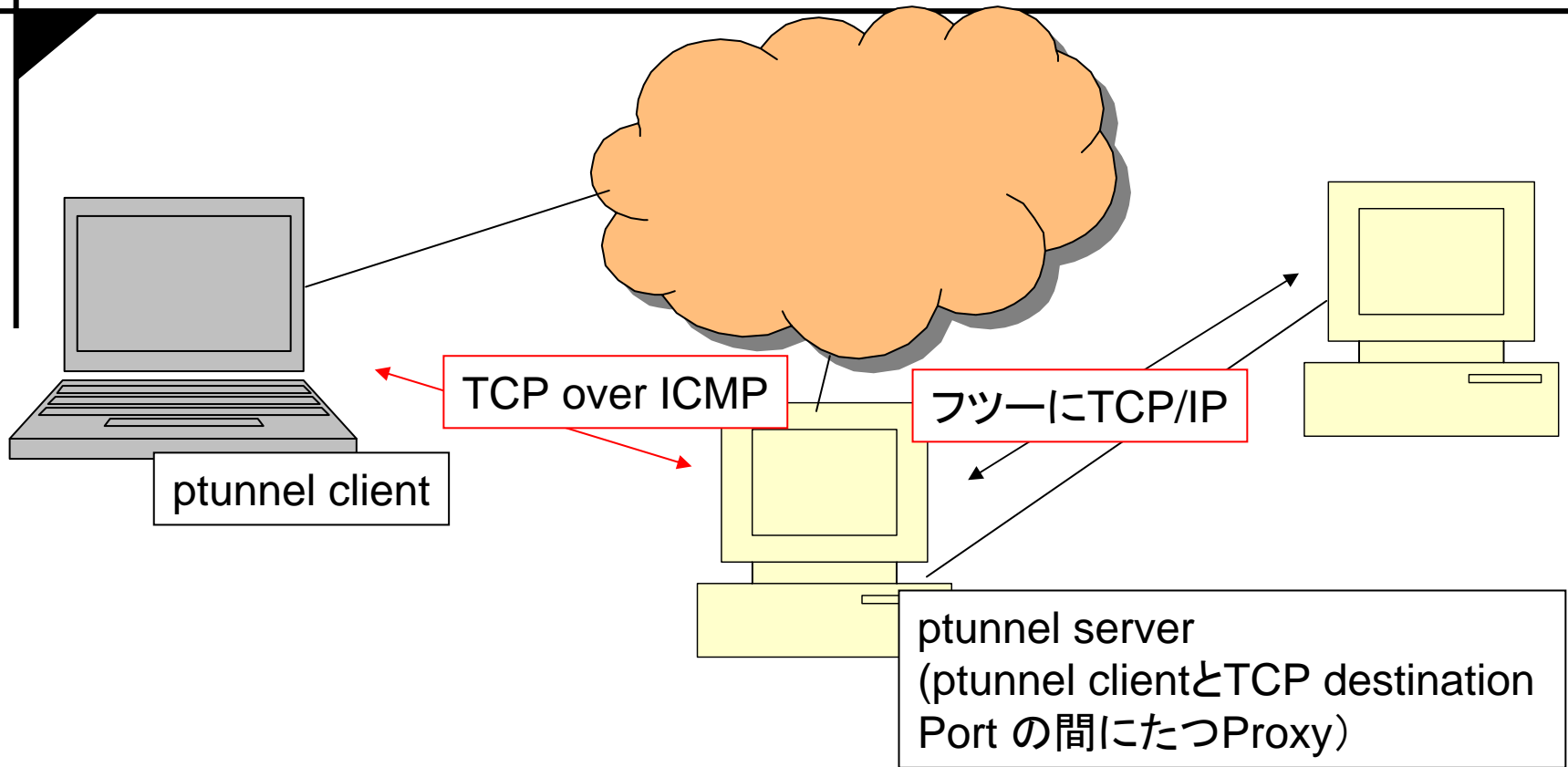
httptunnel (Proxy使用)



ICMPペイロードに隠す

- TunnelShell
 - ICMPパケットを使って、シェルインタラクションを実施
 - トンネルというかシェル
- PingTunnel
 - ICMPパケットを使って、トンネルを実現
 - 体感上、レスポンスは高速

PingTunnelの例



TCPヘッダに隠す

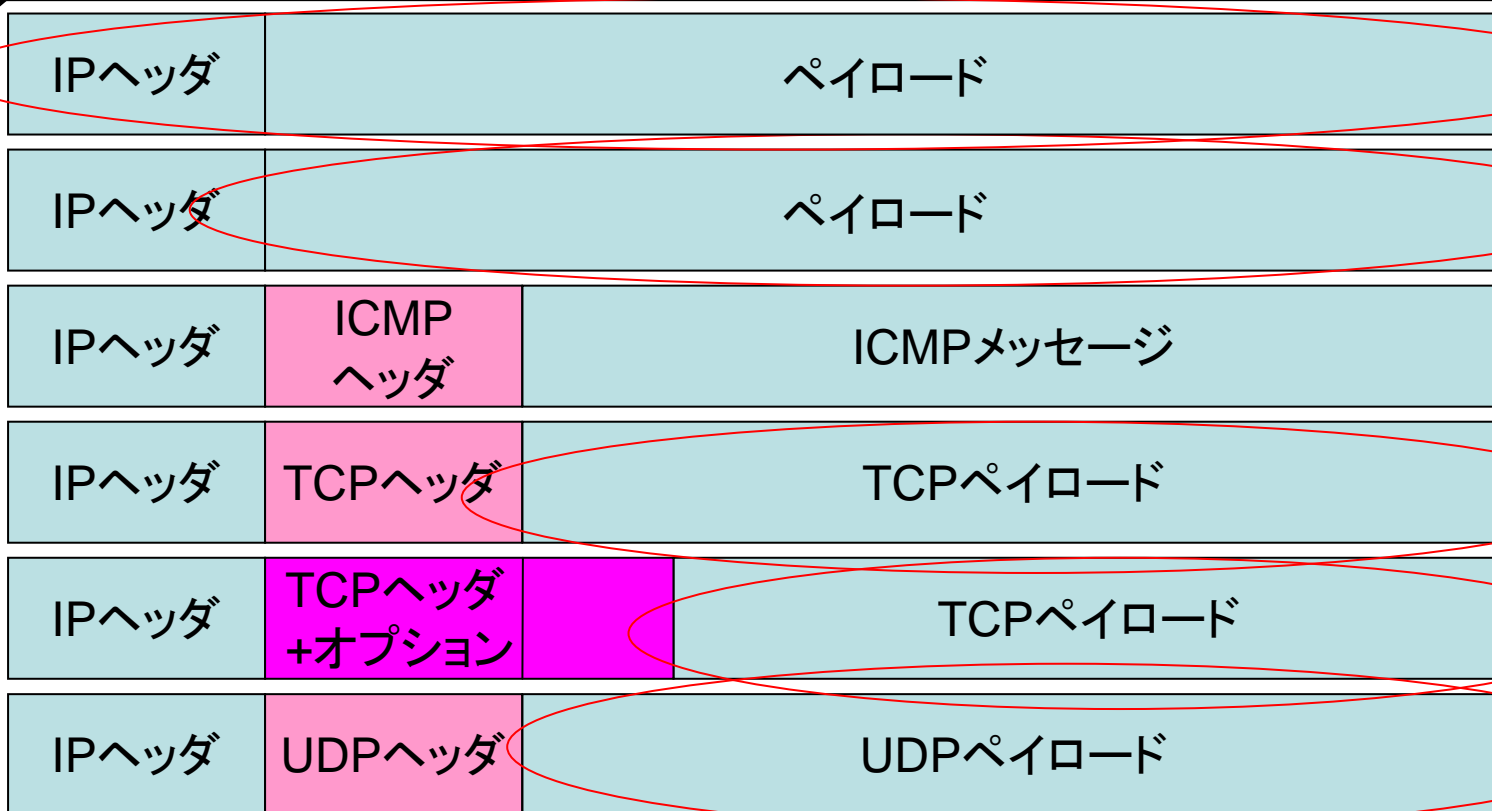
- covert_tcp
 - TCPヘッダの中(例:シーケンス番号)にデータを隠す
 - ペイロードを使わずにデータをやりとりするサンプル
 - かなり古い(1996年とか書いてあった)
 - Linux カーネル2.0でのみ動作とか書いてある
 - 実際は2.4や2.6でも動作する

と・こ・ろ・で...

暗号化通信にはどんなのがあるの？

- IPsec
- SSL
- SSH
- Stone
- Zebedee
- VPE(Virtual Private Ethernet)
- SoftEther

暗号化通信で暗号化される部分



- 基本的には、ペイロードもしくはパケット全体が暗号化対象。
- ヘッダのみとかそういうことはしない

暗号化通信方式／実装(1/2)

- IPsec
 - IPペイロードを暗号化
 - 暗号化／復号化のための情報は、ピアが持つ
 - 認証は別途IKEで実施
- SSH,SSL
 - 認証から暗号化通信までのしくみが含まれる
- Stone
 - 暗号化パケットリピータ

暗号化通信方式／実装(2/2)

- Zebedee
 - セキュアIPトンネル
 - (SSH – シェル) = Zebedee くらいのイメージ
- VPE(Virtual Private Ethernet)
 - L2レベルのトンネルプログラム
- SoftEther
 - いわずと知れたL2 over L4なプログラム

なぜ通信内容を隠したがるの？

- そりゃ、機密内容だからですよ
- 内容が漏洩したら、それは情報漏えいってやつ？
- 恥ずかしい内容だからに決まってるじゃないですか
- 三者三様ではあるものの、内容は見られたら(いろんな意味で)切腹モノ

どんな用途に使われる？

～Encrypted と Covert の違い～

- Encrypted
 - 通信内容を知られたくない
 - 通信しているという事実は知られてもOK

- Covert
 - 有意な通信をしていることを知られたくない
 - 普通に発生しうる通信と誤認させたい
 - 実際は相当怪しい特徴を備えている

PingTunnelの例

- サーバ側
 - ptunnel
- クライアント側
 - ptunnel -p (サーバ) -da (到達先ホスト) -dp (到達先ポート) -lp (ローカルの待ち受けポート)
- こんな感じ
 - サーバ: ptunnel
 - クライアント: ptunnel -p end.example.com -da server.example.com -dp 22 -lp 22222

PingTunnelだけだと心もとない

- じゃ、暗号化も
- お手軽に単一ポートの通信で暗号化が使えて...
- とりあえずSSHを組み合わせしてみよう
- どうなる？

SSH over PingTunnel

- 暗号化はSSHで
 - 隠蔽はPingTunnelで
 - ...最悪です(w
-
- つうか、隠蔽が破られても、通信内容は簡単にはわかられないようにしておこう
→そういうのを作る人

検知側の観点

- われわれには4つの観点
 - 1つ:フルパケットキャプチャ
 - 1つ:パケット解析
 - 1つ:フィルタリング
 - 1つ:解析結果をもとにした事情聴取

あらゆる提案～どれがホントかな？

- 監視のために有効な手段～
- 外に出るあらゆるパケットのダンプを取る
 - PacketBlackHoleなどを使用
 - Tcpdumpなどでもいいが、取りこぼし発生の可能性
 - チューニング必要
 - 後追いで調査する場合に有効
 - L3レベルの対外トラフィックに関連するパケットを全部取っておけば、(手間はかかるが)
 - 漏洩や不正な通信を阻止できるわけではない
- 外に出るパケットを制限する
 - 事前に制限かけるのに有効
 - 完全に制限がかけられるわけではない

見破り方~決定版はないが、アタリはつけられる？

- 通常のトラフィックに関する情報を持つ
 - 自分で監視する範囲のトラフィックを収集
 - 想定している範囲の通信で発生する実トラフィックのデータを収集
 - せめてICMPのトラフィックの癖くらい知っとけ
- 異常なトラフィックの特徴を把握する
 - 自分で使ってみて、特徴を収集するなど
 - 暗号化で隠しても「サイズが極端に変わるわけではない」
 - 暗号化はサイズを変えない(Paddingはするかも)
- それ以前に、「任意のパケット」が「外部に直接到達」するようにさせない
 - 特にヘッダなどが(一部でも)そのまま到達するようなケース
 - P2P通信(あんまりオレは詳しくないです)などで、ペイロードを外部に投げることが可能なケースもデンジャラス？

例: ICMPとPingTunnel(1)

- 普通のICMP
トラフィック
 - 要はping
 - 間隔とサイズ
がだいたいそ
ろってる

```
04:57:39.398448 IP mail2.todo.gr.jp > kid.todo.gr.jp: icmp 64: echo request seq 0
0x0000: 4500 0054 c37b 0000 4001 75fa 3dc5 e2d3 E..T.{..@.u.=...
0x0010: 3dc5 e2d5 0800 df19 14b7 0000 5c7d 9742 =.....¥}.B
0x0020: 256c 0000 0809 0a0b 0c0d 0e0f 1011 1213 %l.....
0x0030: 1415 1617 1819 1a1b 1c1d 1e1f 2021 2223 .....!"#
0x0040: 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233 $%&'()*+,-./0123
0x0050: 3435 3637                               4567

04:57:39.398540 IP kid.todo.gr.jp > mail2.todo.gr.jp: icmp 64: echo reply seq 0
0x0000: 4500 0054 26f3 0000 4001 1283 3dc5 e2d5 E..T&...@...=...
0x0010: 3dc5 e2d3 0000 e719 14b7 0000 5c7d 9742 =.....¥}.B
0x0020: 256c 0000 0809 0a0b 0c0d 0e0f 1011 1213 %l.....
0x0030: 1415 1617 1819 1a1b 1c1d 1e1f 2021 2223 .....!"#
0x0040: 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233 $%&'()*+,-./0123
0x0050: 3435 3637                               4567

04:57:40.405204 IP mail2.todo.gr.jp > kid.todo.gr.jp: icmp 64: echo request seq 256
0x0000: 4500 0054 c382 0000 4001 75f3 3dc5 e2d3 E..T....@.u.=...
0x0010: 3dc5 e2d5 0800 dbfe 14b7 0100 5d7d 9742 =.....]}.B
0x0020: 2687 0000 0809 0a0b 0c0d 0e0f 1011 1213 &.....
0x0030: 1415 1617 1819 1a1b 1c1d 1e1f 2021 2223 .....!"#
0x0040: 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233 $%&'()*+,-./0123
0x0050: 3435 3637                               4567

04:57:40.405252 IP kid.todo.gr.jp > mail2.todo.gr.jp: icmp 64: echo reply seq 256
0x0000: 4500 0054 26f4 0000 4001 1282 3dc5 e2d5 E..T&...@...=...
0x0010: 3dc5 e2d3 0000 e3fe 14b7 0100 5d7d 9742 =.....]}.B
0x0020: 2687 0000 0809 0a0b 0c0d 0e0f 1011 1213 &.....
0x0030: 1415 1617 1819 1a1b 1c1d 1e1f 2021 2223 .....!"#
0x0040: 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233 $%&'()*+,-./0123
0x0050: 3435 3637                               4567
```

例:ICMPとPingTunnel

- PingTunnel
のトラフィック
– 少なくともサ
イズが不審

```
05:02:39.308499 IP kid.todo.gr.jp > ns2.todo.gr.jp: icmp 68: echo reply seq 3
0x0000: 4500 0058 2d7e 0000 4001 0bf1 3dc5 e2d5 E..X~..@!...=...
0x0010: 3dc5 e2d6 0000 57db 5789 0003 d520 0880 =.....W.W.....
0x0020: 0000 0000 0000 0000 4000 0002 0000 0003 .....@.....
0x0030: 0000 0020 0003 5789 fffa 2000 3936 3030 .....W.....9600
0x0040: 2c39 3630 30ff f0ff fa27 00ff f0ff fa18 ,9600....'.....
0x0050: 0078 7465 726d fff0 .....xterm..

05:02:39.312171 IP kid.todo.gr.jp > ns2.todo.gr.jp: icmp 54: echo reply seq 4
0x0000: 4500 004a 038c 4000 4001 f5f0 3dc5 e2d5 E..J..@.!@!...=...
0x0010: 3dc5 e2d6 0000 cef5 5789 0004 d520 0880 =.....W.....
0x0020: 0000 0000 0000 0000 8000 0002 0000 0003 .....
0x0030: 0000 0012 0004 5789 fffb 03ff fd01 fffd .....W.....
0x0040: 22ff fd1f fffb 05ff fd21 .....!

05:02:39.312832 IP ns2.todo.gr.jp > kid.todo.gr.jp: icmp 108: echo request seq 4
0x0000: 4500 0080 0018 4000 4001 f92e 3dc5 e2d6 E.....@.!@!...=...
0x0010: 3dc5 e2d5 0800 7327 5789 0004 d520 0880 =.....s'W.....
0x0020: 0000 0000 0000 0000 4000 0002 0000 0004 .....@.....
0x0030: 0000 0048 0004 5789 fffd 03ff fc01 fffb ...H..W.....
0x0040: 22ff fa22 0301 0000 0362 0304 020f 0500 ".....b.....
0x0050: 0007 621c 0802 0409 421a 0a02 7f0b 0215 ..b.....B.....
0x0060: 0f02 1110 0213 1100 0012 0000 fff0 fffb .....
0x0070: 1fff fa1f 005a 0023 fff0 fffd 05ff fb21 .....Z.#.....!

05:02:39.312870 IP kid.todo.gr.jp > ns2.todo.gr.jp: icmp 108: echo reply seq 4
0x0000: 4500 0080 2d7f 0000 4001 0bc8 3dc5 e2d5 E...-..@!...=...
0x0010: 3dc5 e2d6 0000 7b27 5789 0004 d520 0880 =.....{W.....
0x0020: 0000 0000 0000 0000 4000 0002 0000 0004 .....@.....
0x0030: 0000 0048 0004 5789 fffd 03ff fc01 fffb ...H..W.....
0x0040: 22ff fa22 0301 0000 0362 0304 020f 0500 ".....b.....
0x0050: 0007 621c 0802 0409 421a 0a02 7f0b 0215 ..b.....B.....
0x0060: 0f02 1110 0213 1100 0012 0000 fff0 fffb .....
0x0070: 1fff fa1f 005a 0023 fff0 fffd 05ff fb21 .....Z.#.....!
```

例: HTTPとhttptunnel(1/2)

- 小さい
リクエ
ストに
対して
大きい
レスポ
ンス
(非対
称)

```
05:32:05.136452 mail2.todo.gr.jp.4946 > slashdot.jp.http: . ack 1  
win 16968 (DF)
```

```
4500 0028 6e41 4000 7f06 5f11 3dc5 e2d3  
3dd7 d00d 1352 0050 ef62 c0b5 31f4 fdcc  
5010 4248 4b93 0000
```

```
05:32:05.137704 mail2.todo.gr.jp.4946 > slashdot.jp.http: P  
1:569(568) ack 1 win 16968 (DF)
```

```
4500 0260 6e42 4000 7f06 5cd8 3dc5 e2d3  
3dd7 d00d 1352 0050 ef62 c0b5 31f4 fdcc  
5018 4248 0473 0000 4745 5420 2f20 4854  
5450 2f31 2e31 0d0a 4163 6365 7074 3a20  
696d 6167 652f 6769 662c 2069 6d61 6765  
2f78 2d78 6269 746d 6170 2c20 696d 6167  
652f 6a70 6567 2c20 696d 6167 652f 706a  
7065 672c 2061 7070 6c69 6361 7469 6f6e  
2f78 2d73 686f 636b 7761 7665 2d66 6c61  
7368 2c20 6170 706c 6963 6174 696f 6e2f  
766e 642e 6d73 2d65 7863 656c 2c20 6170  
706c 6963 6174 696f 6e2f 766e 642e 6d73  
2d70 6f77 6572 706f 696e 742c 2061 7070  
6c69 6361 7469 6f6e 2f6d 7377 6f72 642c  
(まだまだつづく)
```

例: HTTPとhttptunnel(2/2)

- httptunnel
 - 不思議に等間隔に同じサイズのトラフィック...
 - トンネルして
る処理に依存

```
05:14:07.803093 IP 192.168.255.131.32834 > 192.168.255.130.8888: P
695372427:695372428(1)
ack 709943961 win 5840 <nop,nop,timestamp 230064 226072>
 0x0000: 4500 0035 c6a5 4000 4006 f3c5 c0a8 ff83 E..5..@.@.....
 0x0010: c0a8 ff82 8042 22b8 2972 8a8b 2a50 e299 .....B".)r..*P..
 0x0020: 8018 16d0 40db 0000 0101 080a 0003 82b0 ....@.....
 0x0030: 0003 7318 45 .....s.E
05:14:07.803134 IP 192.168.255.130.8888 > 192.168.255.131.32834: . ack 1 win 5792
<nop,nop
,timestamp 226569 230064>
 0x0000: 4500 0034 de96 4000 4006 dbd5 c0a8 ff82 E..4..@.@.....
 0x0010: c0a8 ff83 22b8 8042 2a50 e299 2972 8a8c .....B*P.)r..
 0x0020: 8010 16a0 8422 0000 0101 080a 0003 7509 .....u.
 0x0030: 0003 82b0 .....
05:14:07.803716 IP 192.168.255.130.8888 > 192.168.255.131.32835: P
712143080:712143081(1)
ack 699000545 win 5792 <nop,nop,timestamp 226569 229562>
 0x0000: 4500 0035 582c 4000 4006 623f c0a8 ff82 E..5X,@.@.b?....
 0x0010: c0a8 ff83 22b8 8043 2a72 70e8 29a9 e6e1 .....C*rp.)...
 0x0020: 8018 16a0 5611 0000 0101 080a 0003 7509 ....V.....u.
 0x0030: 0003 80ba 45 .....E
05:14:07.807474 IP 192.168.255.131.32835 > 192.168.255.130.8888: . ack 1 win 9648
<nop,nop
,timestamp 230064 226569>
 0x0000: 4500 0034 ac4e 4000 4006 0e1e c0a8 ff83 E..4.N@.@.....
 0x0010: c0a8 ff82 8043 22b8 29a9 e6e1 2a72 70e9 .....C".)....*rp.
 0x0020: 8010 25b0 8a13 0000 0101 080a 0003 82b0 ..%.....
 0x0030: 0003 7509 .....u.
05:14:12.809636 IP 192.168.255.130.8888 > 192.168.255.131.32835: P 1:2(1) ack 1 win 5792 <
nop,nop,timestamp 227070 230064>
 0x0000: 4500 0035 582d 4000 4006 623e c0a8 ff82 E..5X-@.@.b>....
 0x0010: c0a8 ff83 22b8 8043 2a72 70e9 29a9 e6e1 .....C*rp.)...
 0x0020: 8018 16a0 5225 0000 0101 080a 0003 76fe ....R%.....v.
 0x0030: 0003 82b0 45 .....E
```

トンネル経路の絞り方

1. 想定されるCovert Channelの開設経路を調査
 - TCPペイロード経由
 - UDPペイロード経由
 - ICMP経由
2. それぞれの経路をつぶしていく
 - 今回挙げたツールに関して次に

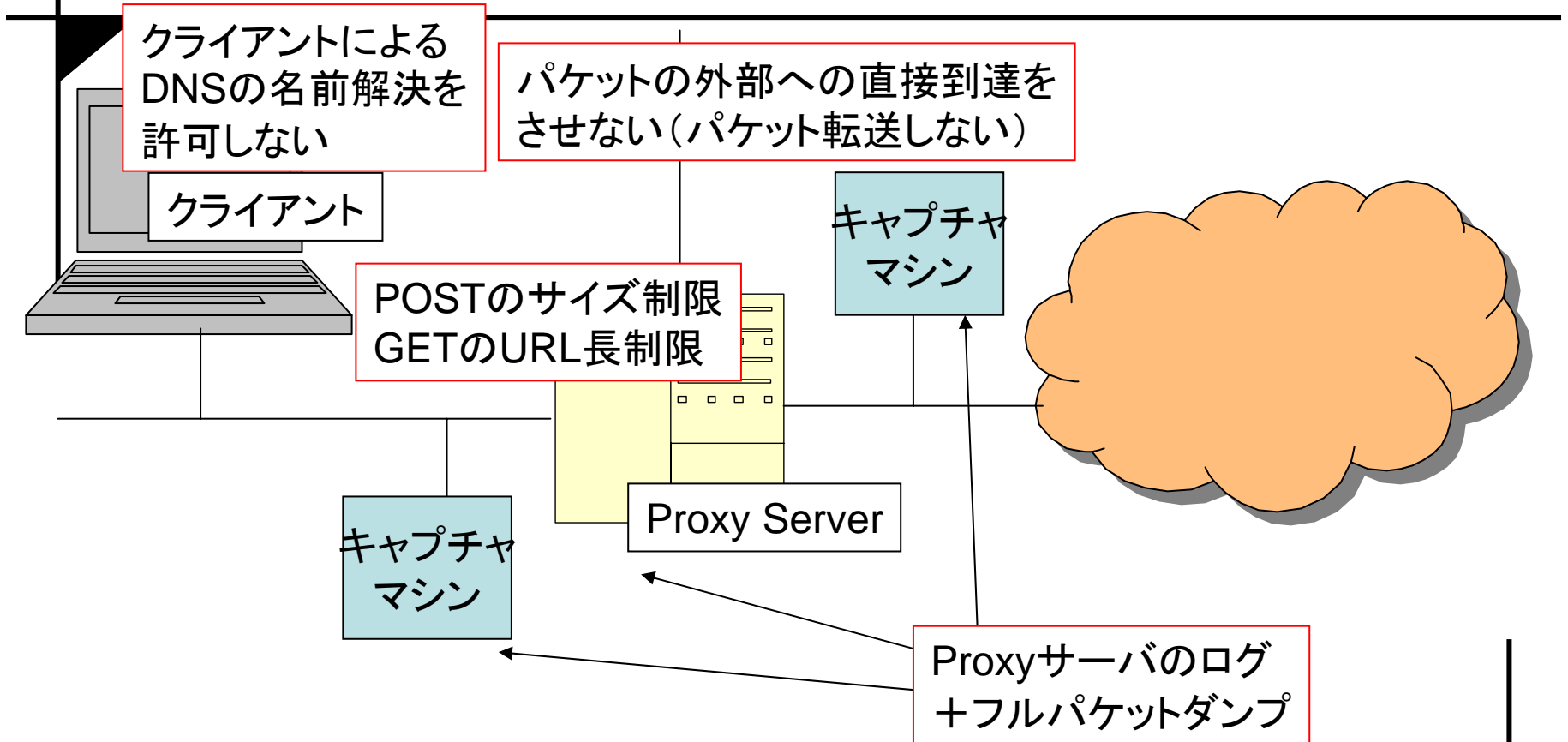
隠蔽経路のつづし方(1)

- nstx
 - 直接名前解決をさせない
 - NSレコード関連のクエリが飛んでいかれるとアウト
 - HTTP Proxyなどにすべて任せる(Proxyを活用)
- Mproxy
 - 不審なメールボックスを作らない／作らせない
- PingTunnel
 - <http://www.cs.uit.no/~daniels/PingTunnel/>
- TunnelShell
 - ダイレクトに外につながせなければOK
- ICMP Tunnel
 - ダイレクトに外につながせなければOK
- Covert_TCP
 - ダイレクトに外につながせなければOK

隠蔽経路のつぶし方(2)

- パケット／通信内容がそのまま外に出て行かないように
 - Proxyを使うことが有効
 - POSTメソッドなど任意のデータを送信するメソッドを制限
 - 利便性とのトレードオフ
 - 繰り返しPOSTをさせない
 - GETのパラメータで送られることもあるため、長さは制限
 - 回避されるけどなorz
 - 大半が絞れる
- HTTPS経由で出て行くものはどうする？
 - 推測しかなさそう
 - CONNECTメソッドをあぶりだす
 - メソッドの発行回数や相手を特定
 - 特定した相手の素性を調査
 - 規模が小さいうちはまだあぶりだせる
 - 規模が大きくなると、監査そのものが大変
 - 出る前に通信相手を制限する(Sygateなどのアプローチ)

制限ネットワークの構成例



設定例

- Squidの設定例
 - POST禁止のACLを記述
 - request_body_max_sizeを小さく設定
 - デフォルト10MB
 - その他通信先も制限
 - 随時追加できるようにする
- Linuxのパケット転送禁止
 - echo 0 > /proc/sys/net/ip_forward

行き過ぎると～ウィルスやワームと同じ手法

- 自前の通信エンジンを持つようになるかも
 - OSのIPスタックに依存しないように
 - OSの通信制御機能では制限は面倒なケースも
- そのうち自前のIPスタックを持つように...
 - 実は意外に小さい>IPスタックの実装
 - 決めうちの実装ならばなおさら
 - TCPスタックもCovert_TCPの例を見ると小さくできる
- Covert Channel over Covert Channel
 - 複数の偽装通信手法／ツールの組み合わせorz
- コバンザメのように、他の通信に寄生の可能性
 - 隠し場所はごろごろしている
- いろいろ妄想広がります...

そこまでやるならば～やっぱ構成管理かな

- Windowsベースであれば可能
- 不審なモジュールの実行を制限
- 不審なモジュールの通信を制限
- これってポリシーで制限できんかな？

...ノードからの通信をすべて監視が現実解？

分散監視エージェントのようなものが置かれれば
そのの方がよさそう

→構成管理ミドルウェアのようなアプローチ

むすび～不毛な争いにしかならない...orz

- すべての通信が管理できるわけではない
 - 理論的には可能だが、現実的にムリ筋
- 正常とされる通信以外はすべてはじけば...
 - あらゆる逆手に取られる可能性
 - 正常とされる通信に乗せたCovert Channelもある
- 通信内容の混在が敵
 - HTTP/SMTP/POP3の3つがCovert Channelに混在するだけでも大変
 - そもそも怪しい通信が発生している段階で疑えという気も
 - トラフィック量は...
HTTP,POP3:To クライアント>To サーバ
SMTP,クライアント向け <> サーバ向け
 - これが混在するだけでもなにやってるかアタリつかなくなる

おまけ:オレオレ* (w(その1)(1/3))

バージョン	IPヘッダ長	サービスタイプ	IPパケット長	
識別子			フラグ	フラグメントオフセット
生存可能時間 (残存ホップ数)	プロトコル番号		チェックサム	
発信元アドレス				
到達先アドレス				
(オプション)				

- オレオレプロトコル番号
 - プロトコル番号は、上位にペイロードを渡す時の手がかりとなるが...
 - 相互に仕切りができてれば、オレオレプロトコル番号ってのもありかな？
 - プロトコル番号をテキトーにしておこう

おまけ:オレオレ*(w(その1)(2/3)

- IP over IP:プロトコル番号4
- TCP:プロトコル番号6
- UDP:プロトコル番号17
- GRE:プロトコル番号47
- ESP:プロトコル番号50
- AH:プロトコル番号51
- IPIP:プロトコル番号94
- L2TP:プロトコル番号115
- 未割り当て:138~252

おまけ:オレオレ* (w(その1)(3/3))

- プロトコル番号を変えてしまうと、それを知っているOS／アプリケーションしか通信できなくなる
- あまり役に立つとも思えないが、そんな方法もあるよ、ということで
- NAPTはTCP/UDPのポート番号がアドレス変換の手がかりなので、使えなくなる可能性大

もひとつおまけ: Covert_TCPの活用法

- Covert_TCP: TCPヘッダにデータを隠すやり方のコンセプトコード
- そのままだと単にデータを送るだけ
- いろんな隠し方がある
- Covert_TCPといいつつも、実はRaw IPソケットをいじくってる
- 改造して遊ぼう

改造例

- Covert_TCPだけど、Covert_UDPにしてしまう
 - 知ったことか！データが届けばいいんだという人むけ
- Covert_TCP_**Opt**にしてしまう
 - TCPヘッダ長を変更
 - オプションをすき放題いじる
 - 取り出しもテキストに
 - 任意の長さのデータを送れる
 - TCPチェックサムは最初から0

おまけ: オレオレ* (w(その2))(1/2)

Src port		Dest port	
Seq			
Ack Seq			
hdl	reserved	Code bits	Window size
Check sum		Urgent Pointer	
Options(n*4 octed)			

- オレオレTCPヘッダ
- NAPTがあろうと知ったことか
- ポート番号とウィンドウサイズをはじめとして、ルータなどで変更される可能性があるものは使わない
 - それ以外って何がある？
 - シーケンス番号があるじゃん
 - 意図的にヘッダを組み替えて解析をちょびっと面倒に

おまけ:オレオレ*(w(その2)(2/2)

NAPT前のパケットダンプ

```
03:46:35.666492 ns2.todo.gr.jp.telnet > sarge.todo.gr.jp.32850: P 49:52(3) ack 74 win 5792 <nop,nop,timestamp 2080335456 125366753> (DF) [tos 0x10]
03:46:35.666681 sarge.todo.gr.jp.32850 > ns2.todo.gr.jp.telnet: P 74:77(3) ack 52 win 1460 <nop,nop,timestamp 125366753 2080335456> (DF) [tos 0x10]
03:46:35.667094 ns2.todo.gr.jp.telnet > sarge.todo.gr.jp.32850: P 52:89(37) ack 77 win 5792 <nop,nop,timestamp 2080335456 125366753> (DF) [tos 0x10]
```

NAPT後のパケットダンプ

```
03:46:35.665663 mail2.todo.gr.jp.32850 > ns2.todo.gr.jp.telnet: P 71:74(3) ack 49 win 1460 <nop,nop,timestamp 125366753 2080335455> (DF) [tos 0x10]
03:46:35.666456 ns2.todo.gr.jp.telnet > mail2.todo.gr.jp.32850: P 49:52(3) ack 74 win 5792 <nop,nop,timestamp 2080335456 125366753> (DF) [tos 0x10]
03:46:35.666720 mail2.todo.gr.jp.32850 > ns2.todo.gr.jp.telnet: P 74:77(3) ack 52 win 1460 <nop,nop,timestamp 125366753 2080335456> (DF) [tos 0x10]
```

- シーケンス番号フィールドは影響なし

参考資料(1/4)

- Covert Channels in the TCP-IP Protocol Suite
<http://www.ouah.org/rowlandcover.htm>
 - Covert_TCPのコードもこの中に
- TunnelShell, Tunneling Shell Access via TCP/UDP/Fragmented/ICMP/RawIP Packets
<http://www.securiteam.com/tools/5OP0O1P6AE.html>
- FRYXAR Home Page
<http://www.geocities.com/fryxar/>
 - TunnelShellはこちらで配布

参考資料(2/4)

- HTTP-Tunnel
<http://www.http-tunnel.com/>
- nstx
<http://nstx.dereference.de/nstx/>
- HTTP Tunnel@JUMPERZ.NET
<http://www.jumperz.net/index.php?i=2&a=0&b=0>
- GNU HTTP Tunnel
<http://www.nocrew.org/software/httpunnel.html>

参考資料(3/4)

- stone
<http://www.gcd.org/sengoku/stone/Welcome.ja.html>
- Zebedee
<http://www.winton.org.uk/zebedee/>
- GNU VPE (Virtual Private Ethernet)
<http://savannah.gnu.org/projects/gvpe>

参考資料(4/4)

- SoftEther
<http://www.softether.com/>
- OpenSSH
<http://www.openssh.org/>
- トンネルの掘り方/見つけ方
<https://www.7th-angel.net/seculog/media/1/20050329-OSC2005-Tunnel.pdf>
- PROTOCOL NUMBERS
<http://www.iana.org/assignments/protocol-numbers>

宣伝？～ぼくのパソコンを守って



著者
根津さん、園田さん、私

イラストレータ：
武礼堂さん

ISBN:4-7980-1032-4
秀和システムより出てたり