

Service Cloaking and Anonymous Access; Combining Tor with Single Packet Authorization (SPA)

Michael Rash
CTO, Solirix Inc.

michael.rash@solirix.com
<http://www.cipherdyne.org/>

DEF CON
08/05/2006

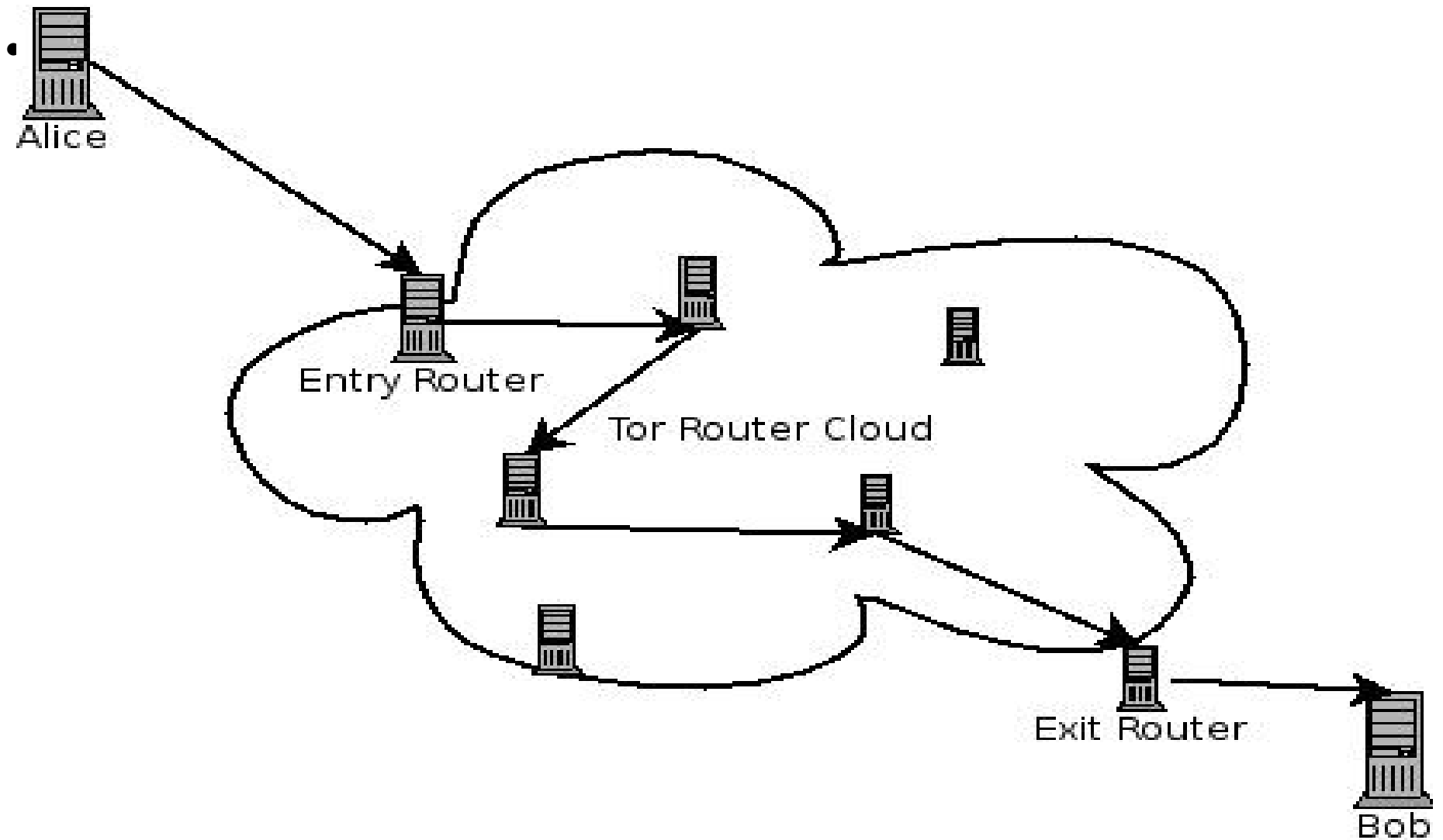
Agenda

- The Onion Router (Tor) and anonymous service access
- Default-drop packet filters and Single Packet Authorization
- SPA over Tor
- Live demonstration
- Future development

Tor

- Network of virtual circuits
- Packets take a random path through several servers (onion routers)
- No individual router knows the complete path through the router cloud
- Compatible with any application with SOCKS support
- Traffic is encrypted

Virtual Circuit



Single Packet Authorization

- Use default-drop packet filters to minimize code execution paths
- Authentication and authorization data is passively monitored via libpcap (or the ulogd pcap writer)
- No traditional “server” in the Berkeley sockets sense

Default-Drop

- A default-drop packet filter is the next best thing to Marcus Ranum's perfect firewall:



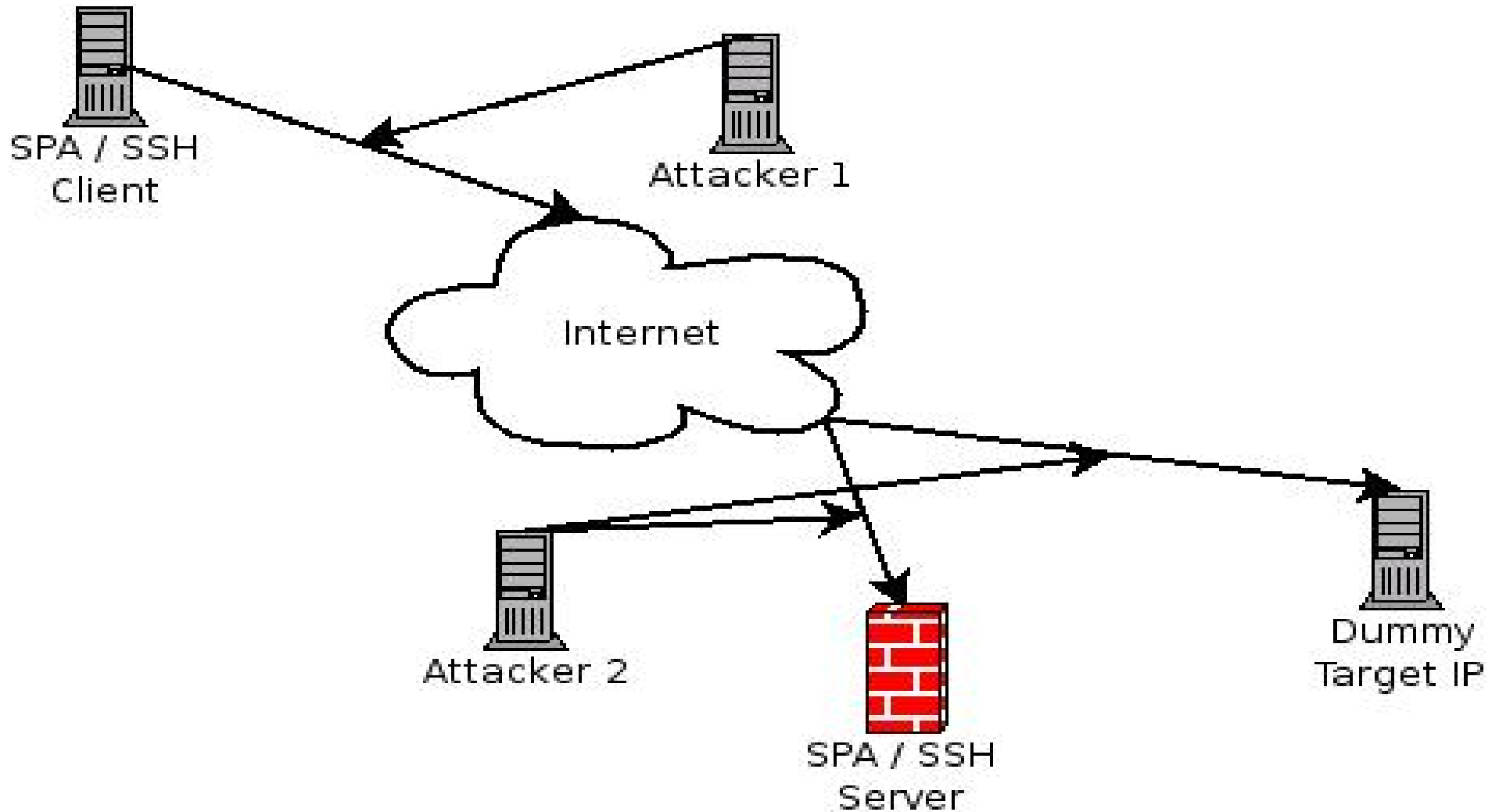
Single Packet Authorization (cont'd)

- Up to minimum MTU of data can be sent
- Large data size makes it possible to use 2048-bit GnuPG keys
- Replay attacks easily thwarted by MD5 calculation and storage on the server side
- Authorization packets can be spoofed (except over Tor)

SPA vs. Port Knocking

- Both use default-drop packet filters
- Both can timeout ACCEPT rules but use connection tracking to allow a TCP session to remain open
- SPA solves the replay problem
- SPA is compatible with asymmetric ciphers
- SPA cannot be broken by trivial sequence busting attacks
- SPA does not look like a port scan

Who can sniff what?



Security Through Obscurity?

- No more than passwords or GnuPG private keys
- SPA is additive, i.e. other security mechanisms already built into various protocols are still used

<http://bastille-linux.org/jay/obscurity-revisited.html>

SPA over Tor

- Why not just always run client SSH connections (or other services) over Tor?
 - Still need SPA and default-drop packet filter since an attacker can also run connections over Tor
- Sending the SPA packet over Tor adds another layer; traffic analysis (which Tor is designed to thwart) is made more difficult
- socat SOCKS 4a proxy

SPA over Tor (cont'd)

- Tor uses TCP for transport
- Cannot influence TCP stacks used to build virtual circuit (passive OS fingerprinting of these stacks still works)
- Port knocking is incompatible with Tor; must run SPA
 - Over socat proxy, Nmap -sS never sets up a virtual circuit. Nmap -sT sets up a circuit, but many TCP stacks get involved unless a real server is available

Tor and Bi-directional Communication

- Using TCP for transport implies bi-directional communication is required
- Technically, SPA model of single blind UDP packet does not fit the Tor transport requirement
- Cannot simply include SPA data within a TCP SYN packet
- Must have a real TCP server that is accessible on the server side

Tor and Bi-directional Communication (cont'd)

- In `ENABLE_TCP_SERVER` mode, `fwknop` spawns a minimal TCP server
 - Runs as “nobody” on port 62201 (configurable)
 - Does `bind()`, `listen()`
 - Loops over successive `accept()` and `recv()` calls with no other code
 - Session is FINished by server after first TCP data packet is sent by the client
 - Number of potential vulnerabilities in this server is less than the potential vulnerabilities in a more complex server (SSH)
 - Data is still acquired via `pcap` by `fwknopd` instead via the minimal server, so SPA packets to other ports continue to work

Making a Connection

- Tor is designed to make the exit router hard to predict
- Must send SSH connection over the open Internet

SYN Scan (over socat)

```
[tor-client]$ socat TCP4-LISTEN:62201,fork  
SOCKS4A:localhost:70.x.x.x:62201,socksport=9050
```

```
[tor-client]# nmap -sS -P0 -p 62201 127.0.0.1
```

```
Starting Nmap 4.01 ( http://www.insecure.org/nmap/ ) at 2006-07-09 19:21  
EDT
```

```
Interesting ports on localhost.localdomain (127.0.0.1):
```

```
PORT      STATE SERVICE
```

```
62201/tcp open  unknown
```

```
Nmap finished: 1 IP address (1 host up) scanned in 0.043 seconds
```

```
[ssh-server]# tcpdump -i eth0 -l -nn port 62201
```

(NOTHING HERE, a virtual circuit is never established)

Connect() Scan (over socat)

```
[tor-client]$ socat TCP4-LISTEN:62201,fork  
SOCKS4A:localhost:70.x.x.x:62201,socksport=9050
```

```
[tor-client]$ nmap -sT -P0 -p 62201 127.0.0.1
```

```
Starting Nmap 4.01 ( http://www.insecure.org/nmap/ ) at 2006-07-11 07:19  
EDT
```

```
Interesting ports on localhost.localdomain (127.0.0.1):
```

```
PORT      STATE SERVICE
```

```
62201/tcp open  unknown
```

```
Nmap finished: 1 IP address (1 host up) scanned in 0.007 seconds
```

Connect() Scan (over socat) cont'd

```
[ssh-server]# tcpdump -i eth0 -l -nn port 62201
```

```
19:23:03.236140 IP 64.74.207.50.20087 > 70.x.x.x.62201: S  
478646557:478646557(0) win 5840 <mss 1460,sackOK,timestamp  
288052901 0,nop,wscale 7>
```

```
19:23:09.316140 IP 82.224.104.98.4984 > 70.x.x.x.62201: S  
1512871859:1512871859(0) win 64240 <mss 1460,nop,nop,sackOK>
```

```
19:23:18.315758 IP 128.2.141.33.59959 > 70.x.x.x.62201: S  
1531387242:1531387242(0) win 65535 <mss  
1460,nop,nop,sackOK,nop,wscale 1,nop,nop,timestamp 79586290 0>
```

Operating Systems Running Tor

```
# fwknopd --os --fw-log /var/log/messages
```

```
[+] Entering OS fingerprinting mode.
```

```
[+] Parsing iptables log: /var/log/messages
```

```
[+] 80.190.242.130
```

```
      S4:64:1:60:M*,S,T,N,W2      Linux:2.5::Linux 2.5 (sometimes 2.4)
```

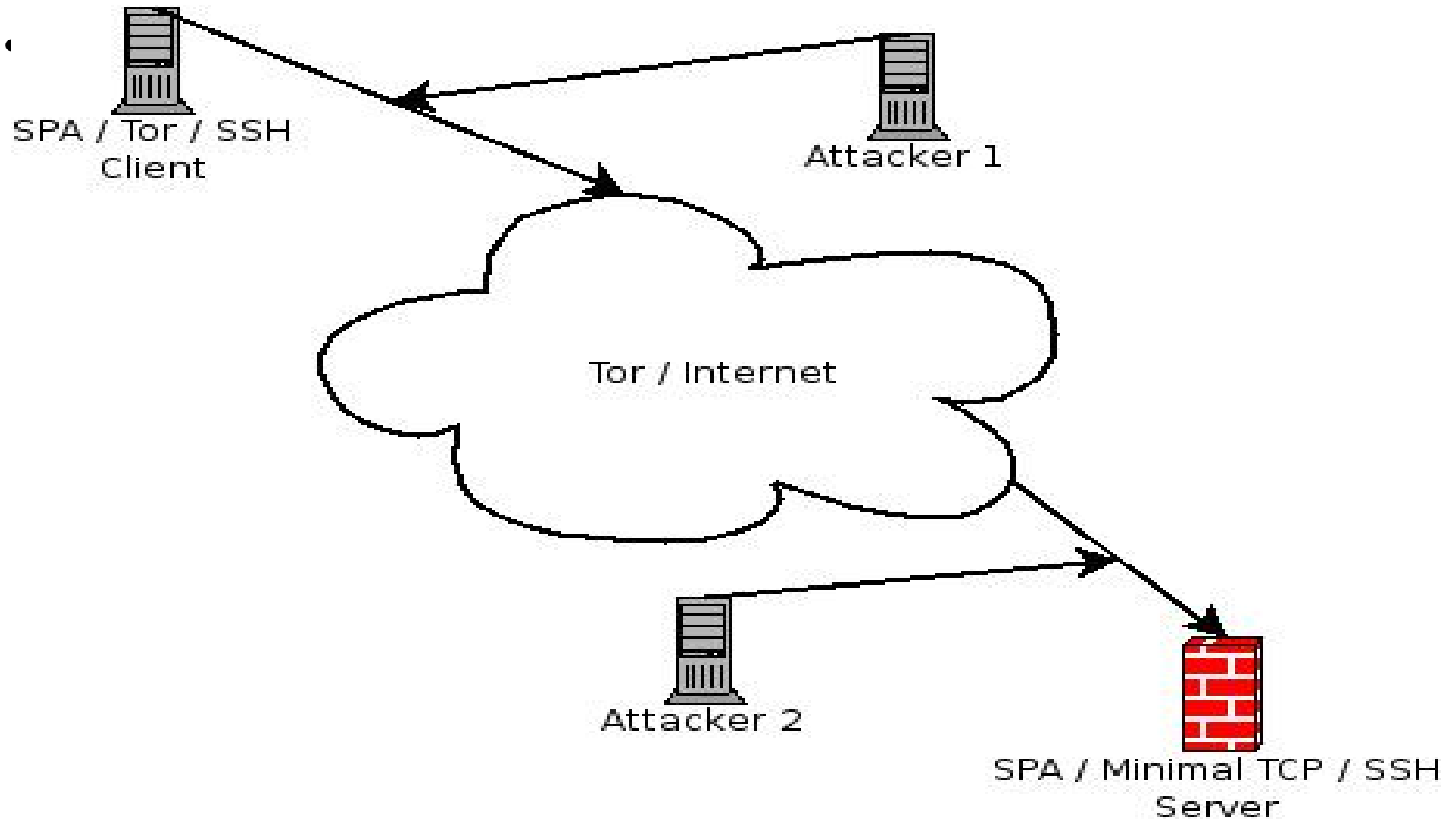
```
[+] 24.9.31.51
```

```
      32768:64:1:60:M*,N,W0,N,N,T   FreeBSD:5.0-5.1::FreeBSD 4.8-5.1 (or MacOS X)
```

```
[+] 210.17.245.14
```

```
      32768:64:1:44:M*              HP-UX:B.10.20::HP-UX B.10.20
```

Who can sniff what? (revisited)



Live Demonstration...

Questions?

michael.rash@solirix.com

<http://www.cipherdyne.org/>