

Open Source

Secure Wireless Networks

in a

Windows[™] World

Sean Comeau

An Open Source Solution
that's Free, Secure,
User Friendly
and with no
Commercial Lock-in

Secure Wireless Access Points (SWAP)

Why "**SWAP**"? ...because we need more acronyms.

Should Wireless Access Points be given the same due diligence as Internet Firewalls?

Does your Wireless Network have the following?

- **Strong Authentication** (IPsec, NOT WEP!)
- **Strong Encryption** (Dual Factor)
- **Firewall protection on the Access Points**
- **Intrusion Detection System integration**

Reality check!

How many Commercial Access Points do you know have the above?

Insecure Wired Corporate Networks

Are you better off without a wireless network?

Are you really more secure?

Even if you don't have a wireless network, if your work force uses laptops with embedded wireless support, then you better thing again.

At home, they connect to the Internet using their Wireless Broadband Access Points. At work, while plugged into the corporate network, these laptops will happily connect to Open Access Points with nwid's such as "default", "wireless", "Linksys", etc, as if they were at home.

It gets more interesting and scary when they are unknowingly bridging between the wireless and wired lan.

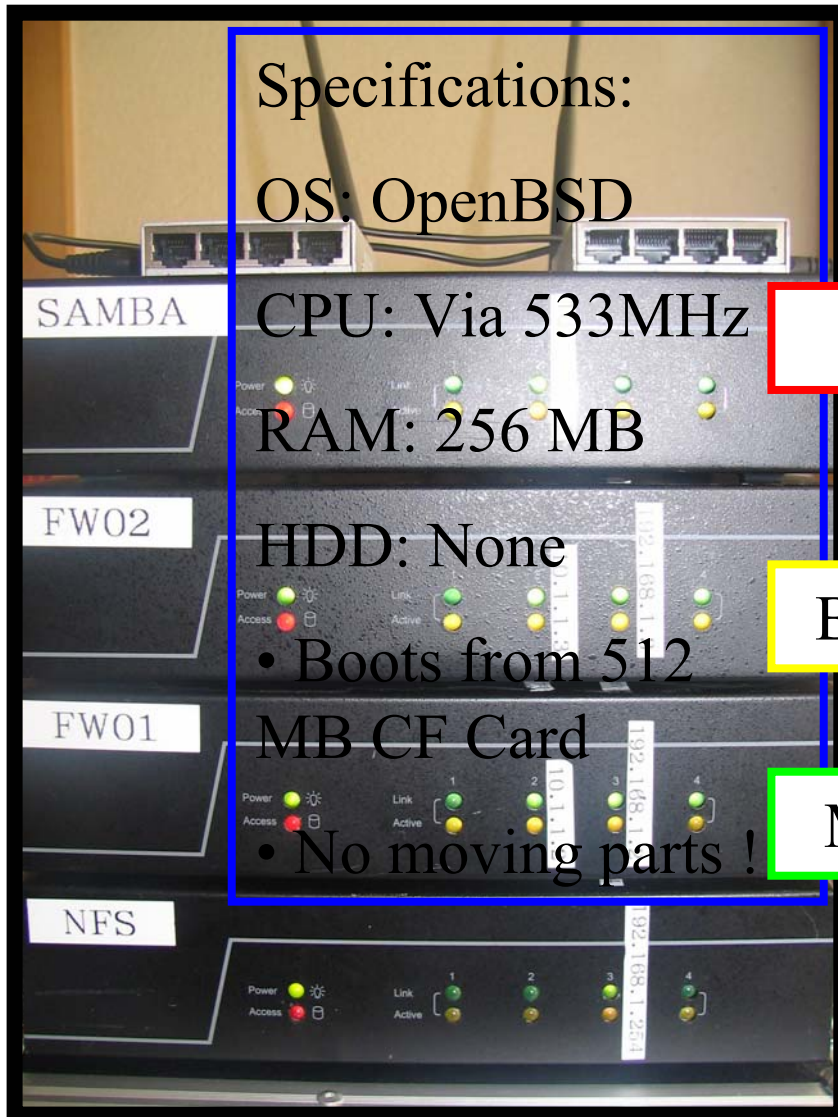
Requirements for a Secure Wireless Network

We wanted to have a solution with the following requirements:

- Strong authentication (Dual Factor)
- Strong encryption (IPsec with 3DES or better)
- Strong Firewall protection (A firewall with a track record)
- Easy to use and idiot proof (1 button, 1 passphrase)
- No interoperability issues with Windows clients
- Scalable and Robust (Fault Tolerant, Stateful Roaming)

We also wanted to ensure that the solution...

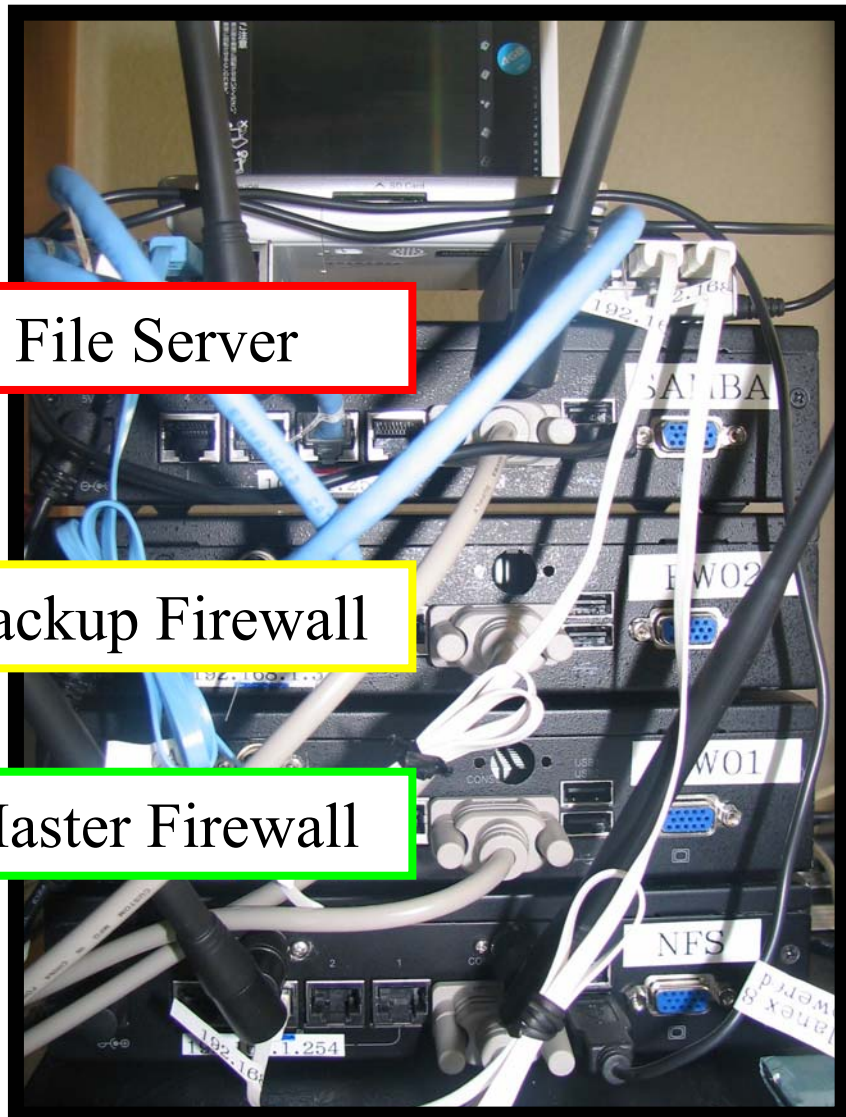
- Was not too onerous for Administrators to maintain
- Did not rely on Proprietary Hardware or Software



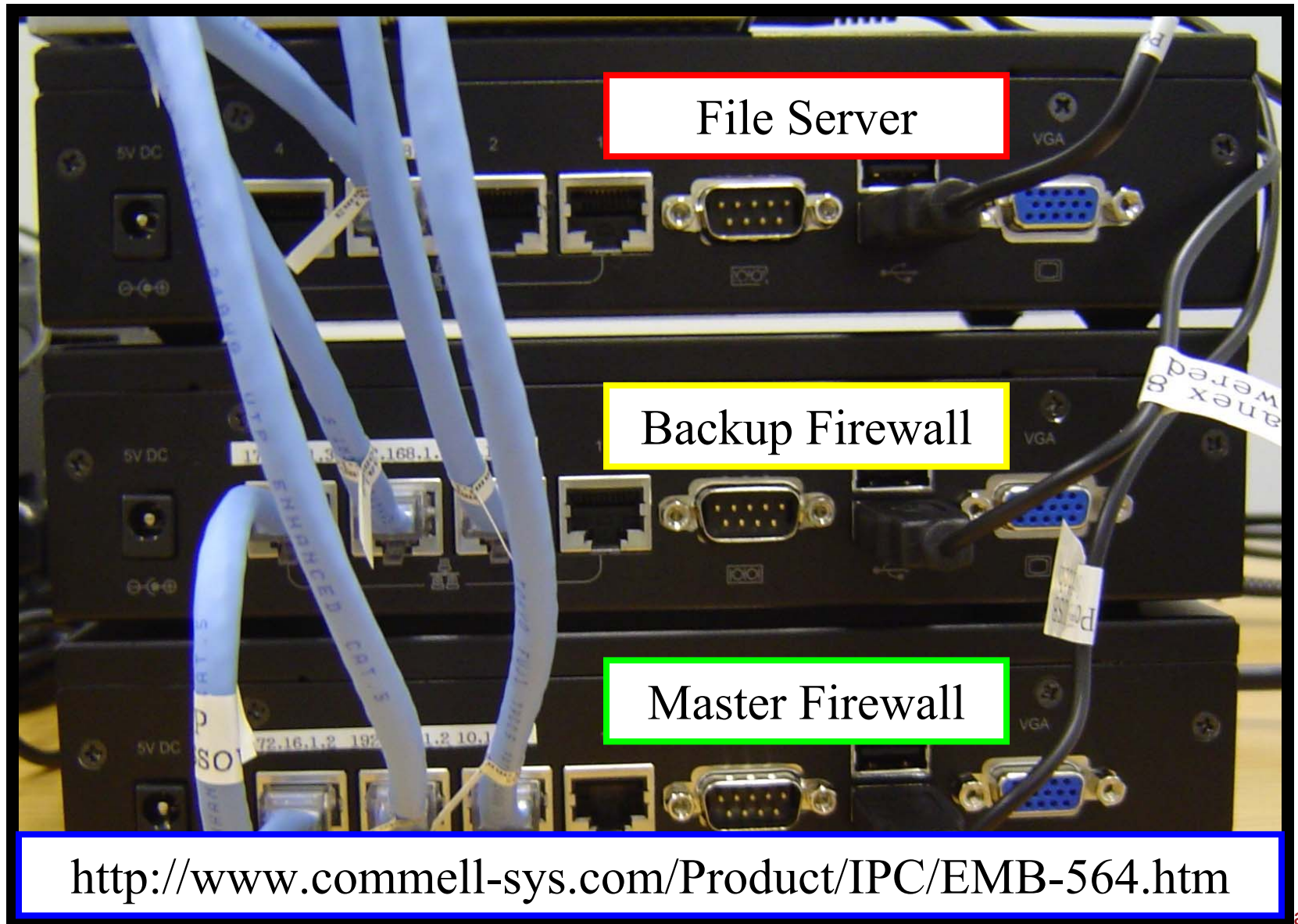
File Server

Backup Firewall

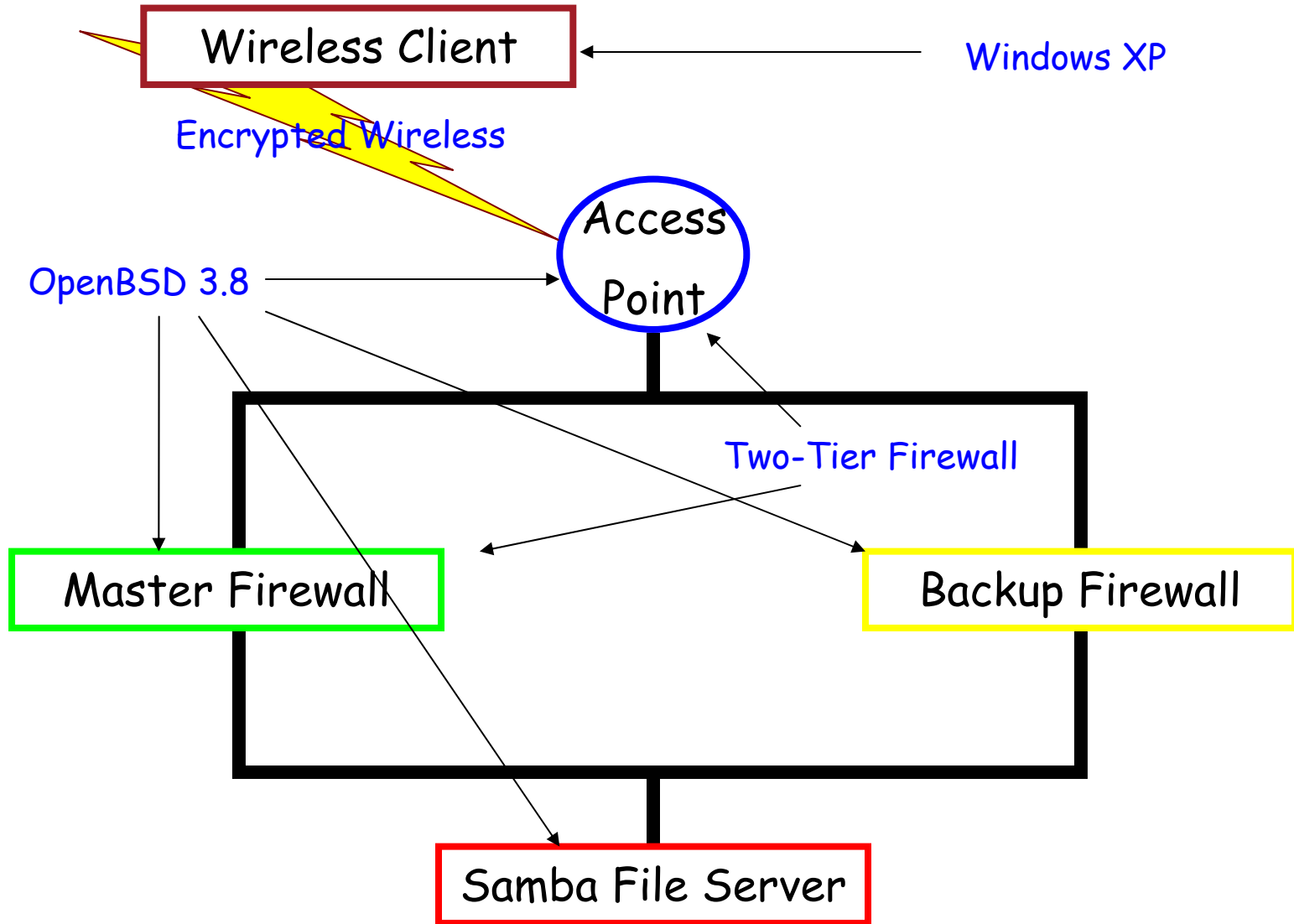
Master Firewall



A close up shot for the really curious



Demonstration #1 - Network Diagram



Demonstration #1 - Securing a Wireless Network

At this point, I am going to...

- connect to a Secure Wireless Access Point
- automatically start encryption using IPsec
- use OpenSSH with dual factor authentication
- upon successful authentication, a unique rule set is loaded
- I will then connect to the Samba File Server and
- play some music

We can easily secure Wireless Networks and demonstrate UN*X/Windows interoperability at its best without the need for additional commercial software or hardware!

Demonstration #1 - Wireless Network: Authentication

I authenticate using a private key and a strong passphrase.

```
C:\ Command Prompt
Main Mode SA #1:
From 172.16.1.254
To 172.16.1.1
Policy Id : <72442564-5C57-49E3-ADE0-AD7365BD455>
Offer Used :
    3DES MD5 DH Group 2
    Quickmode limit : 0, Lifetime 0Kbytes/288
Auth Used : Preshared Key
Initiator cookie 60016738bb1cd1f8
Responder cookie 2411c3e794490fb1
Source UDP Encap port : 500 Dest UDP Encap port: 500

Quick Mode SAs
-----
Quick Mode SA #1:
Filter Id : <EF6B0850-E4C8-49CC-9214-B1B4AE472338>
Tunnel Filter
From 172.16.1.254
To Any
Protocol : 0 Src Port : 0 Des Port : 0
Direction : Outbound
Tunnel From 172.16.1.254
Tunnel To 172.16.1.1
Policy Id : <415EE6D0-D17D-48FC-930F-5CB69FDD9ADE>
Offer Used :
    Algo #1 : Encryption 3DES MD5 <24bytes/0rounds> <16secbytes/0secrounds>
    MySpi 2421051995 PeerSpi 3145519196
    PFS : False, Lifetime 100000Kbytes/3600seconds
Initiator cookie 60016738bb1cd1f8
Responder cookie 2411c3e794490fb1

The command completed successfully.
C:\>
```

```
Main Mode SAs
-----
No SAs
Quick Mode SAs
-----
No SAs
The command completed successfully.
C:\>
```

Before IPsec

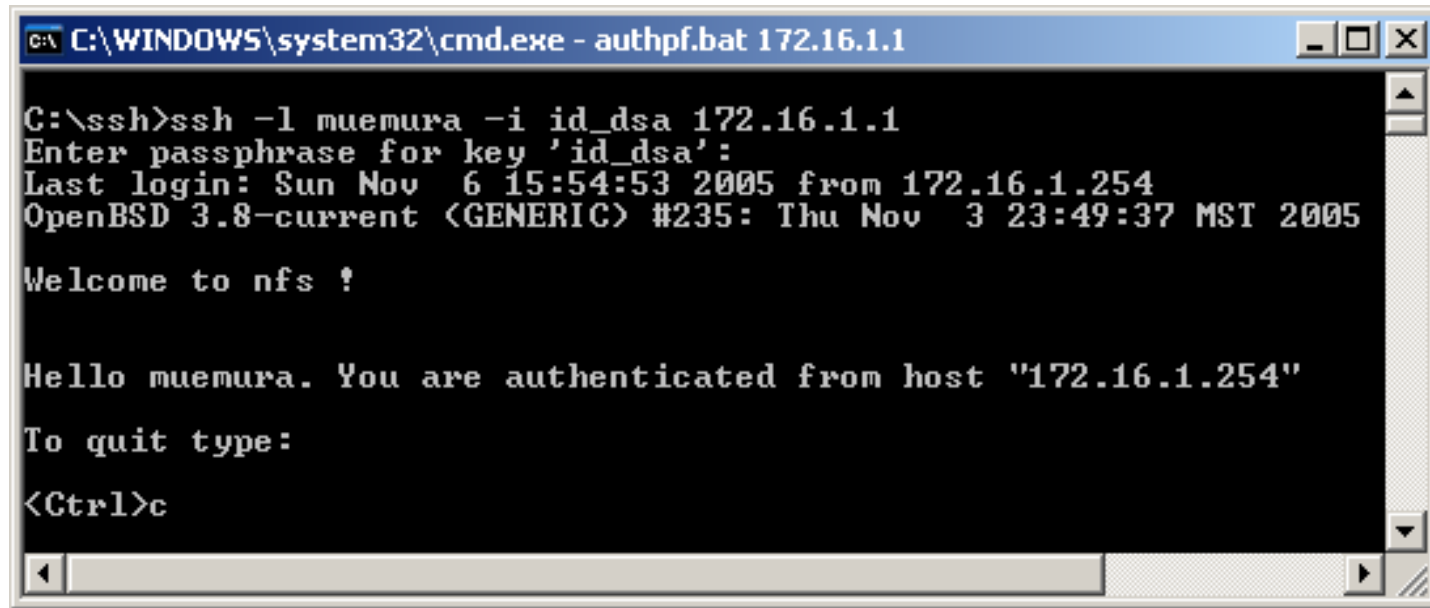
After IPsec

Authentication

```
C:\> C:\WINDOWS\system32\cmd.exe - authpf.bat 172.16.1.1
C:\ssh>ssh -l muemura -i id_dsa 172.16.1.1
Enter passphrase for key 'id_dsa':
```

Demonstration - Wireless Network: Authentication (Cont'd)

A new firewall rule set is loaded upon successful authentication.



```
C:\WINDOWS\system32\cmd.exe - authpf.bat 172.16.1.1
C:\ssh>ssh -l muemura -i id_dsa 172.16.1.1
Enter passphrase for key 'id_dsa':
Last login: Sun Nov  6 15:54:53 2005 from 172.16.1.254
OpenBSD 3.8-current <GENERIC> #235: Thu Nov  3 23:49:37 MST 2005

Welcome to nfs !

Hello muemura. You are authenticated from host "172.16.1.254"

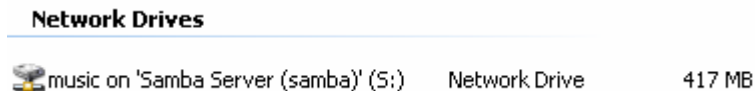
To quit type:
<Ctrl>c
```

```
[nfs] mtu $ netstat -rn -f encap
Routing tables
```

Output of netstat -rn -f encap on the SWAP

```
Encap:
Source      Port  Destination      Port  Proto SA(Address/Proto/Type/Direction)
172.16.1.254/32  0    0/0              0     0    172.16.1.254/50/use/in
0/0         0    172.16.1.254/32  0     0    172.16.1.254/50/require/out
[nfs] mtu $
```

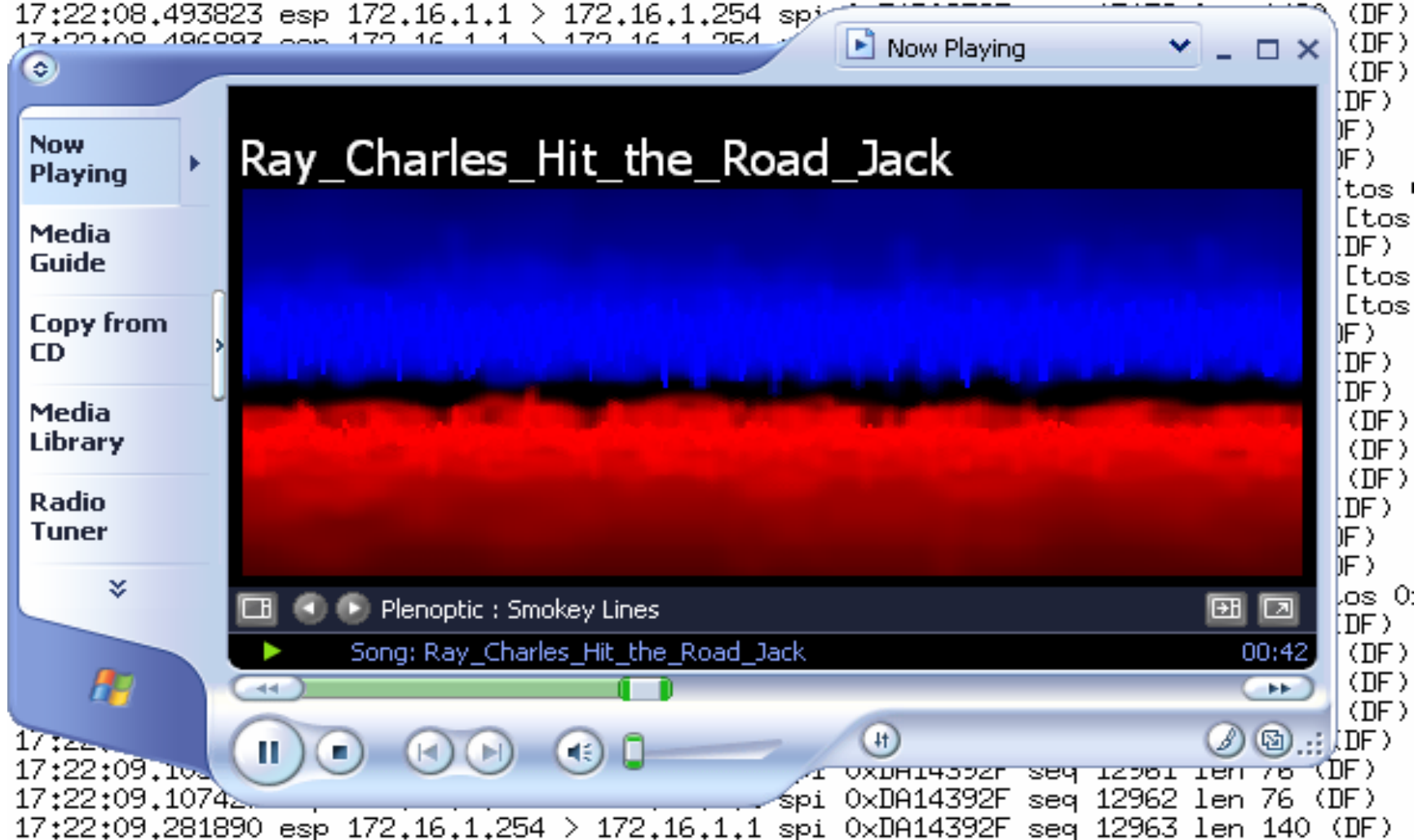
I can then map a Samba network drive share across the firewall and play some music.



Demonstration - Wireless Network: Data Access

Encrypted wireless network traffic

```
17:22:08.479387 esp 172.16.1.254 > 172.16.1.1 spi 0xDA14392F seq 12951 len 140 (DF)
17:22:08.493823 esp 172.16.1.1 > 172.16.1.254 spi 0xDA14392F seq 12952 len 140 (DF)
17:22:08.496893 esp 172.16.1.1 > 172.16.1.254 spi 0xDA14392F seq 12953 len 140 (DF)
17:22:08.499963 esp 172.16.1.254 > 172.16.1.1 spi 0xDA14392F seq 12954 len 140 (DF)
17:22:08.503033 esp 172.16.1.1 > 172.16.1.254 spi 0xDA14392F seq 12955 len 140 (DF)
17:22:08.506103 esp 172.16.1.254 > 172.16.1.1 spi 0xDA14392F seq 12956 len 140 (DF)
17:22:08.509173 esp 172.16.1.1 > 172.16.1.254 spi 0xDA14392F seq 12957 len 140 (DF)
17:22:08.512243 esp 172.16.1.254 > 172.16.1.1 spi 0xDA14392F seq 12958 len 140 (DF)
17:22:08.515313 esp 172.16.1.1 > 172.16.1.254 spi 0xDA14392F seq 12959 len 140 (DF)
17:22:08.518383 esp 172.16.1.254 > 172.16.1.1 spi 0xDA14392F seq 12960 len 140 (DF)
17:22:08.521453 esp 172.16.1.1 > 172.16.1.254 spi 0xDA14392F seq 12961 len 76 (DF)
17:22:08.524523 esp 172.16.1.254 > 172.16.1.1 spi 0xDA14392F seq 12962 len 76 (DF)
17:22:08.527593 esp 172.16.1.1 > 172.16.1.254 spi 0xDA14392F seq 12963 len 140 (DF)
```



The screenshot shows a Windows XP desktop environment. In the background, a network monitor window displays a list of encrypted wireless network traffic. The traffic is captured on the 'nfs - mtu' interface and shows ESP (Encapsulated Security Payload) packets between IP addresses 172.16.1.254 and 172.16.1.1. The traffic is encrypted using SPI 0xDA14392F. The sequence numbers range from 12951 to 12963, with varying lengths (140 and 76 bytes). The traffic is identified as '(DF)'. In the foreground, a 'Now Playing' window is open, displaying the song 'Ray Charles - Hit the Road Jack' by Plenoptic. The song is currently playing, and the progress bar is at approximately 00:42. The window title is 'Now Playing' and the artist is 'Plenoptic : Smokey Lines'. The album art shows a blue and red abstract design.

How did I setup the SWAP ?

Secure Wireless Access Point (SWAP)

- All you need is OpenBSD and a supported wireless card that you can run in hostap mode, i.e. as an Access Point. Hostap mode is not a requirement for this solution.
- We did nothing special to the OpenBSD installation or configuration. Everything that you will need is in the base install. Of course, you will have to define your firewall rule set to enforce your security policy and know how to set up ISAKMP but that's all.
- We used Commell EMB-564 embedded hardware because the CPU was fast enough to handle IPsec traffic. They came with four NICs and fairly good serial BIOS. There are no moving parts as the Operating System boots and runs off of a CF card.

... and what about the Wireless Windows Client?

Wireless Windows Client

- You need to install the native encryption support for Windows found in the Support Tools folder of the XPSP2 CD or you can download it from Microsoft on the Internet.
- For a truly automated process, you will need to install OpenSSH for Windows. Any ssh client will work but you will lose automation and that is really one of the strong points of this solution.
- You will need to create ssh keys for the client. You can use username/password authentication but dual factor authentication is what we recommend.

...the Wireless Windows Client continued

Wireless Windows Client

There are few simple files to be placed on the client:

- A Visual Basic Script file that determines the IP address and Gateway issued by the SWAP

This information is automatically fed into the following 3 bat files:

- One bat file initiates the IPsec tunnel.
- The other bat file starts the ssh authentication.
- The final bat file takes no arguments and is used to delete any existing tunnels.

The VBS file (error checking code removed)

```
Dim strMyGate
Dim strMyIP
On Error Resume Next
set IPConfigSet = \
    GetObject("winmgmts:{impersonationLevel=impersonate}!root\cimv2") \
    .ExecQuery("SELECT IPAddress, DefaultIPGateway, IPsecPermitTCPPorts \
    FROM Win32_NetworkAdapterConfiguration WHERE IPEnabled=TRUE")
for each IPConfig in IPConfigSet
    if Not IsNull(IPConfig.IPAddress) then
        for i = LBound(IPConfig.IPAddress) to \
            UBound(IPConfig.IPAddress)
            if IPConfig.IPAddress(i) <> "0.0.0.0" then
                strMyIP = IPConfig.IPAddress(i)
            end if
        next
    end if
end if
```


The VBS file continued

```
if Not IsNull(IPConfig.DefaultIPGateway) then
    for i=LBound(IPConfig.DefaultIPGateway) to \
        UBound(IPConfig.DefaultIPGateway)
        strMyGate = IPConfig.DefaultIPGateway(i)
    next
end if
next
Set d = CreateObject("WScript.Shell")
d.Run "cmd /c deletetunnel.bat "
WScript.Sleep 2000
Set t = CreateObject("WScript.Shell")
t.Run "cmd /c vpn.bat " & strMyGate & " " & strMyIP
WScript.Sleep 1000
Set w = CreateObject("WScript.Shell")
w.Run "cmd /k authpf.bat " & strMyGate
```

The bat files

vpn.bat

```
ipseccmd.exe -u
```

```
ipseccmd.exe -f 0=* -n ESP[3DES,MD5] -t %1 -a \
```

```
        P:"long-shared-secret" -1s 3DES-MD5-2
```

```
ipseccmd.exe -f *=0 -n ESP[3DES,MD5] -t %2 -a \
```

```
        P:"long-shared-secret" -1s 3DES-MD5-2
```

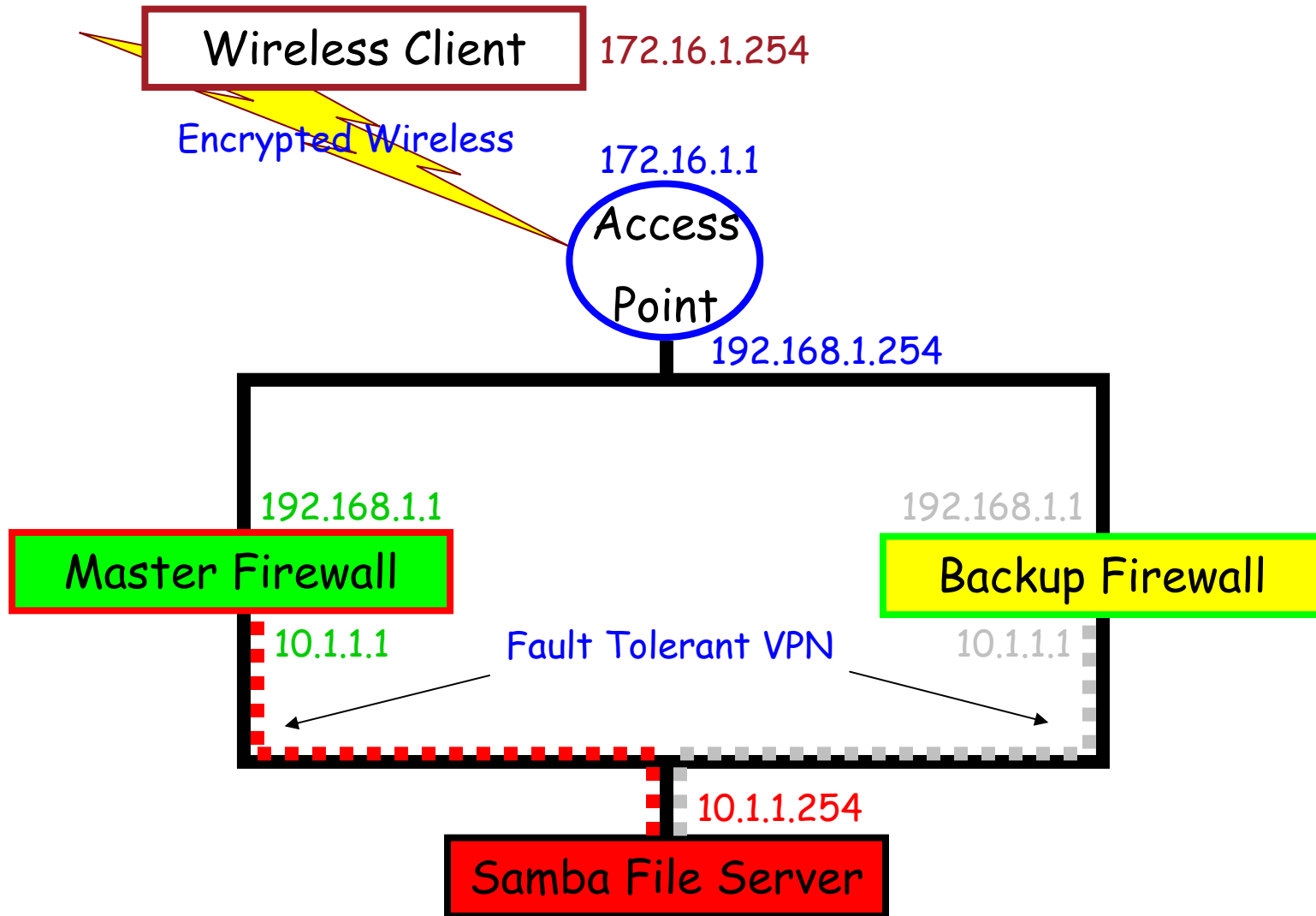
deletetunnel.bat

```
ipseccmd.exe -u
```

authpf.bat

```
ssh -l username -i username-openssh-dsa-2048-date %1
```

Demonstration #2 - High Availability VPN



Demonstration # 2 - High Availability VPN

```
[samba] etc $ netstat -rn -f encap
```

```
Routing tables
```

```
Encap:
```

Source	Port	Destination	Port	Proto	SA (Address/Proto/Type/Direction)
10.1.1.1/32	0	10.1.1.254/32	0	0	10.1.1.1/50/use/in
192.168.1/24	0	10.1.1.254/32	0	0	10.1.1.1/50/use/in
10.1.1.254/32	0	10.1.1.1/32	0	0	10.1.1.1/50/require/out
10.1.1.254/32	0	192.168.1/24	0	0	10.1.1.1/50/require/out

```
[fw01] etc $ netstat -rn -f encap
```

```
Routing tables
```

```
Encap:
```

Source	Port	Destination	Port	Proto	SA (Address/Proto/Type/Direction)
10.1.1.254/32	0	10.1.1.1/32	0	0	10.1.1.254/50/use/in
10.1.1.254/32	0	192.168.1/24	0	0	10.1.1.254/50/use/in
10.1.1.1/32	0	10.1.1.254/32	0	0	10.1.1.254/50/require/out
192.168.1/24	0	10.1.1.254/32	0	0	10.1.1.254/50/require/out

Demonstration # 2 - High Availability VPN

```
[fw02] etc $ netstat -rn -f encap
```

```
Routing tables
```

```
Encap:
```

Source	Port	Destination	Port	Proto	SA (Address/Proto/Type/Direction)
10.1.1.254/32	0	10.1.1.1/32	0	0	10.1.1.254/50/use/in
10.1.1.254/32	0	192.168.1/24	0	0	10.1.1.254/50/use/in
10.1.1.1/32	0	10.1.1.254/32	0	0	10.1.1.254/50/require/out
192.168.1/24	0	10.1.1.254/32	0	0	10.1.1.254/50/require/out

```
[laptop] etc $ ping samba
```

(before encryption)

```
PING samba (10.1.1.254): 56 data bytes
```

```
64 bytes from 10.1.1.254: icmp_seq=0 ttl=254 time=0.805 ms
```

```
64 bytes from 10.1.1.254: icmp_seq=1 ttl=254 time=0.435 ms
```

```
64 bytes from 10.1.1.254: icmp_seq=2 ttl=254 time=0.419 ms
```

```
...
```

(after after encryption)

```
64 bytes from 10.1.1.254: icmp_seq=34 ttl=254 time=2.090 ms
```

```
64 bytes from 10.1.1.254: icmp_seq=35 ttl=254 time=2.094 ms
```

```
64 bytes from 10.1.1.254: icmp_seq=36 ttl=254 time=2.182 ms
```

Demonstration #2 - High Availability VPN (Live)

Music is encrypted and the Master Firewall (VPN peer) is rebooted...

Encrypted Music and ICMP traffic is flowing through the Master VPN

```
18:16:15.355561 esp 172.16.1.254 > 172.16.1.1 spi 0x8353FB8D seq 1405 len 76 (DF
18:16:16.350835 esp 172.16.1.1 > 172.16.1.254 spi 0xCDF5365C seq 1951 len 204 [t
18:16:16.351923 esp 172.16.1.1 > 172.16.1.254 spi 0xCDF5365C seq 1952 len 204 [t
18:16:16.354558 esp 172.16.1.254 > 172.16.1.1 spi 0x8353FB8D seq 1406 len 76 (DF
```

Now Playing

James_Brown_I_Got_You

[nfs] mtu \$ ping samba
PING samba (10.1.1.254): 56 data bytes
64 bytes from 10.1.1.254: icmp_seq=0 ttl=254 time=2.755 ms
64 bytes from 10.1.1.254: icmp_seq=1 ttl=254 time=2.335 ms
64 bytes from 10.1.1.254: icmp_seq=2 ttl=254 time=2.897 ms
64 bytes from 10.1.1.254: icmp_seq=3 ttl=254 time=21.770 ms

64 bytes from 10.1.1.254: icmp_seq=9 ttl=254 time=2.653 ms
64 bytes from 10.1.1.254: icmp_seq=10 ttl=254 time=2.095 ms
64 bytes from 10.1.1.254: icmp_seq=11 ttl=254 time=2.287 ms
64 bytes from 10.1.1.254: icmp_seq=12 ttl=254 time=2.074 ms
64 bytes from 10.1.1.254: icmp_seq=13 ttl=254 time=2.112 ms

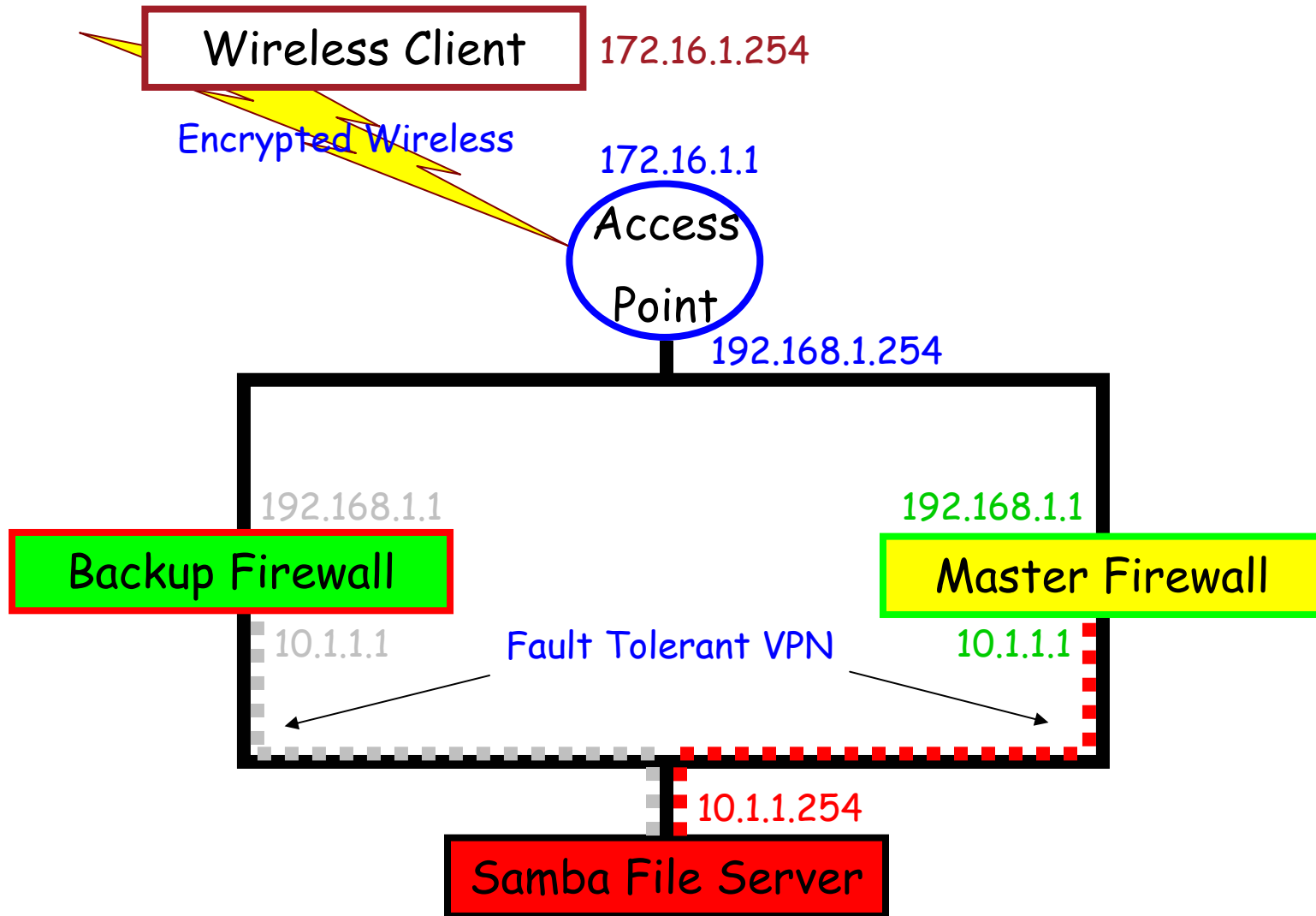
00:49

A little pause but not much and the music plays on.

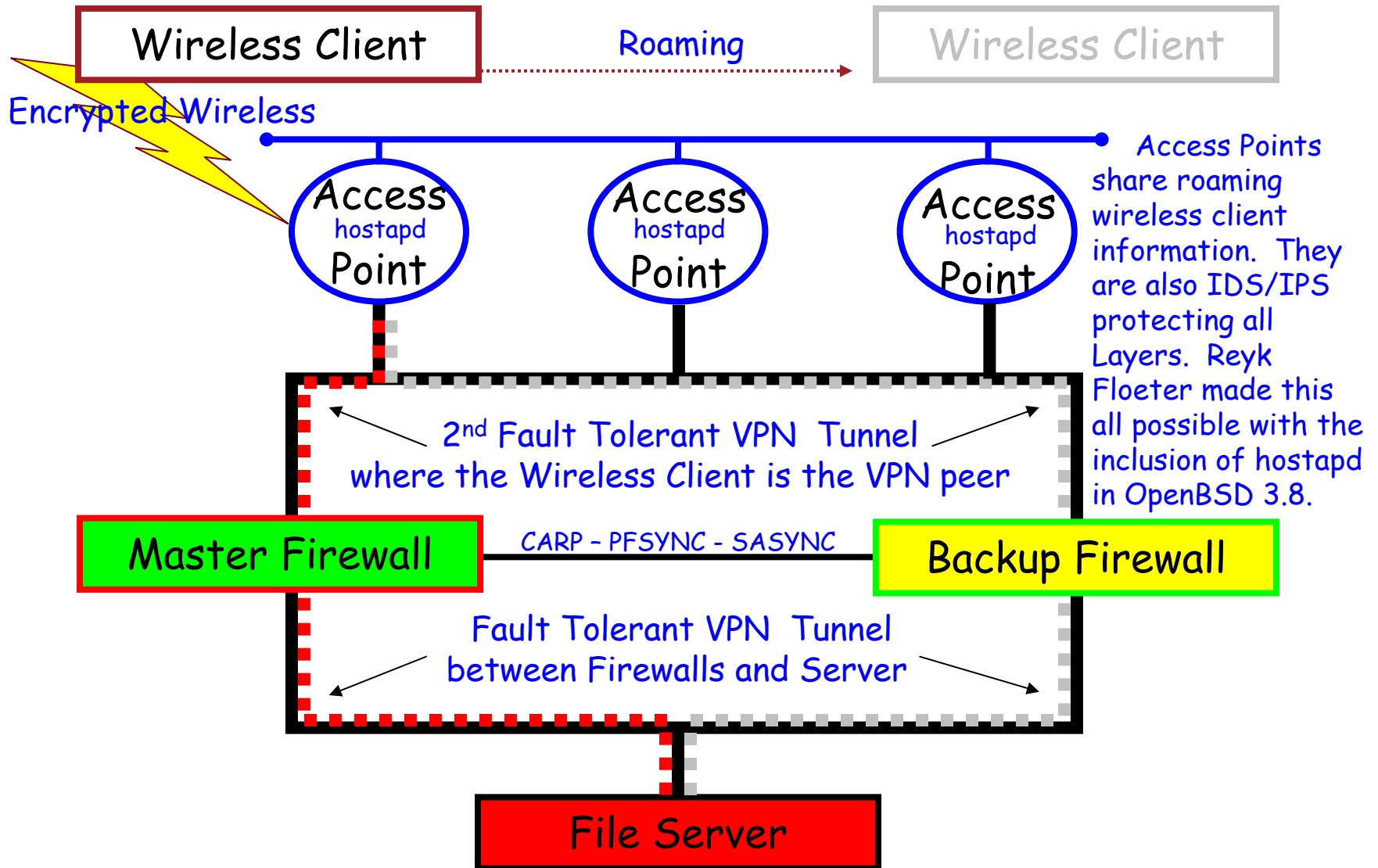
All traffic now flows through the Backup VPN Peer

```
16:24.351... spi 0xCDF5365C seq 1969 len 204 [t
16:24.35485... spi 0x8353FB8D seq 1415 len 76 (DF
16:25.350824 esp 172.16.1.1 > 172.16.1.254 spi 0xCDF5365C seq 1970 len 204 [t
16:25.351930 esp 172.16.1.1 > 172.16.1.254 spi 0xCDF5365C seq 1971 len 284 [t
```

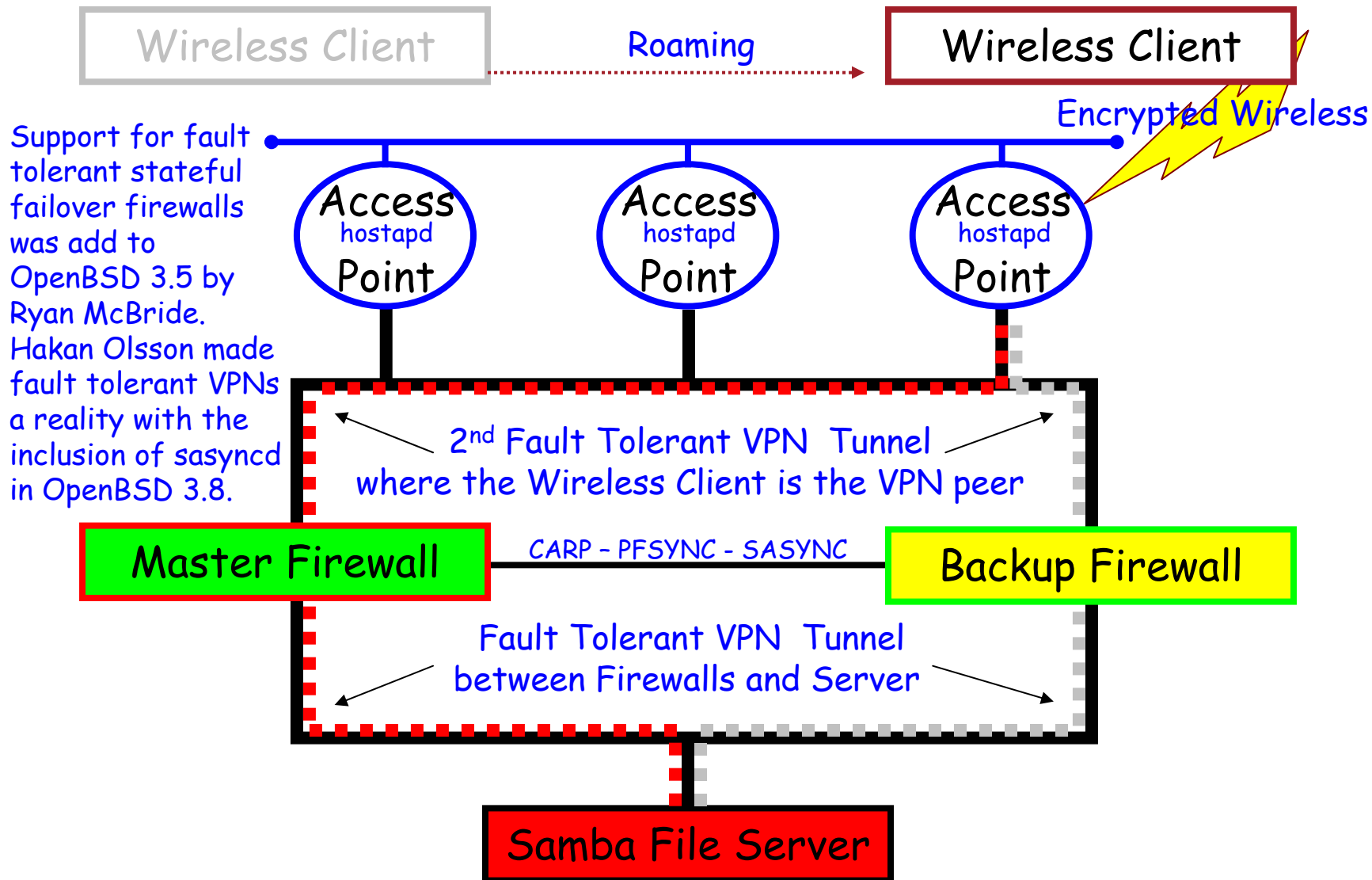
Demonstration #2 - High Availability VPN



Secure High Availability Wireless Solution (The Future)



Secure High Availability Wireless Solution (The Future)



Support for fault tolerant stateful failover firewalls was added to OpenBSD 3.5 by Ryan McBride. Hakan Olsson made fault tolerant VPNs a reality with the inclusion of sasyncd in OpenBSD 3.8.

Summary

OpenBSD and OpenSSH gives us wireless networking with:

- Strong Encryption
- Strong Dual Factor Authentication
- Firewall Protection on the Access Points
- Intrusion Detection and Prevention on the Access Points
- Fault Tolerant Stateful Firewalls
- Fault Tolerant Stateful VPN support
- Support for roaming wireless clients keeping state
- mobility without having to compromise on security
- a highly automated and very user friendly solution
- an Open Source solution that interoperates with Windows
- all for *free* and with
- *no dependence* on Commercial hardware or software

Questions?

<http://openbsd.org>

<http://openssh.org>

<http://openbsd-support.com>

A special thanks to:

Theo de Raadt

Ryan McBride

Bob Beck

David Gwynne

Jonathan Gray

Michael Paddon

Damien Miller

Reyk Floeter

Brent Uemura

**for their help with
this presentation.**

This presentation
would not be if it
wasn't for the:

OpenBSD Developers

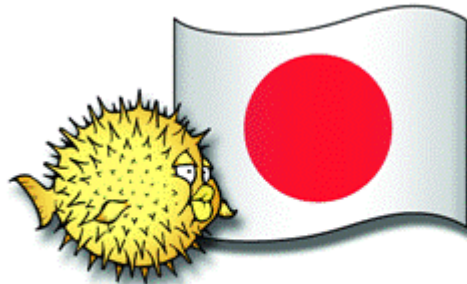
OpenSSH Developers

Documentation Team

and

the companies and
individuals that support
the ongoing development
of OpenBSD and
OpenSSH with regular
CD purchases and
donations.

Thank you very much!



OpenBSD Support Japan Inc.

Secure Open Source Solutions for your business