

Secure Open Wireless Networking

Backup Talk for Blackhat USA 2011

Christopher Byrd, Tom Cross, & Takehiro
Takahashi

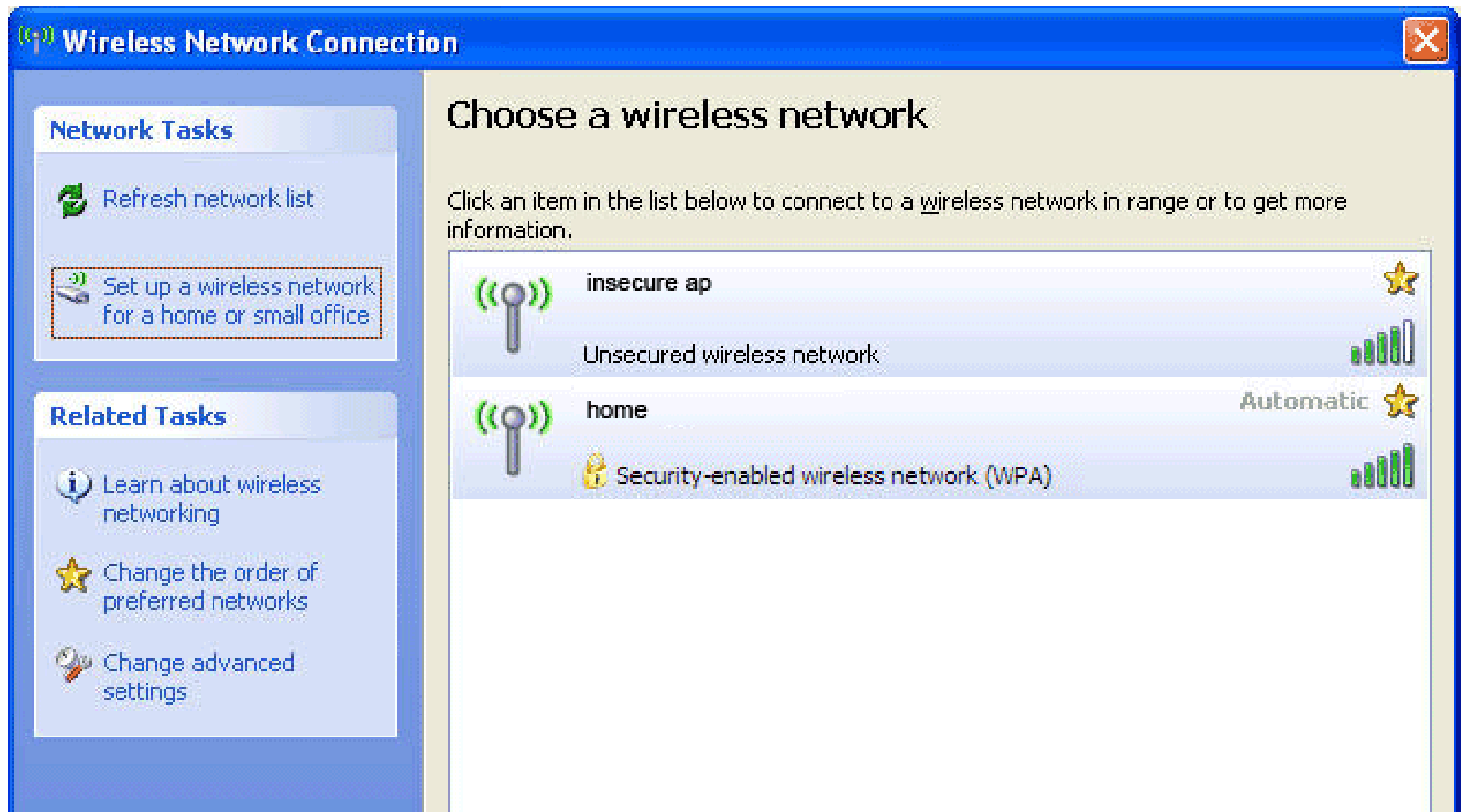
Elevator Pitch

- Free Internet is good! We love open wireless...
- Unencrypted Internet is bad. We don't want to be spied on while we surf the web.
- 802.11 networks are either access controlled, or unencrypted. :-/
- We can solve this problem by encrypting our wireless access using a digital certificate that is uniquely tied to the SSID of the access point. This is similar to the way that HTTPS works, but uses SSIDs instead of Domain names to identify the network provider.
- This talk will explain how this works and what the advantages of this approach are.
- We're releasing code for this under the GPL. Please play with it.

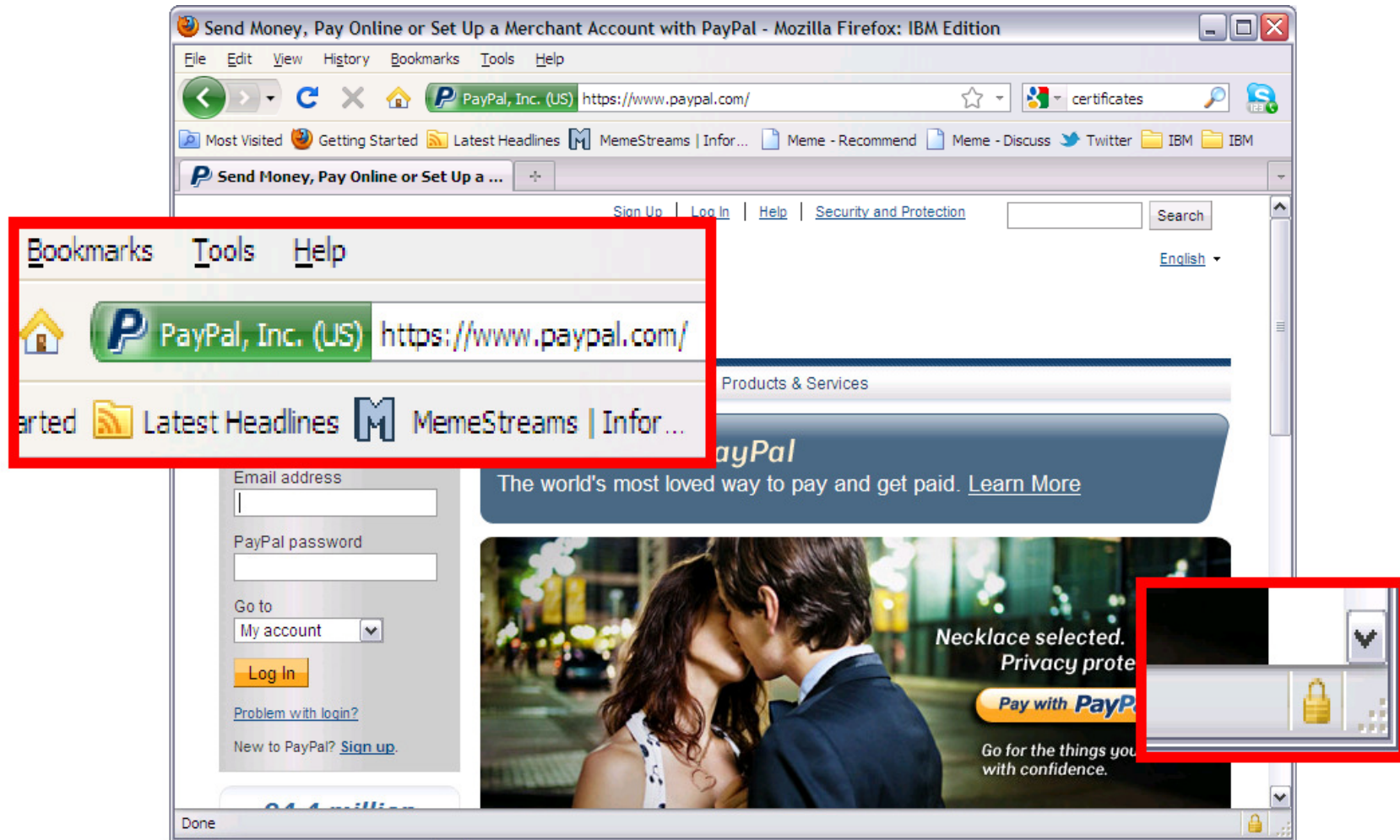
Two Paths, One Solution

- Christopher released paper on “Open Secure Wireless”
- Tom and Takehiro released paper on “Secure Open Wireless Networking”
- Both were developed in parallel independently
- Both have same fundamental approach, differ in some details
- We are working together to make this a reality

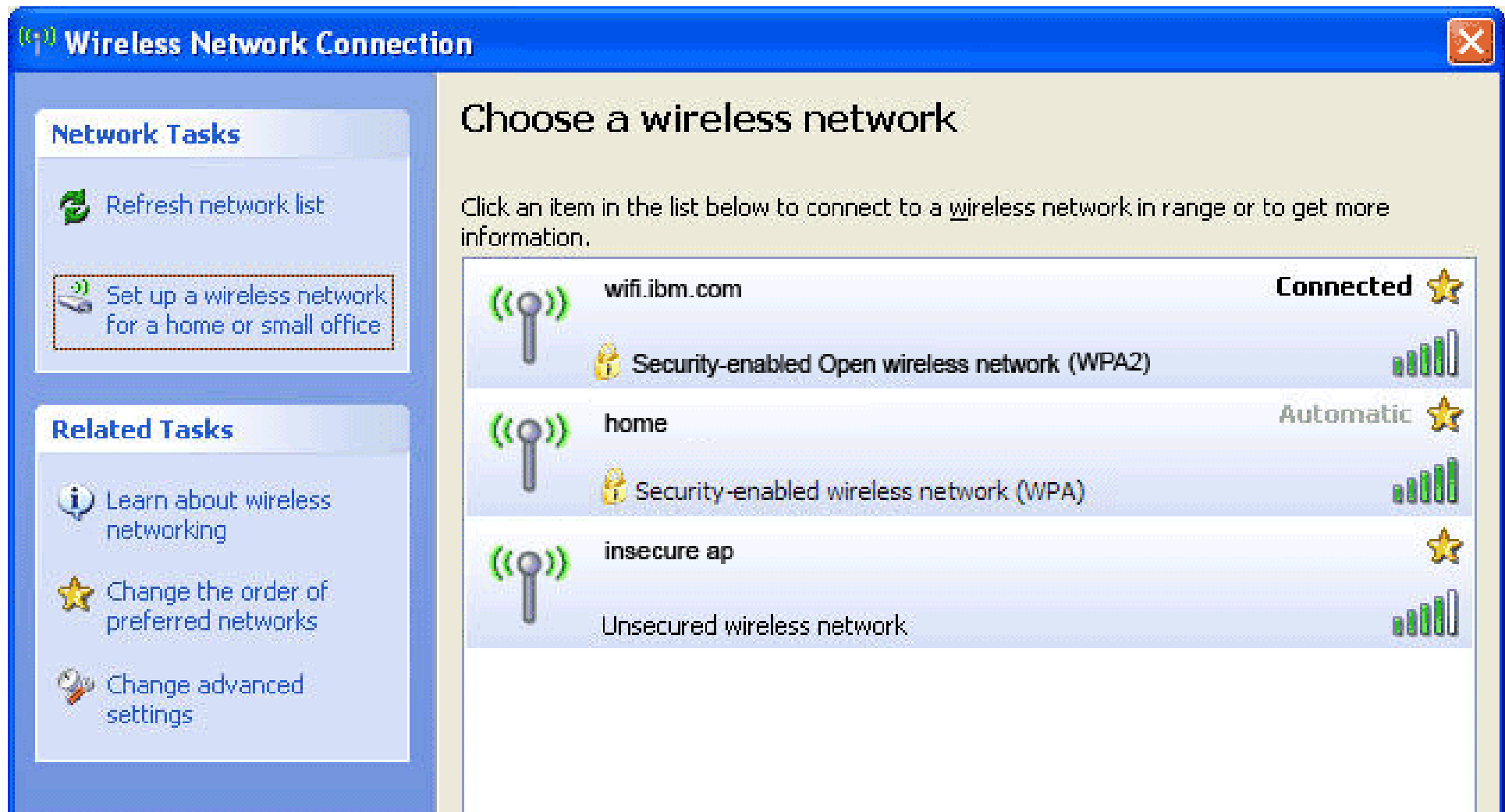
Wireless Networking today is a Hobson's choice – require auth or don't encrypt.



But, we can establish encrypted connections to websites without a password...



Why not do the same thing with wireless networks?



Advantages

- Your traffic is encrypted from your laptop to the access point.
- You don't have to have a login, password, or other previously established access credential.
- You know what organization is running the network you've connected to. Only IBM could obtain a digital certificate for wireless.ibm.com.

The three main objections we've heard:

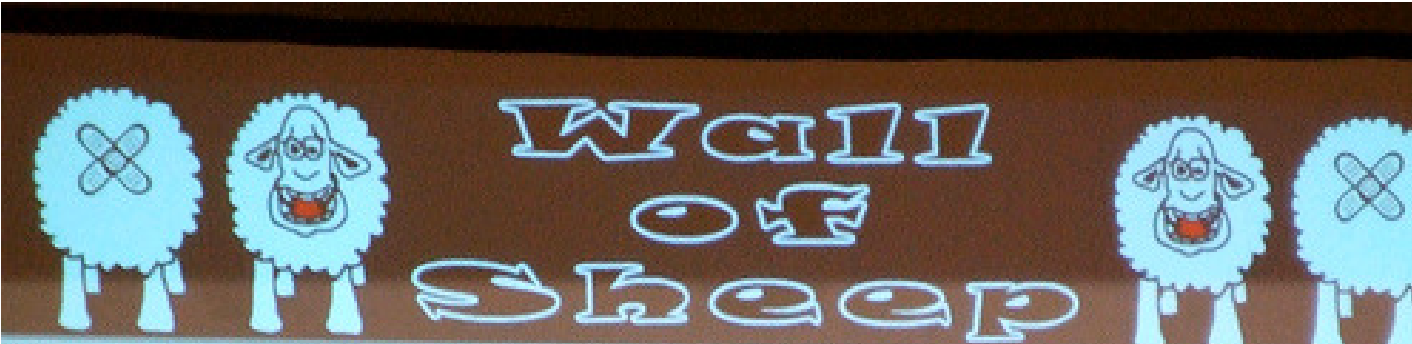
1. We don't need this – HTTPS solves all of the problems with insecure wireless networking.
2. HTTPS doesn't work, so why would this work? People are too stupid to tell the difference between secure and insecure networks. They'll click on anything!
3. You haven't done enough science.

Addressing the Objections:

1. We don't need this – HTTPS solves all of the problems with insecure wireless networking.

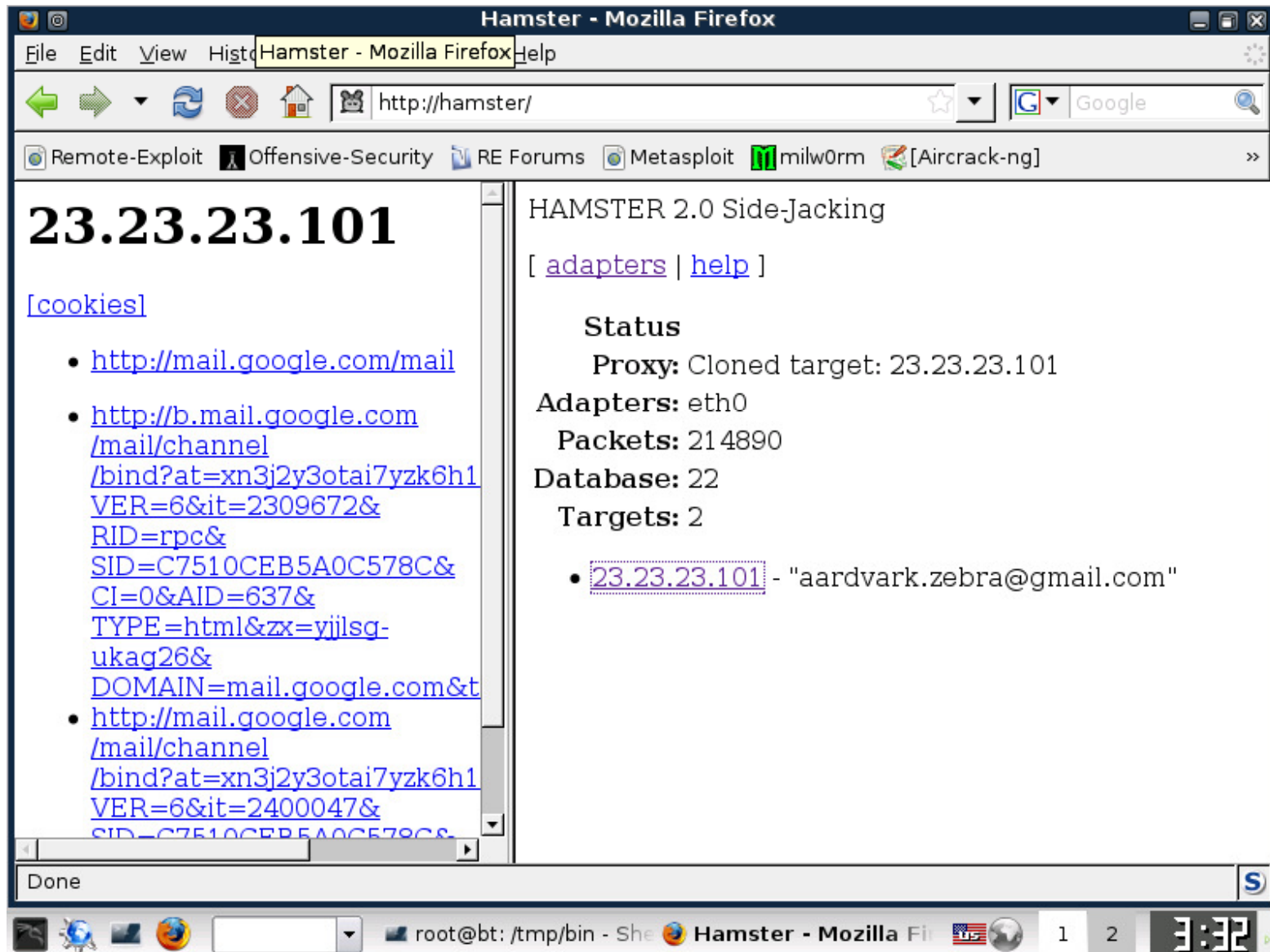
Over the years there have been many demonstrations of the security risks associated with unencrypted wireless.

The Wall of Sheep (Since 2001)

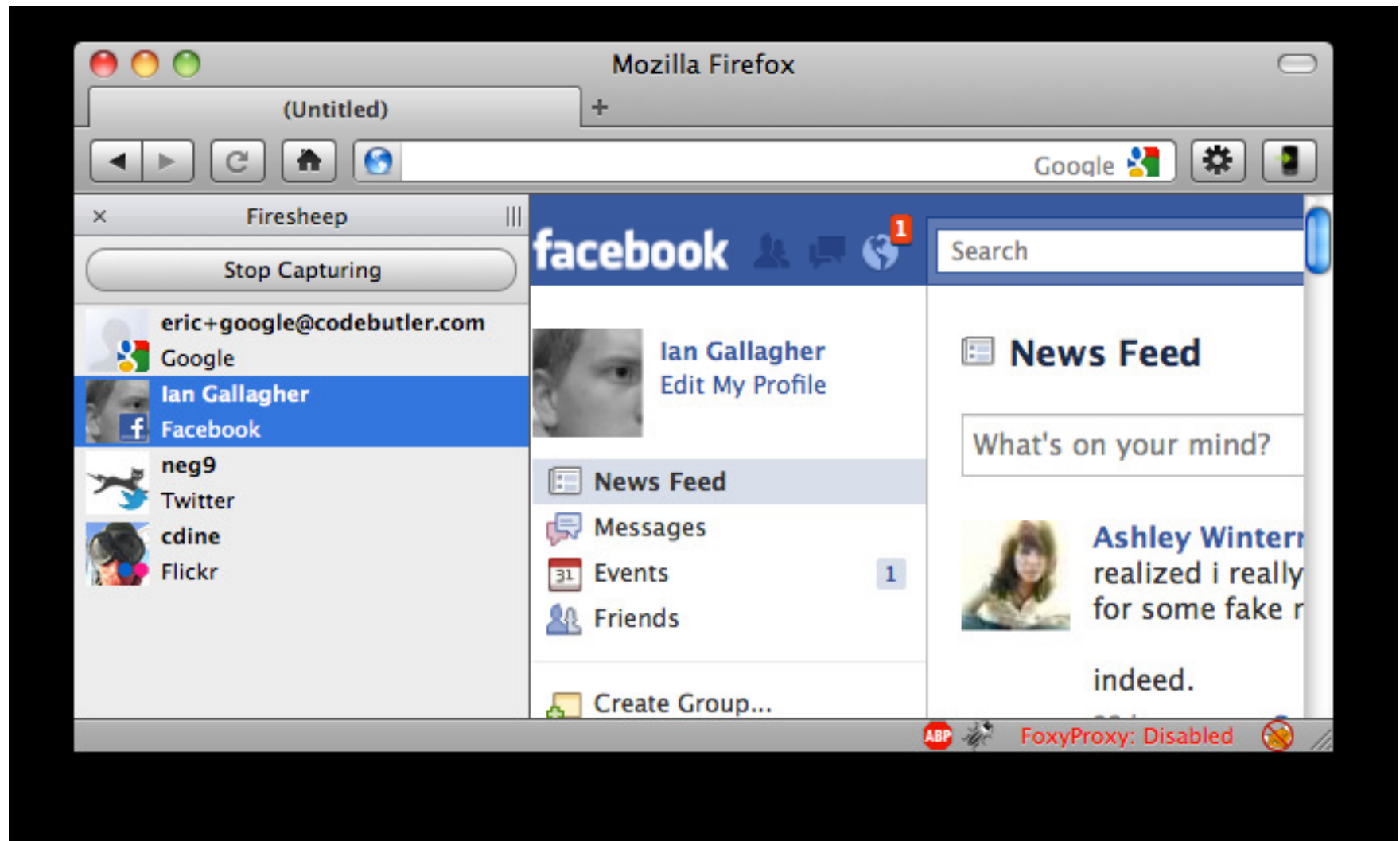


login	pass	domain_ip	application	MAC ADDR
collins	lah*****	monitor.hawaii.gov	HTTP	xx:xx
dvbg	app*****	mail.mae.com	HTTP	xx:xx
cyrus001	216*****	keroraa.org	HTTP	xx:xx
tajjibz	ent*****	holysar.net	HTTP	xx:xx
Lupin1004	900*****	ktfsmembers.com	HTTP	xx:xx
N0C'DC'isNotFukingWithU	Man*****	www.warfactory.net	HTTP	xx:xx
motorola	hac*****	cisco.ru.hackerarussia	HTTP	xx:xx
pantherghn	bik*****	myspace.com	HTTP	xx:xx
hunspower	int*****	pop.west.cox.ne	POP	xx:xx
guyman	Dud*****	www.chroniclesofreal.co	HTTP	xx:xx
arm_619	ll*****	myspace.com	HTTP	xx:xx
stealthdc5tusha	hon*****	yahoo.com	HTTP	xx:xx
Ben.Lucas	bon*****	forum.geekssquadforums	HTTP	xx:xx
web61pl	des*****	62.153.103.10	IMAP	xx:xx
adam	pen*****	66.218.85.170	POP	xx:xx
bmedis@innovered.com	xy7*****	innoveted.com	POP	xx:xx
vik	chl*****	195.47.75.55	POP	xx:xx
noah@noahcomputers.net	ave*****	myspace.com	HTTP	xx:xx
thiland	nop*****	cisco.com	HTTP	xx:xx
1747***9344	z9h*****	sipphone.com	SIP (VoIP)	xx:xx
erealas	qwo*****	192.117.154.52	HTTP	xx:xx

SideJacking (2007)



FireSheep (2010)



SSL Adoption

- Demonstrations have prompted major websites to adopt HTTPS to a certain extent.
 - Some have only adopted SSL for logins rather than allowing full SSL sessions.
 - If site doesn't go "all in" with HTTPS only and HSTS, then MitM is always an option
- Adoption of HTTPS is helpful, but
 - It will never be adopted by every site.
 - Gaps in some HTTPS implementations have exposed cookies. (i.e. XMLHttpRequest)
 - Side-channel leaks such as AJAX SSL packet size information disclosure
- Its not sufficient to solve all of the problems associated with insecure wireless.
 - Encryption of various IP protocols presents a variety of different ease of use and architectural challenges.

The AOL Search Data

- In August, 2006 AOL released 3 months of search keywords for over 650,000 users.
- Some real people were identified based entirely upon their search histories within this dataset.
- Several search histories included personal information such as social security numbers, credit card numbers, full names, addresses, and dates of birth.

Privacy Concerns

- Have you ever performed a web search of a personal nature?
 - At a hotel?
- Do we expect every site to adopt HTTPS?
 - The fact is that this is not happening.
- Even if every site adopts HTTPS, you're still broadcasting a list of every site you are visiting to everyone within range of your transmitter.
- Wired networks may be going away.
 - The latest Apple MacBook Air does not have a built in Ethernet port.
- **These privacy concerns alone ought to be enough to prompt the adoption of a more secure solution.**

Rogue Access Points

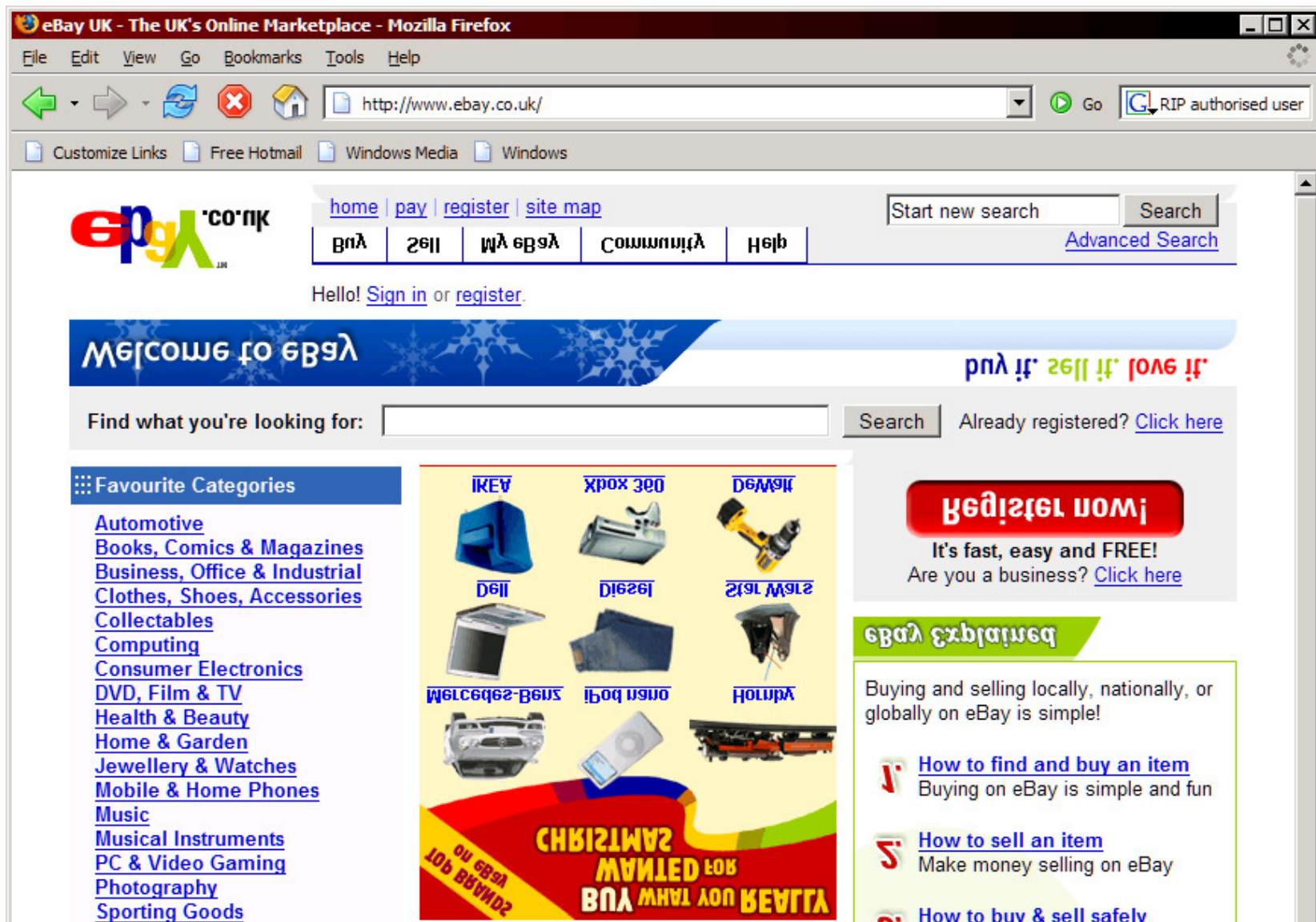
- The WiFi Pineapple
- Responds positively to Probe Request frames and provides an attacker controlled connection to the victim.
- “Of course all of the Internet traffic flowing through the pineapple such as e-mail, instant messages, and browser sessions are easily viewed or even modified by the pineapple holder.”
- Available for \$99 at HakShop.com



Airjack – Monkey-Jack

- Advanced 802.11 Attack Techniques
 - Blackhat 2002 - Mike Lynn & Robert Baird
- Attacker spoofs 802.11 Deauthenticate frame to disassociate victim from legitimate AP
- Attacker assumes legitimate AP's MAC and SSID on a different wireless channel
- Victim finds attacker's AP while scanning channels for the legitimate AP and connects to the attacker instead
- Attacker connects to the legitimate AP and bridges the two connections

Upside-down-ternet



SSL Strip

- “New Tricks for defeating SSL in Practice” - Moxie Marlinspike – Blackhat Federal 2009
- Proxy that changes HTTPS links into HTTP links on the fly in network traffic (and does some other stuff you’d want to have along with that).
- Users often don’t notice that encryption should be in use.

Active Man-in-the-Middle Attacks

- Presented at OWASP AU by IBM Rational Researchers Roi Saltzman and Adi Sharabani
- Presents a set of HTTP layer attacks that can be performed by a rogue access point that can force a victim to fetch files through injected iframes.
- One interesting suggestion is poisoning the HTTP cache with Javascript that will later be loaded in a secure context, such as an Intranet website.

Attacking the Client Host

- An active man-in-the-middle attack can inject malformed content that exploits remote code execution vulnerabilities in the victim's client software.
- This allows targeted client exploitation...
 - ...at the bar or coffee shop across the street from a business.
 - ...at the airport where people who work for many companies travel.
- Victims could be identified based on their homepage choices or the destinations of their VPN connections.
- Business travel locations are crawling with high value targets.

Malicious Captive Gateways

- Open Wireless networks often charge their users for access by collecting credit card numbers.
- Fortunately, these captive gateways often use SSL.
- Unfortunately, when you connect to a network you have no idea what gateway address you are going to be sent to.

Malicious Captive Gateways

- In practice, we've found that the URLs used by captive portals are often unrelated to the SSIDs of the network or the name of the network provider.
- This can make it hard for users to know what to trust.
- The opportunity for attackers to set up their own SSL encrypted credit card collection site is certainly there.

Theft of Service

- Its trivial for an attacker to spoof the MAC address of an authorized user of an unencrypted wireless network after the user has authenticated to a captive gateway.
- Encryption is required to address this problem.

Addressing the Objections:

2. HTTPS doesn't work, so why would this work? People are too stupid to tell the difference between secure and insecure networks. They'll click on anything!

The problem with HTTPS

- Most web surfing involves insecure websites – unencrypted connections are the “normal” situation as you surf the web.
- If a warning message popped up every time you accessed a “normal” website, the web would be very annoying.
- Indicators that encryption is in use can be subtle.

HTTPS demands a lot from users.

- As they surf the web, users should:
 - KNOW the difference between HTTP and HTTPS.
 - Constantly BE AWARE, while they surf, of when HTTPS is needed.
 - LOOK for the subtle signs that HTTPS is enabled.
 - INTERPRET those signs correctly.

Secure Open Wireless is different.

- Users are only required to make a security choice once, while initially connecting to the network, and not constantly while they surf the web.



- Unencrypted connections can always result in a clear warning message without breaking up the user's workflow.
- Encrypted connections can become the “normal” case.
- The SSID of the network that users are connecting to is protected and clear to the user. This is different from HTTPS, where misleading hyperlinks and graphics can distract the user from the actual URL of the host they are communicating with.

Its worth asking advocates of this point of view why people continue to adopt HTTPS and why they continue to spend money on SSL certificates.

Branding will be important.

- Today, anyone can run an access point with any SSID. Anyone can run a network called “IBM Wireless” or “AT&T Wireless” and there is no way that even a sophisticated user can tell if that network is really run by IBM or AT&T.
- Secure Open Wireless creates a mechanism whereby a particular SSID can be owned by a particular network provider and no one else can use it.
- This allows users to know who they are communicating with.
 - Only IBM will be able to run a network called “IBM Wireless.”
 - Only AT&T will be able to run a network called “AT&T Wireless.”
- Network Operators want an SSID that people will trust.
 - Generic sounding names don’t convey the unique identity of the network operator.
 - We’ll have to teach users to avoid using networks with names that they don’t know or trust.

Branding will be important.

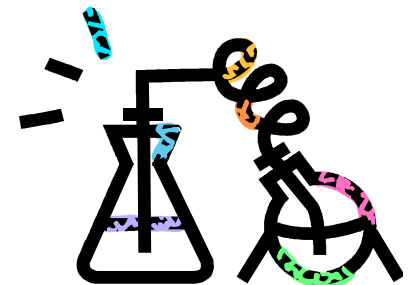
- Network Operators will need to monitor to detect the registration of potentially misleading SSIDs.
 - “IBM Wire less”
 - A central database of names used for secure wireless will be needed to prevent duplicates and monitor for attacks.
 - Trademark claims could be a useful tool here as this is all about preventing user confusion.

Addressing the Objections:

3. You haven't done enough science.

Will users make the right choices?

- Will most users avoid networks with generic sounding names and disconnect when they see unexpected warning messages?
- There might be an academic paper in security usability for someone who had the time to perform a controlled study on people's reactions in different contexts to the user interface ideas presented here.



Is that really the right question?

Random, Feckless Users

Lets assume that all users are completely unable to choose between a secure and insecure network, regardless of the interface queues and warning messages placed in front of them.

When presented with a choice between a secure network and an insecure network, the totally random user will choose the secure network half the time and the insecure network half the time.

The situation today is that all open wireless networks are unencrypted, and that all traffic on these networks is subject to attack by passive sniffers.

Connections that employ HTTPS are (usually) immune but as previously established in this presentation, a lot of important traffic is still in the clear and still susceptible to attack.

Adoption of Secure Open Wireless Access would eliminate the threat of passive sniffers almost completely, without requiring every website on the Internet to adopt HTTPS.

This raises the bar for attackers – they must setup a rogue access point and cannot merely download a sniffer.

When attackers do set up rogue access points, they are going to be operating them in competition with secure access points.

When faced with two choices, our random feckless users will pick the secure access points half the time.

We have, therefore, significantly reduced the attack surface area. The threat of passive sniffers is eliminated and rogue access points are less effective than they would otherwise be.

This progress is valuable

- Secure Open Wireless Access moves the problem of wireless security from being a technical problem to being a social problem.
- As Secure Open Wireless gains popularity, user familiarity will increase naturally.
- User actions can be improved through:
 - Training and awareness
 - Context
 - If I'm at a Hilton hotel should I pick the Hilton wireless access point or the CoolFreeWireless access point?
- We could choose to limit the availability of certificates to highly vetted network operators rather than allowing anyone to purchase them.
 - The effect of this would be to make the interface differences between secure and rogue networks even more stark.

Alternative Approaches?

What about VPNs?

- Some services allow users of suspect access points to encrypt their traffic to a trusted destination on the Internet.
 - This is inefficient and therefore expensive.
 - It has to be set up beforehand.
 - Not for occasional travelers
- Besides, what happens if the attacker just DoSes the VPN connection?

Guest accounts?

- George Ou – PEAP with guest account
- Chester Wisniewski – WPA/WPA2-PSK with the password “Free Wifi”
- Wisniewski’s approach can be decrypted because everyone knows the key and the session key derivation is in the clear.
- In PEAP the session key derivation occurs within a TLS tunnel.
- Neither solution protects against rogue access points.

How does SOWN work?

- WPA/WPA2 – AES/TKIP encryption
- EAP-TLS anonymous authentication
 - A flavor of Extensible Authentication Protocol (EAP)
 - One-way TLS negotiation – only requires server certificate to complete 802.1x authentication

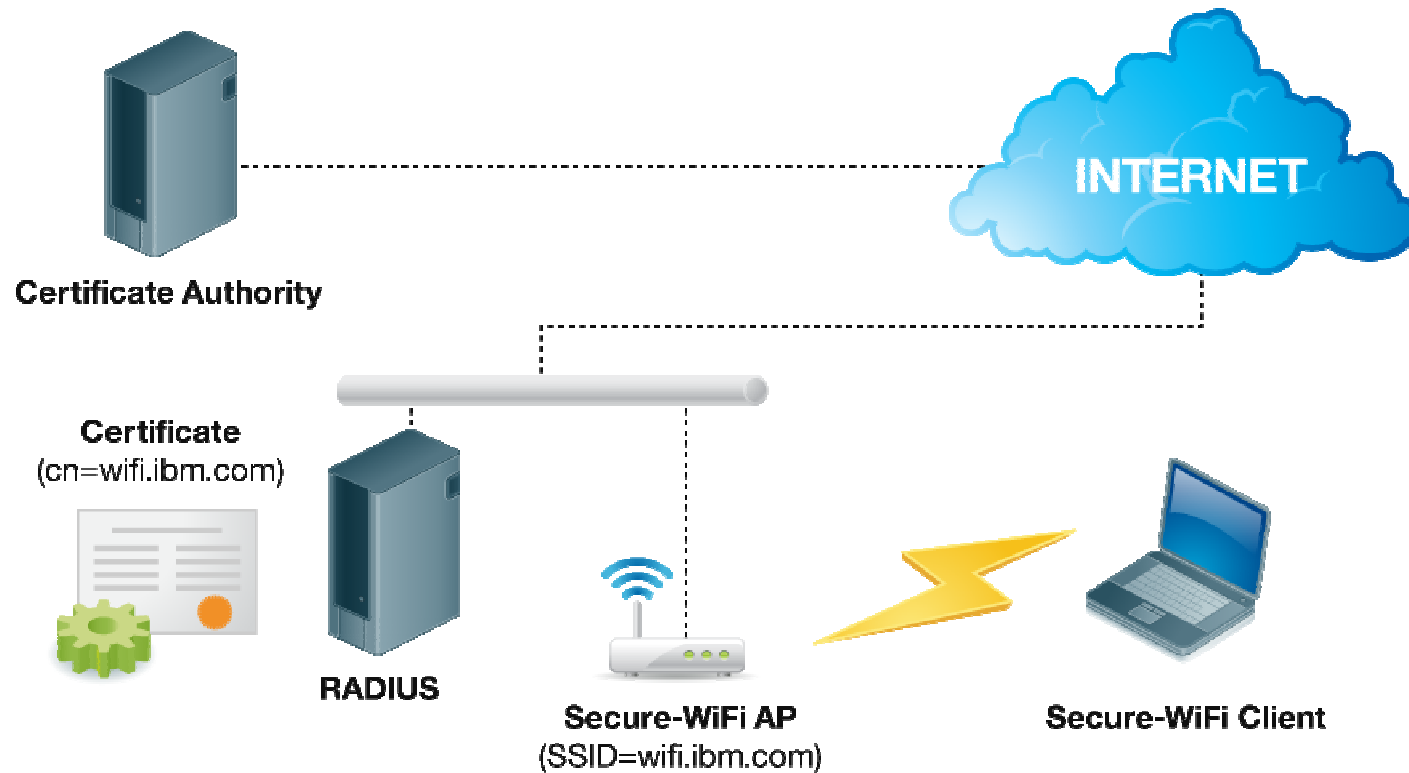
Authenticating Peer	Authenticator
-----	-----
	<- EAP-Request/ Identity
EAP-Response/ Identity (MyID) ->	
	<- EAP-Request/ EAP-Type=EAP-TLS (TLS Start)
EAP-Response/ EAP-Type=EAP-TLS (TLS client_hello)->	
	<- EAP-Request/ EAP-Type=EAP-TLS (TLS server_hello, TLS certificate, [TLS server_key_exchange, TLS certificate_request, TLS server_hello_done)
EAP-Response/ EAP-Type=EAP-TLS (TLS certificate, TLS client_key_exchange, TLS certificate_verify, TLS change_cipher_spec, TLS finished) ->	
	<- EAP-Request/ EAP-Type=EAP-TLS (TLS change_cipher_spec, TLS finished)
EAP-Response/ EAP-Type=EAP-TLS ->	
	<- EAP-Success

RFC5216

The certificate_request message is included when the server desires the peer to authenticate itself via public key. While the EAP server SHOULD require peer authentication, this is not mandatory, since there are circumstances in which peer authentication will not be needed (e.g., emergency services, as described in [UNAUTH]), or where the peer will authenticate via some other means.

If the server omits the certificate_request, the client can omit the certificate and certificate_verify in the EAP-Response.

How does SOWN work?



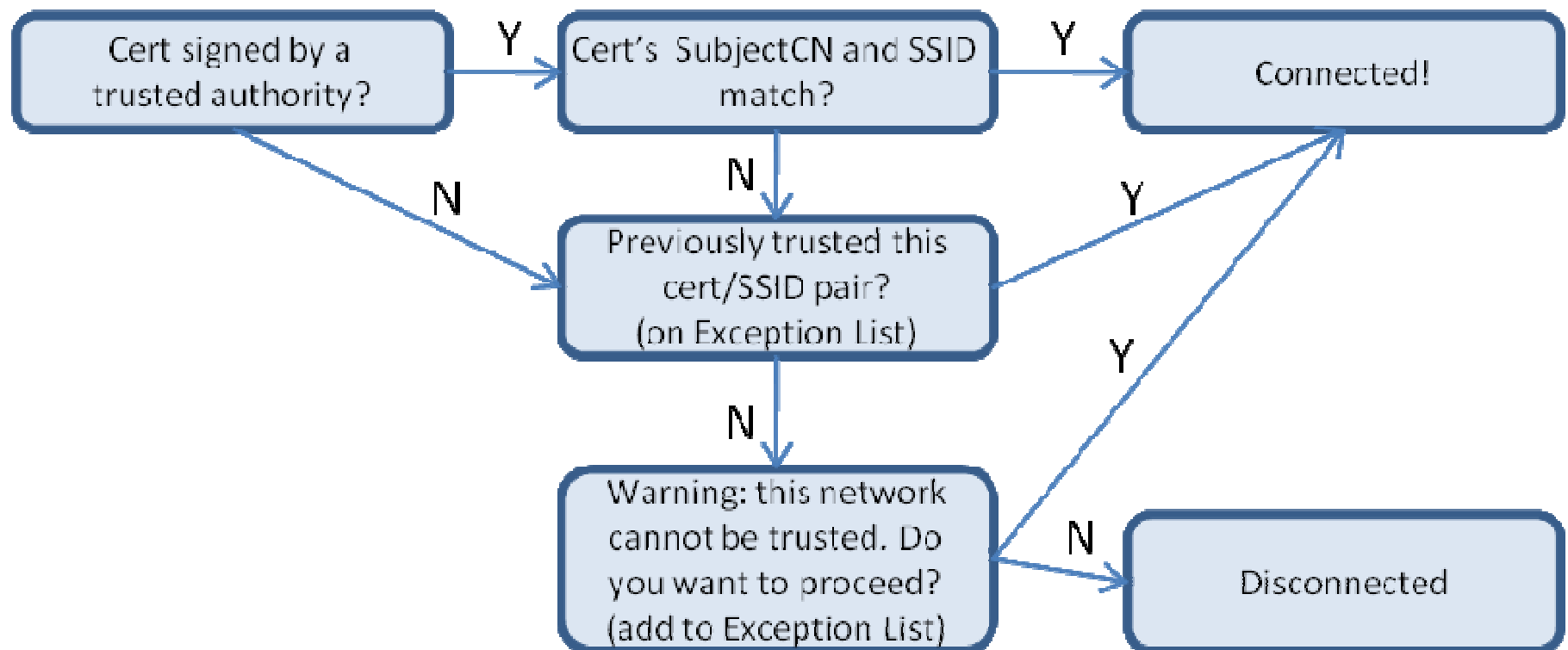
Certificate Authorities

- Exclusive SSID
 - CAs register SSIDs using a global database
 - Different CAs need to avoid issuing the same cert
 - Everyone needs to monitor for misleading names
 - DNS?
 - We already have the certificate infrastructure
 - It may be desirable to differentiate these from SSL Certs
 - » AFAWK, X.509 doesn't currently offer a suitable field
 - Although 802.11 supports “dots” in SSIDs, many APs do not
 - If you want nicer SSIDs you'll need a new DB
 - Whois?
 - More complicated verification process

Certificate Revocation

- Online Certificate Status Protocol
 - Allows a client to query a CA about the status of a certificate
 - But we're not online
- OSCP Stapling
 - Allows a server to cache the CA's timestamped OCSP response and send it to the client
 - Could be transmitted during EAP-TLS conversation

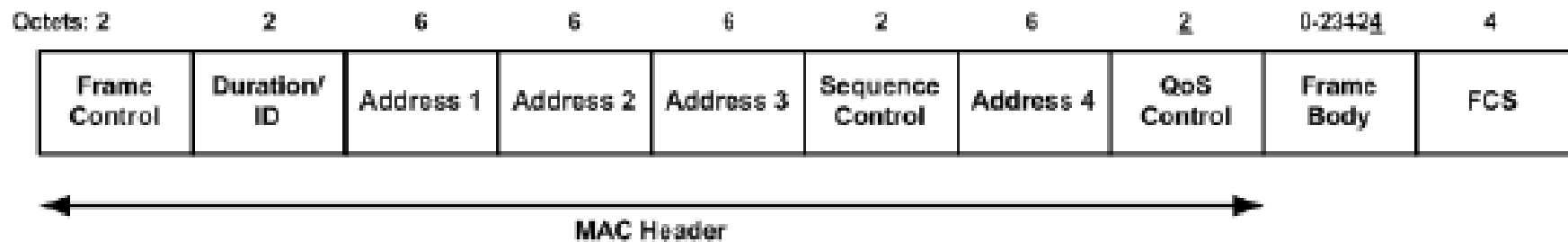
SSH style solution for uncertified APs



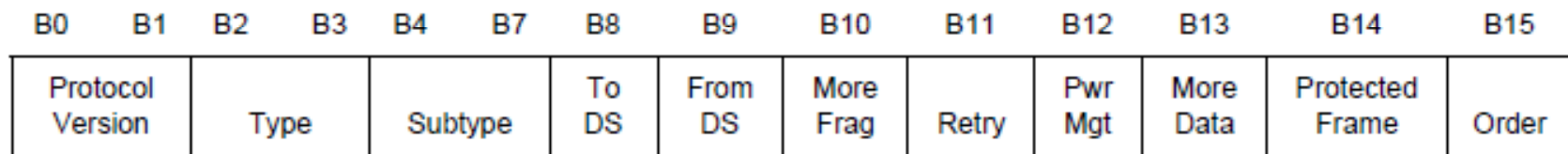
Changes to 802.11

- Users need to be able to differentiate networks that support SOWN from other kinds of secure WiFi networks
 - We suggest using the RSN Capabilities Field
- Current 802.11 SSIDs are not long enough to accommodate domain names
 - We propose a new field called eXtended SSID
- 802.11u may provide some of these capabilities in a different way

802.11 Frames

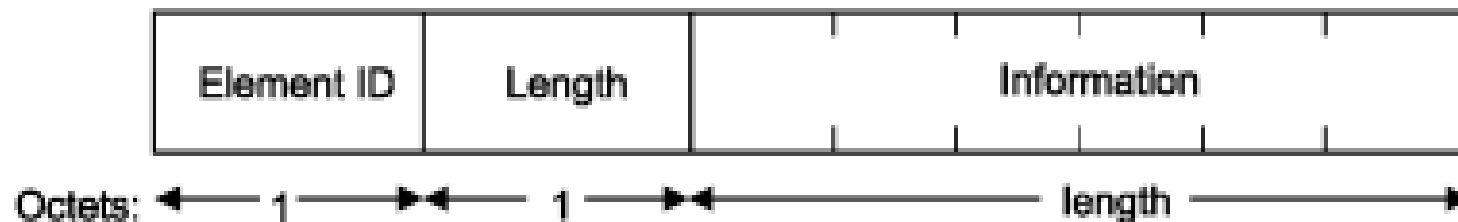


802.11 Frame Header



Bits : 2 2 4 1 1 1 1 1 1 1 1

Frame Control Field in 802.11 Frame Header



Information Element

The RSN Capabilities Field

Element ID	Length	Version	Group Cipher Suite	Pairwise Cipher Suite Count	Pairwise Cipher Suite List	AKM Suite Count	AKM Suite List	RSN Capabilities	PMKID Count	PMKID List	
Octets:	1	1	2	4	2	4-m	2	4-n	2	2	16-s

RSN Information Element

B0	B1	B2	B3	B4	B5	B6	B8	B9	B10	B15
Pre-Auth	No Pairwise	PTKSA Replay Counter	GTKSA Replay Counter	Reserved	PeerKey Enabled	Reserved				

RSN Capabilities Field Format

- New RSN capability bit fields to indicate support for
 1. Anonymous 802.1X Authentication (EAP-TLS)
 2. Certificate validation against network's SSID

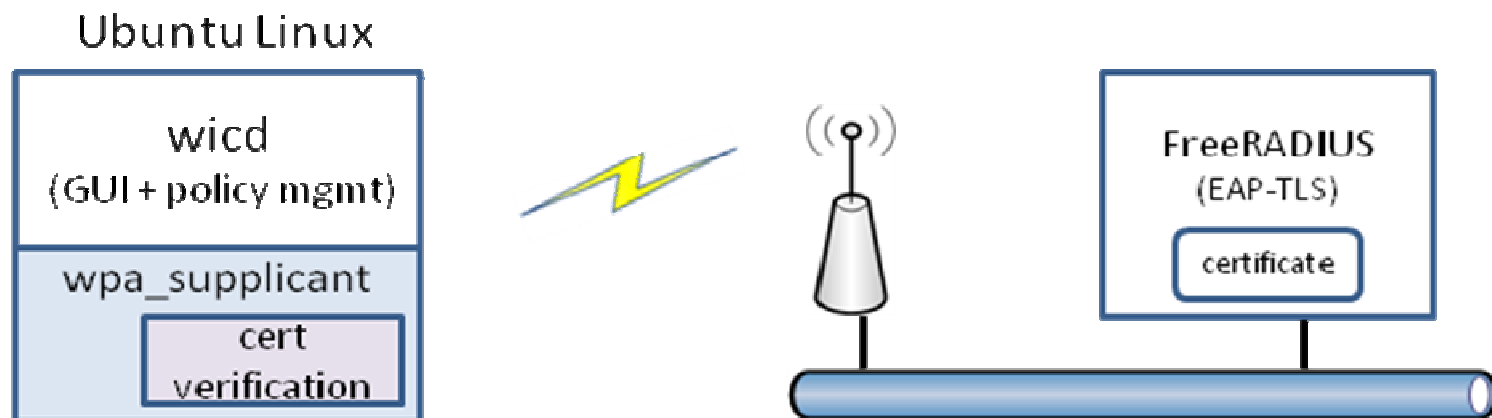
The XSSID



The XSSID Information element allows SSIDs to be as long as a domain name.

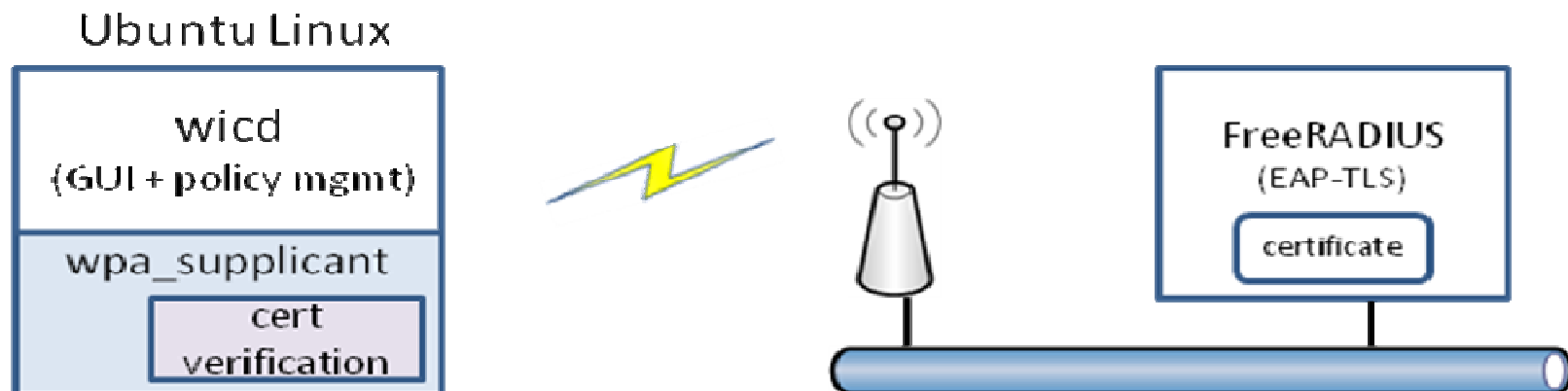
Implementation

- Tak built a prototype by extending the following modules on Ubuntu Linux
 - FreeRADIUS (RADIUS server supporting 802.1x)
 - wpa-supPLICANT (WPA supplicant)
 - wicd (GUI + Policy management tool)



FreeRADIUS

- A very small change made to FreeRADIUS to support anonymous EAP-TLS
 - Modified `rlm_eap_tls.c` to use the `SSL_VERIFY_NONE` flag for the OpenSSL function "`SSL_CTX_set_verify`".
 - This flag prevents the server from sending a `certificate_request` message to the client.



Hostapd

- Christopher's implementation used Hostapd instead of FreeRadiusD
- Simple one *bit* change to hostapd

```
int eap\_server\_tls\_ssl\_init (struct eap\_sm *sm, struct eap\_ssl\_data *data, int verify_peer)
```

```
src/eap_server/eap_tls.c
```

```
68c68
```

```
< if (eap_server_tls_ssl_init(sm, &data->ssl, 0)) {
```

```
---
```

```
> if (eap_server_tls_ssl_init(sm, &data->ssl, 1)) {
```

Supplicant Support

- Windows, Linux supplicants think they need a client certificate
- Just a basic IF statement before connection starts...

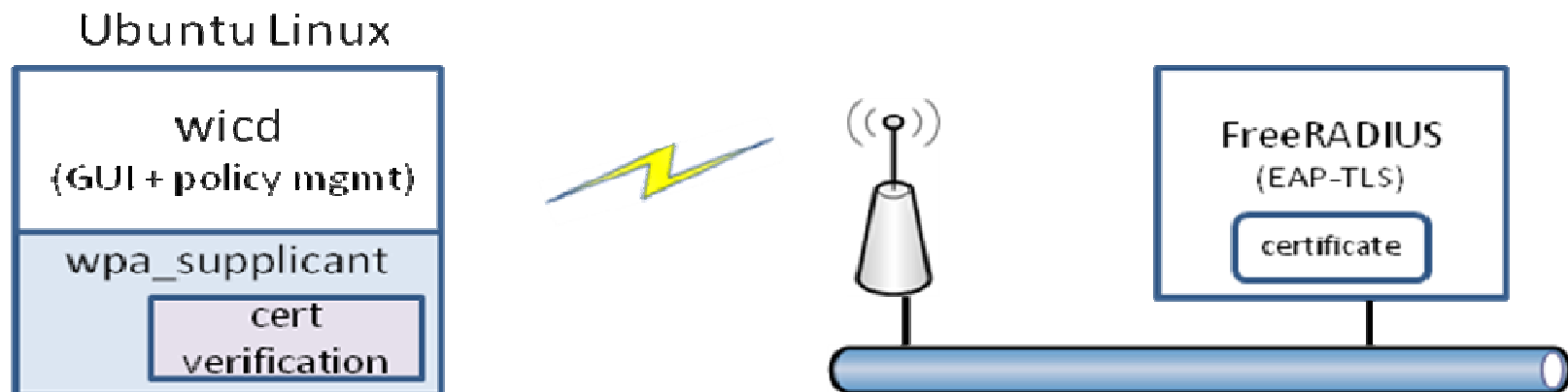
```
wpa_supplicant-0.6.9/src/eap_peer/eap_tls.c
    if (config== NULL ||
        ((sm->init_phase2 ? config->private_key2 : config->private_key)
         == NULL &&
         (sm->init_phase2 ? config->engine2 : config->engine) == 0)) {
        wpa_printf(MSG_INFO, "EAP-TLS: Private key not configured");
        return NULL;
    }
```

Supplicant Support

- Configuring supplicants with ANY certificate satisfies this requirement
 - It doesn't have to be a valid cert
 - They'll never be asked for it anyway
- Changing wpa_supplicant to remove the if statement removes the cert requirement

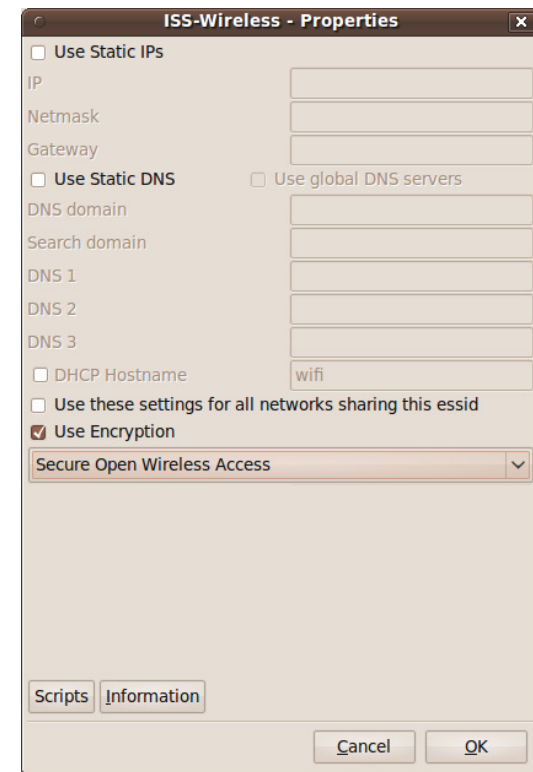
wpa-supPLICant

- We registered a callback function for certificate verification in wpa-supPLICant's EAP-TLS component.
 - Verifies that the SSID of the access point matches the Common Name in the digital certificate.



Implementation - future work

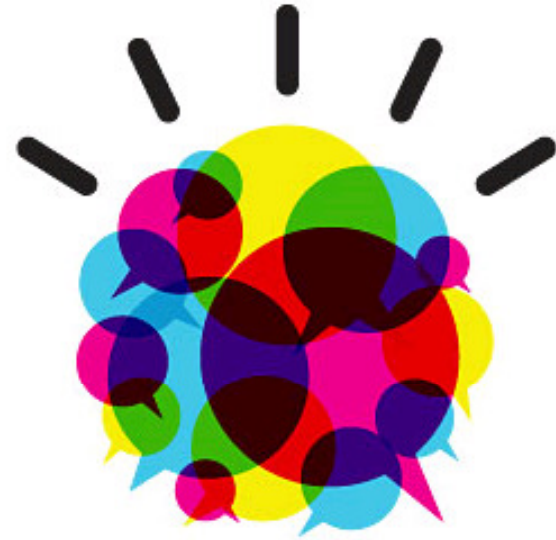
- 802.11 extension (Extended SSID (XSSID) and RSN)
- FreeRADIUS policy management for Secure Open Wireless Access
- Better GUI
 - A lock icon next to “ISS-Wireless” represents Secure Open Wireless Access



Code Release

- FreeRADIUS, wpa-supPLICANT, and wcid are all released under the GPLv2.
- We will release the changes we made to this code under the GPLv2.
- Should be available soon at <http://blogs.iss.net>
- Please download and experiment with it.

The result: A smarter wireless Internet



Contact us

Christopher Byrd: chris@riosec.com

Tom Cross: tcross@us.ibm.com

Takehiro Takahashi: takehiro.takahashi@gmail.com