School of Computing
FACULTY OF ENGINEERING & PHYSICAL SCIENCES

UNIVERSITY OF LEEDS

COMP3911 Secure Computing

10: DNS, ARP & Application Protocols

Nick Efford

https://comp3911.info

Last Time

- We discussed the role played by firewalls and highlighted some of the limitations of this technology
- We discussed how TLS slots in between the application and transport layers, providing support for confidentiality and checking of authenticity & integrity
- We noted the challenges of dealing with complex protocols and discussed some real-world examples of TLS implementation and configuration errors
- We saw how IPSec provides similar benefits to TLS but more transparently, in the Internet layer

Objectives

- To consider how supplementary protocols dealing with name/address resolution can be attacked
- To consider insecurities in selected protocols of the application layer (and some solutions)
    - Remote access
    - Email

Address Resolution

Happens at two levels:

- **DNS** resolves human-readable **domain names** into the numeric IP addresses needed for routing
- **ARP** (Address Resolution Protocol) resolves IP addresses into the 48-bit **MAC addresses** that identify specific network interfaces of hardware

Spoofing attacks are
possible with both of these!

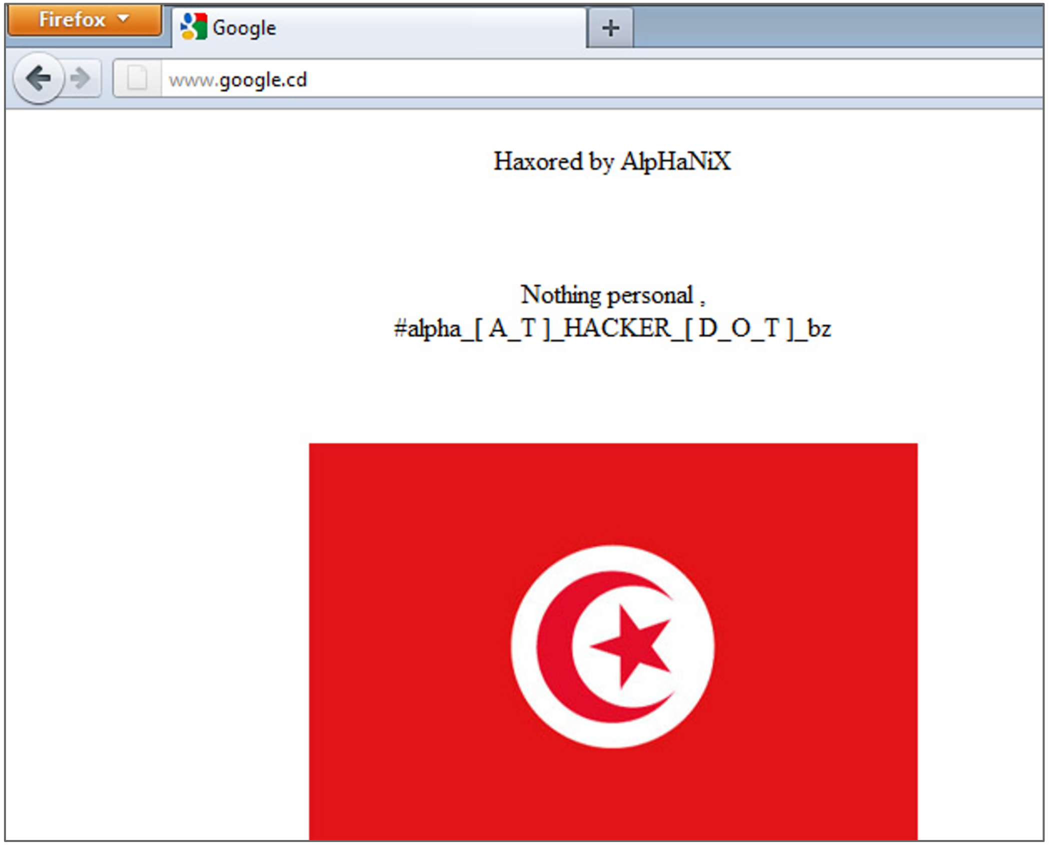
Attacking DNS

UNIVERSITY OF LEEDS

- Directly, via **cache poisoning**
- Indirectly, by hijacking site’s DNS records via their registrar

Example:

Cache poisoning affected Google’s domains in the Democratic Republic of Congo in December 2011



Haxored by AlpHaNiX

Nothing personal ,
#alpha_[A_T]_HACKER_[D_O_T]_bz

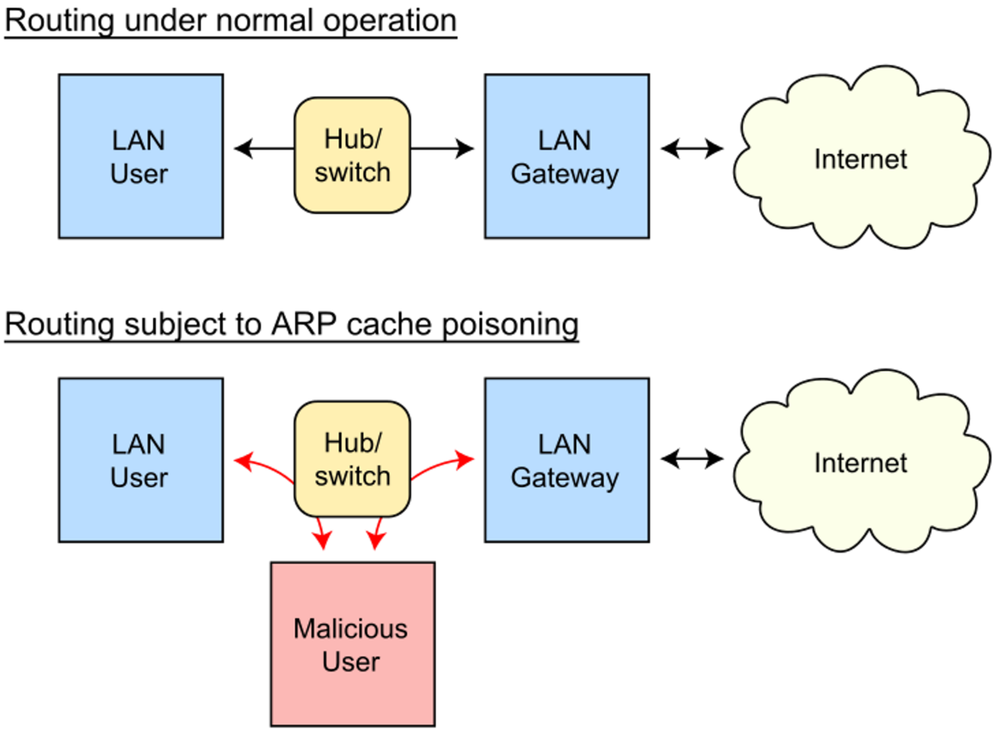
DNS Cache Poisoning

- Attacker can race to respond first to a DNS query, but they have to guess the correct 16-bit transaction ID
- If attacker gets it wrong, correct query result will end up being cached for a while, preventing further attacks
- … but attacker can fire off **multiple replies** with different IDs, all beating the real name server's response
- … and can also make requests for **multiple related domains** (1.foo.com, 2.foo.com, etc)

(For the rest of the details, see Dan Kaminsky's BlackHat 2008 presentation)

ARP Cache Poisoning

- Same basic idea as DNS cache poisoning
- Hijacks the process of resolving an IP address to a MAC address
- Like DNS, this can be used as a 'man-in-the-middle' attack…



Remote Access: The Bad Old Days

- Remote logins using `telnet` or `rlogin`
- `rcp` to copy files to/from remote machine, `rsh` to execute single commands on remote machine
- **All traffic is unencrypted!**
- `telnet` always requires username & password
- ‘r’ tools recognise **trusted hosts**, avoiding the need for a password, but these hosts can be spoofed

An Aside: ‘Banner Grabbing’

- Deliberate use of `telnet` with non-`telnet` ports
- Common services running on those ports will sometimes respond with **banners**, leaking information about software vendor, versions, etc
- Threat: **information disclosure**

```
$ telnet 127.0.0.1 22
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
SSH-1.99-OpenSSH_3.6.1p1+CAN-2004-0175
asdf
Protocol mismatch.
Connection closed by foreign host.
```

Sniffing a `telnet` Session

```
$ dsniff -m
dsniff: listening on eth0
-----------------
18/02/14 20:11:07 tcp client.example.com.1112
                   -> server.example.com.23 (telnet)
matthew
LeedsUtd
ls -l
logout
```

ARP spoofing can be used to ensure that sniffer gets to see traffic intended for other machines…

Fixing it: Secure Shell (SSH)

- Cryptographically-enabled protocol for remote login, file transfer and TCP connection tunnelling
  - Replaces ‘r’ tools with `slogin`, `ssh`, `scp`…
- Traffic is encrypted with a symmetric cipher, negotiated between client and server
- Public key cryptography is used for client authentication and symmetric key exchange

Note: SSH is not the same thing as SSL!

How Does It Work?

- Every host has its own **SSH key pair**
- When client connects, server provides its public key
  - Client can confirm identity of server by checking against a local database of stored public keys or (more rarely) by verifying a certificate
  - Client can use it to encrypt auth data sent to server
- Multiple approaches to client authentication:
  - You provide valid password for remote machine
  - You create a key pair and put public key on remote machine; SSH allows a login if you have corresponding private key

SSH Problems

UNIVERSITY OF LEEDS

- Complex, custom-designed protocol
- SSH-1 had design flaws and is now considered insecure
- SSH-2 is superior but has had its own issues
  - 2008 ‘CBC vulnerability’ http://www.kb.cert.org/vuls/id/958563
  - Attacker could recover 14 bits of plaintext from arbitrary blocks of ciphertext, with a probability of 2^{-14}
  - Fixed by using CTR mode instead of CBC

‘Man In The Middle’ Attacks

- Attacker could steal credentials if they could impersonate a server – and you might not notice if they forwarded your connection attempt on to real server!
- SSH countermeasure: maintain a local store of trusted public keys for previously-accessed servers
  - Users asked whether they trust a server the first time they connect; public key added if they say Yes
- Clearly still a risk, but ‘window of opportunity’ is smaller

Sending Email: SMTP

- A simple request-response protocol, using port 25
- No authentication (⇒ easy spoofing of sender)
- All traffic sent as ASCII plaintext
- Banner leaks name / vendor / version of server software
- EXPN and VRFY commands leak user details
- ‘Open relays’ are heavily abused (spam)

Receiving Email: POP3 & IMAP

POP3

- Username and password normally sent as plaintext
- APOP extension: server sends a timestamp and client to sends back MD5 hash of this and a shared secret
- Mail itself is unprotected

IMAP

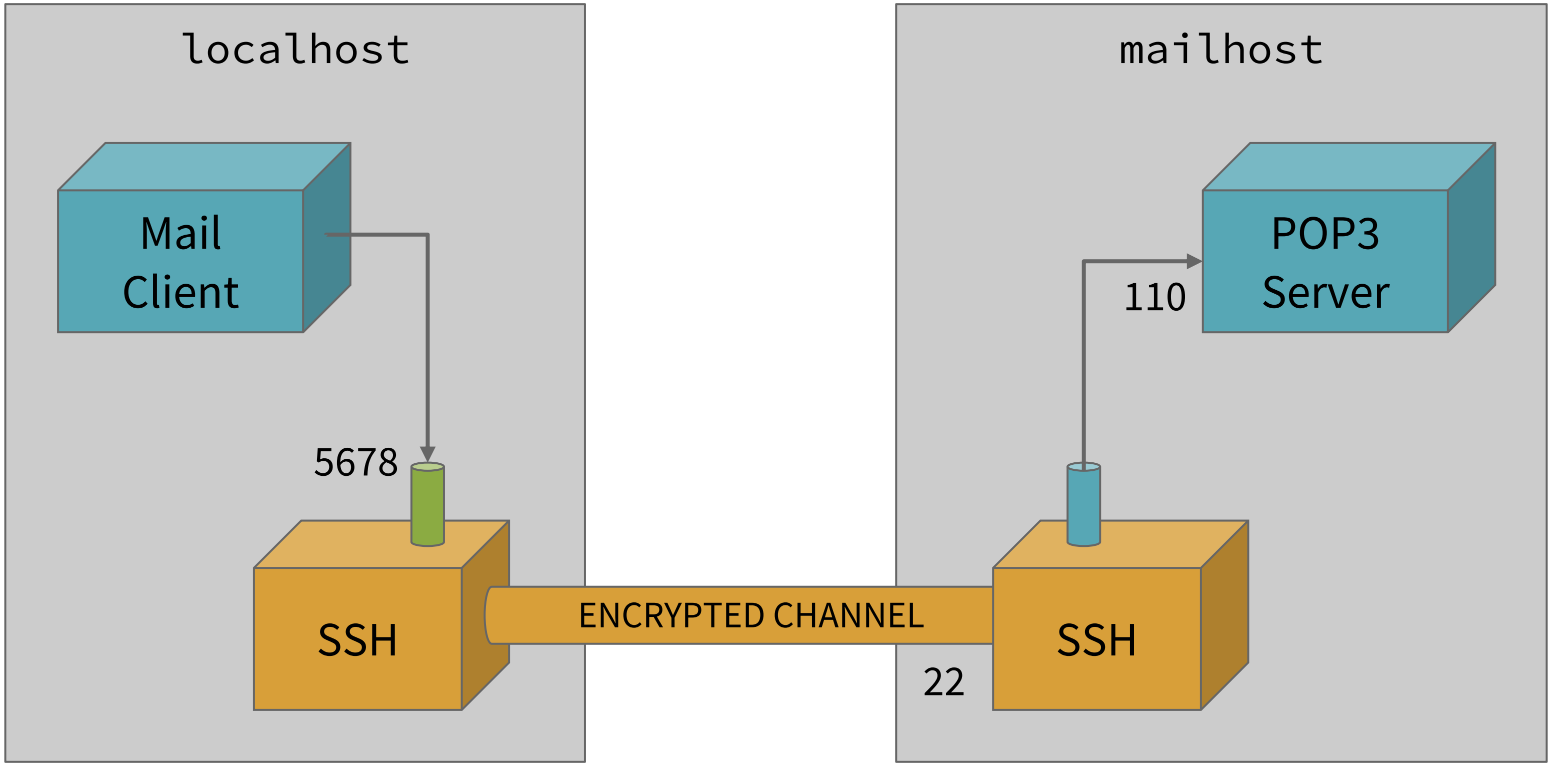
- Similar auth options to POP3 (equally weak)
- Mail itself is also unprotected

Securing Email Protocols

- SSH provides one solution, since it allows us to **tunnel** insecure protocols over an encrypted channel
- … but protocols themselves have all been extended to add support for TLS, which is the preferred option

Tunnelling

UNIVERSITY OF LEEDS



```
ssh -f -N -L 5678:localhost:110 mailhost
```

Better Approaches

- Alternate ports for TLS-based services
  - Equivalent to the approach used for HTTP
  - SMTPS on port 587, instead of 25
  - POP3S on port 995, instead of 110
  - IMAPS on port 993, instead of 143
- ‘Opportunistic TLS’
  - Continue using standard ports but add a ‘STARTTLS’ command to the protocols
  - Issuing the command triggers TLS handshake and upgrading to a secure connection

STRIPTLS Attack

- MITM attack affecting two major ISPs in Thailand in 2014
- Transparent proxy intercepted SMTP connections and removed STARTTLS command, then issued a ‘TLS unavailable’ response
- Client would fall back on authenticating over an unencrypted channel
- … proxy would thus capture usernames and passwords!

Trade-offs

- Desirable to protect email using ‘end-to-end encryption’
- But email is a common vector for **malware**
- Scanning for malware at mail gateways isn’t feasible if content is encrypted…

Summary

UNIVERSITY OF LEEDS

We have

- Seen that DNS and ARP are vulnerable to spoofing
- Noted that traditional Unix tools for remote login, remote command execution and file transfer are very insecure, and that **SSH** provides secure drop-in replacements
- Observed that email protocols also fail to adequately protect user credentials or message content
- Learned that email can be protected in transit via **SSH tunnelling**, full TLS or **opportunistic TLS**

Follow-Up / Further Reading

UNIVERSITY OF LEEDS

- “It’s the end of the cache as we know it”: Dan Kaminsky’s BlackHat 2008 presentation (PDF)
- Short video on consequences of DNS cache poisoning
- Statistics on DNSSEC adoption
- Alternative approach: DNS over HTTPS
- OpenSSH
- STARTTLS Everywhere: mail domain evaluation tool