

Attacks on TLS and SASL clients

Michael Samuel

Protocols affected

StartTLS and SASL

- XMPP (used in the examples)
- IMAP
- SMTP
- LDAP

Related:

- Web (see Moxie Marlinspike's work)
- SIP

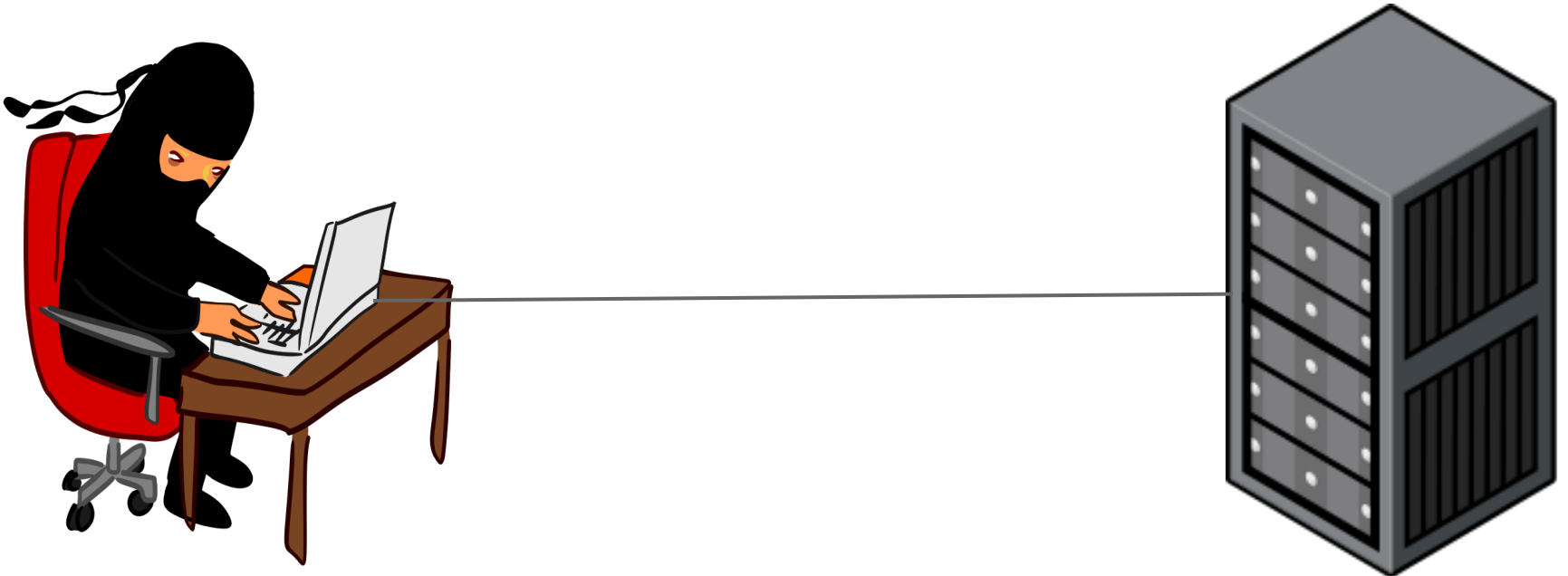
Why target clients?

They're an easy target!

- Features and compatibility sells - error messages don't
- Servers don't normally send the password over the network, clients need to sometimes
- Useful error dialogs are annoying to code
- Race to market with "business" and "enterprise" IM clients - aka "Unified Communication"

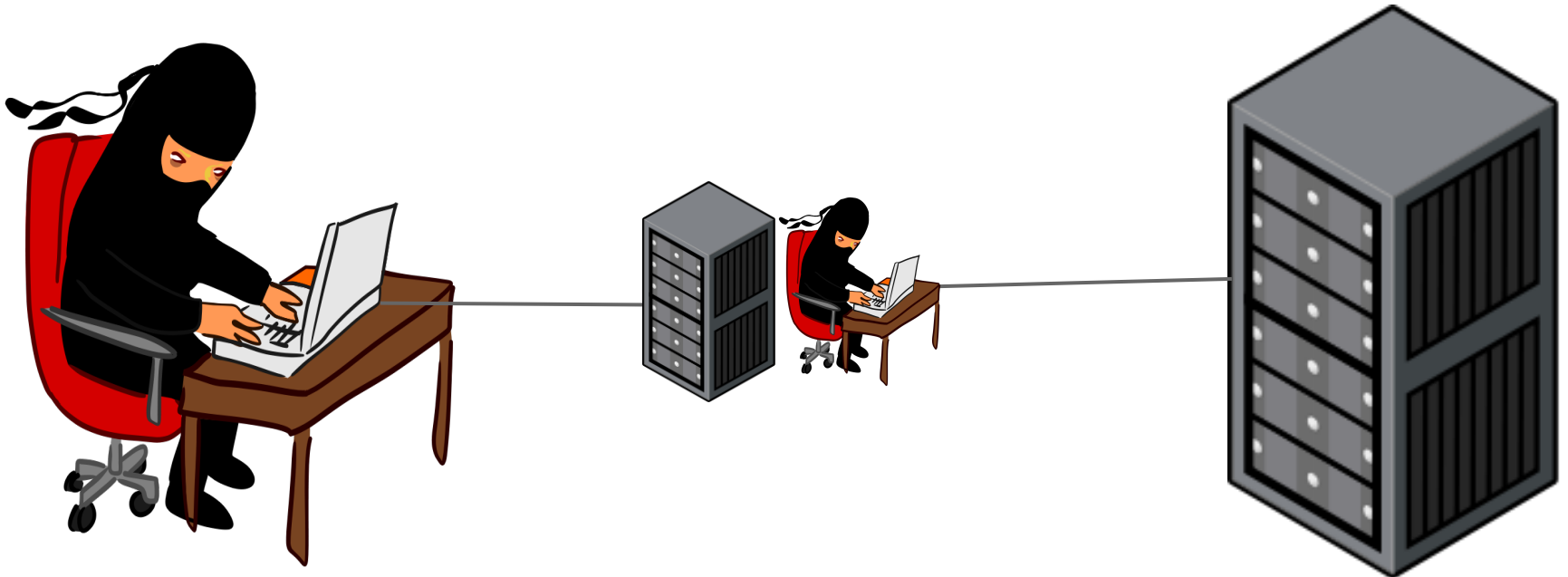
Man in the middle attack

TLS and SASL are used in client-server protocols



Man in the middle attack

1. Intercept client connection and answer like a server.
2. Connect to the real server, using information that the client sends you to login
3. Insert, delete, replace or just log data as needed



MITM attack - Linux Quickstart

1. `iptables -t nat -A PREROUTING -p tcp --dport 5222 -j REDIRECT --to-port 5002`
2. Run your client program, listening on 5002
3. Route the traffic *through* your linux box using arpspoof, dhcp spoofing or maybe it already does.

To get the original IP address in your proxy code (C/C++):

```
getsockopt(s, SOL_IP, SO_ORIGINALDEST, &addr, &addrlen);
```

or Python (IPv4 only, sorry!):

```
packedDest = s.getsockopt(socket.SOL_IP, 80, 16)
(destPort, ) = struct.unpack(">H", packedDest[2:4])
destHost = socket.inet_ntoa(packedDest[4:8])
```

MITM attack - Quickstart

1. Proxy connection until you see the STARTTLS handshake - this might not happen
2. Fire up your SSL libraries and start talking TLS to both the client (as a server) and the server (as a client).
3. Proxy the decrypted traffic between the two SSL sockets

Without a copy of the server's private key this will fail... right?

Before you can talk privately...

You need to make sure you're talking to the right person!



Certificates are how SSL solves this problem

Before you can talk privately...

- **SSL/TLS isn't secure if the certificates aren't validated**
- People don't like paying for certs
 - The CAs *are* gouging you
 - They take time - new servers on a VM cluster can be built in minutes
 - You have to ask for budget (or the boss's credit card)
- Rant about CA system's weird incentives, strange notions of trust and the huge CA list will have to wait for the pub

Unchecked certificates

☐ Ignore SSL certificate errors

- This option belongs under "debugging" at best
- Many apps never check certificates - this should be reported as a security bug
- Even more apps offer this as the only option for self-signed certs. Don't use these apps if you use self-signed certs. (Some do it silently!)
- Click-accept for cert errors is also way too common
- If you see an obvious error with the server cert (expired, wrong CN, etc) then you know that all clients connecting to it have ticked this box

If some clients are showing error messages and others aren't, which app will the helpdesk think is broken?

Cloning a chain

When looking at a "View Certificate" dialog, what do you check for?

The only thing that matters in an ***unverified*** certificate are the fingerprint and public key - if you can find something to verify them against.

If the attacker created the root CA, they control everything in the certificate.

- Have you ever heard anyone reading out a fingerprint?
- Who would they even talk to?
- How do I get my mail if the fingerprint doesn't match?!?

Cloning a chain

Root CA is basically a self-signed certificate

issuer= /L=ValiCert Validation Network/O=ValiCert, Inc./OU=ValiCert Class 2 Policy Validation Authority/CN=http://www.valicert.com/emailAddress=info@valicert.com

subject= /L=ValiCert Validation Network/O=ValiCert, Inc./OU=ValiCert Class 2 Policy Validation Authority/CN=http://www.valicert.com/emailAddress=info@valicert.com

Intermediate CAs and Certificate are signed by parent CA

issuer= /L=ValiCert Validation Network/O=ValiCert, Inc./OU=ValiCert Class 2 Policy Validation Authority/CN=http://www.valicert.com/emailAddress=info@valicert.com

subject= /C=US/O=The Go Daddy Group, Inc./OU=Go Daddy Class 2 Certification Authority

issuer= /C=US/O=The Go Daddy Group, Inc./OU=Go Daddy Class 2 Certification Authority

subject= /C=US/ST=Arizona/L=Scottsdale/O=GoDaddy.com, Inc./OU=http://certificates.godaddy.com/repository/CN=Go Daddy Secure Certification Authority/serialNumber=07969287

issuer= /C=US/ST=Arizona/L=Scottsdale/O=GoDaddy.com, Inc./OU=http://certificates.godaddy.com/repository/CN=Go Daddy Secure Certification Authority/serialNumber=07969287

subject= /O=*.miknet.net/OU=Domain Control Validated/CN=*.miknet.net

You can pre-create a few keypairs and generate certificate chains on the fly in your MITM proxy.

This will still fail to verify - but is way less likely to raise alarm bells than a self-signed cert or one signed by the "wrong" CA.

Google encouraging bad behaviour

```
$ host -t srv _xmpp-client._tcp.miknet.net
```

```
_xmpp-client._tcp.miknet.net has SRV record 20 0 5222 talk4.l.google.com.
```

```
_xmpp-client._tcp.miknet.net has SRV record 5 0 5222 talk.l.google.com.
```

```
_xmpp-client._tcp.miknet.net has SRV record 20 0 5222 talk1.l.google.com.
```

```
_xmpp-client._tcp.miknet.net has SRV record 20 0 5222 talk2.l.google.com.
```

```
_xmpp-client._tcp.miknet.net has SRV record 20 0 5222 talk3.l.google.com.
```

Google presents this certificate:

```
issuer= /C=US/O=Google Inc/CN=Google Internet Authority
```

```
subject= /C=US/ST=California/L=Mountain View/O=Google Inc/CN=talk.google.com
```

The CN doesn't match!

Have you ever asked why you're not getting a cert error when you login to your Google Apps domain?

SRV records for fun and profit

`_xmpp-client._tcp.miknet.net IN SRV 5 0 5222 xmpp.miknet.net.`

If I present a valid certificate with a CN of `xmpp.miknet.net` should it be trusted?

SRV records for fun and profit

`_xmpp-client._tcp.miknet.net IN SRV 5 0 5222 xmpp.miknet.net.`

If I present a valid certificate with a CN of `xmpp.miknet.net` should it be trusted?

What if the DNS response was:

`_xmpp-client._tcp.miknet.net IN SRV 5 0 5222 evil.attacker.org.`

Certificates must match the name that the user requested

This will be a common problem in SIP stacks when they finally start using TLS
Polycom phones have an option to enable this vulnerability (disabled by default).

Why bother with SSL Attacks?

```
<?xml version='1.0' ?>
<stream:stream to='jabber.org' xmlns='jabber:client'
xmlns:stream='http://etherx.jabber.org/streams'
version='1.0'>
<?xml version='1.0'?>
<stream:stream xmlns='jabber:client' xmlns:stream='http:
//etherx.jabber.org/streams' from='jabber.org'
id='5ce74cfce8e91fc4' version='1.0'>
  <stream:features>
    <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls' />
    <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
      <mechanism>DIGEST-MD5</mechanism>
      <mechanism>PLAIN</mechanism>
    </mechanisms>
  </stream:features>
  <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls' />
  <proceed xmlns='urn:ietf:params:xml:ns:xmpp-tls' />
```


Why bother with SSL Attacks?

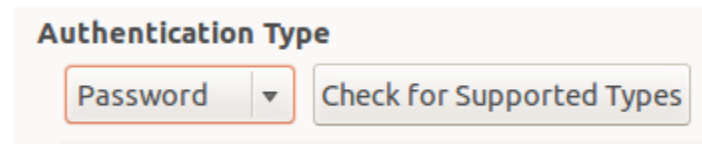
```
<?xml version='1.0' ?>
<stream:stream to='jabber.org' xmlns='jabber:client'
xmlns:stream='http://etherx.jabber.org/streams'
version='1.0'>
<?xml version='1.0'?>
<stream:stream xmlns='jabber:client' xmlns:stream='http:
//etherx.jabber.org/streams' from='jabber.org'
id='5ce74cfce8e91fc4' version='1.0'>
  <stream:features>
    <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
    <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
      <mechanism>DIGEST-MD5</mechanism>
      <mechanism>PLAIN</mechanism>
    </mechanisms>
  </stream:features>
```

While we're at it

```
<?xml version='1.0' ?>
<stream:stream to='jabber.org' xmlns='jabber:client'
xmlns:stream='http://etherx.jabber.org/streams'
version='1.0'>
<?xml version='1.0'?>
<stream:stream xmlns='jabber:client' xmlns:stream='http:
//etherx.jabber.org/streams' from='jabber.org'
id='5ce74cfce8e91fc4' version='1.0'>
  <stream:features>
    <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
    <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
      <mechanism>DIGEST-MD5</mechanism>
      <mechanism>PLAIN</mechanism>
    </mechanisms>
  </stream:features>
```

SASL feature advertisement

Evolution has the right idea:

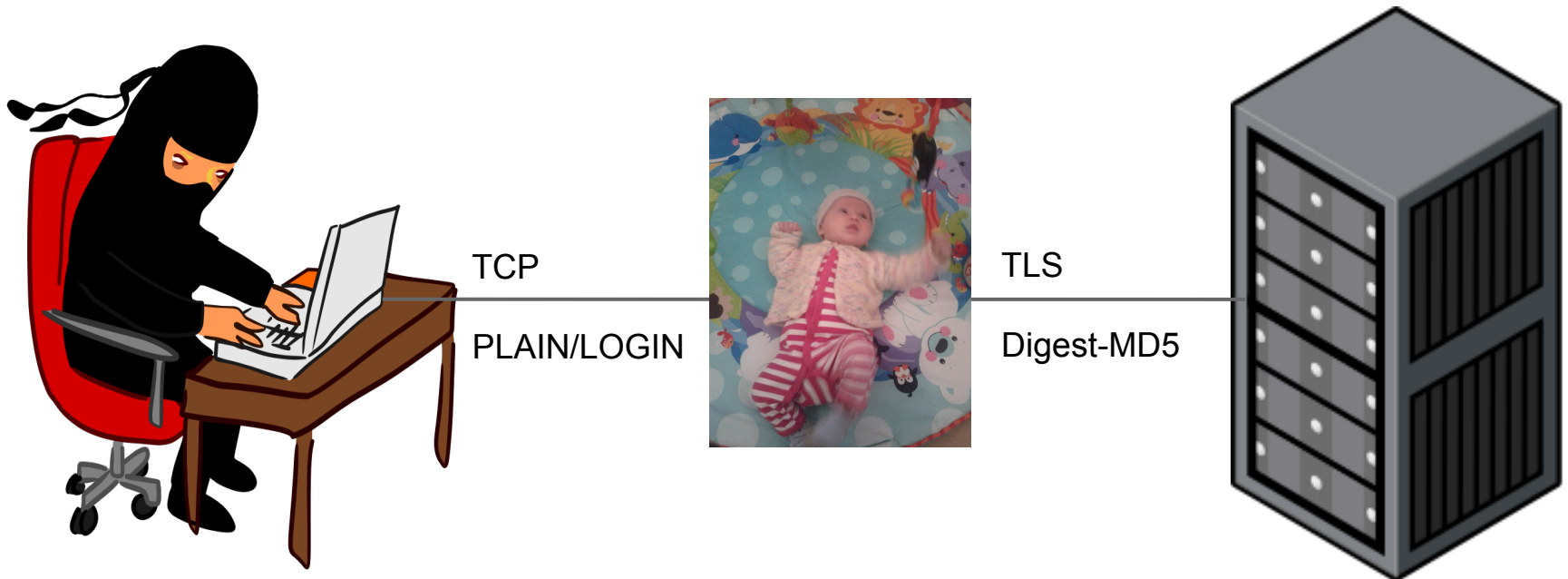


The image shows a user interface element titled "Authentication Type". It features a dropdown menu with "Password" selected, indicated by a small downward arrow. To the right of the dropdown is a button labeled "Check for Supported Types".

- Negotiating security over the network is not a good idea
- Discovering security settings at account setup is a good compromise
- This is also where you should cache the SSL certificate if it's self-signed
- Why would the user all of a sudden want to allow "PLAIN authentication over an unencrypted connection" this morning?

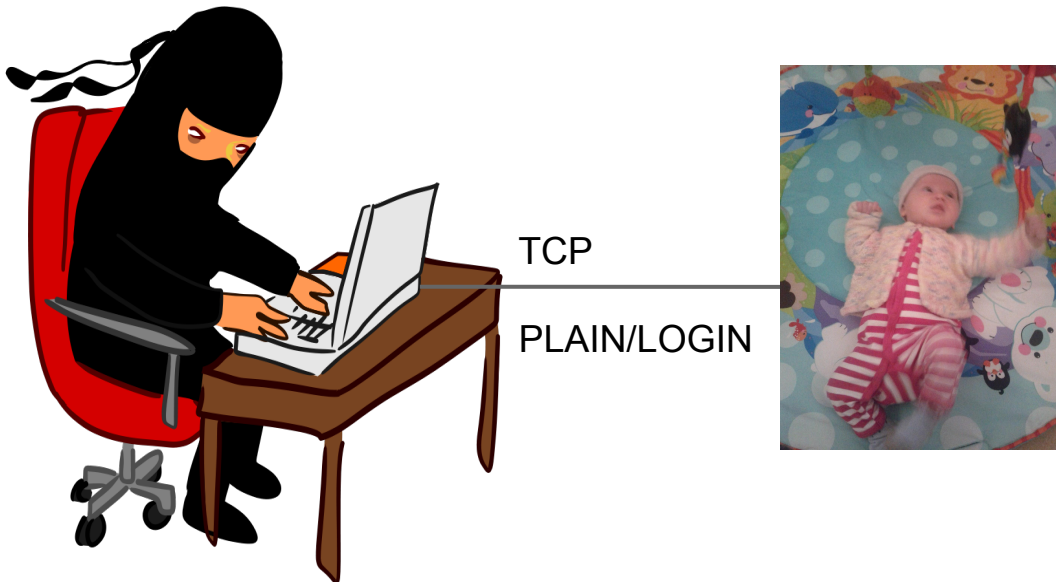
Paranoid Server Settings...

... are easily bypassed



Paranoid Server Settings...

... or ignored



^D

Grab a copy of these slides at:

<http://www.miknet.net/>