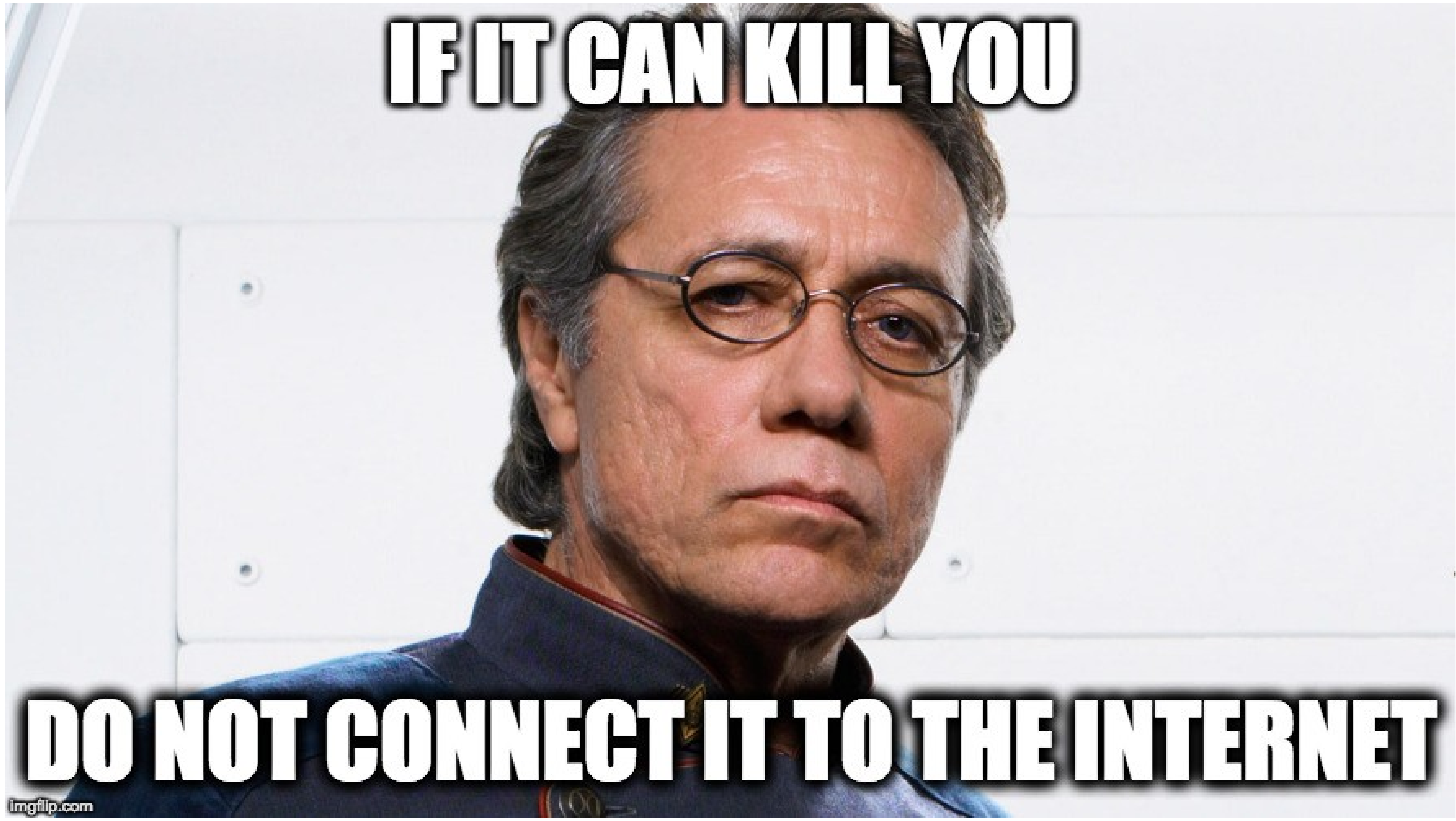
Network Security 5

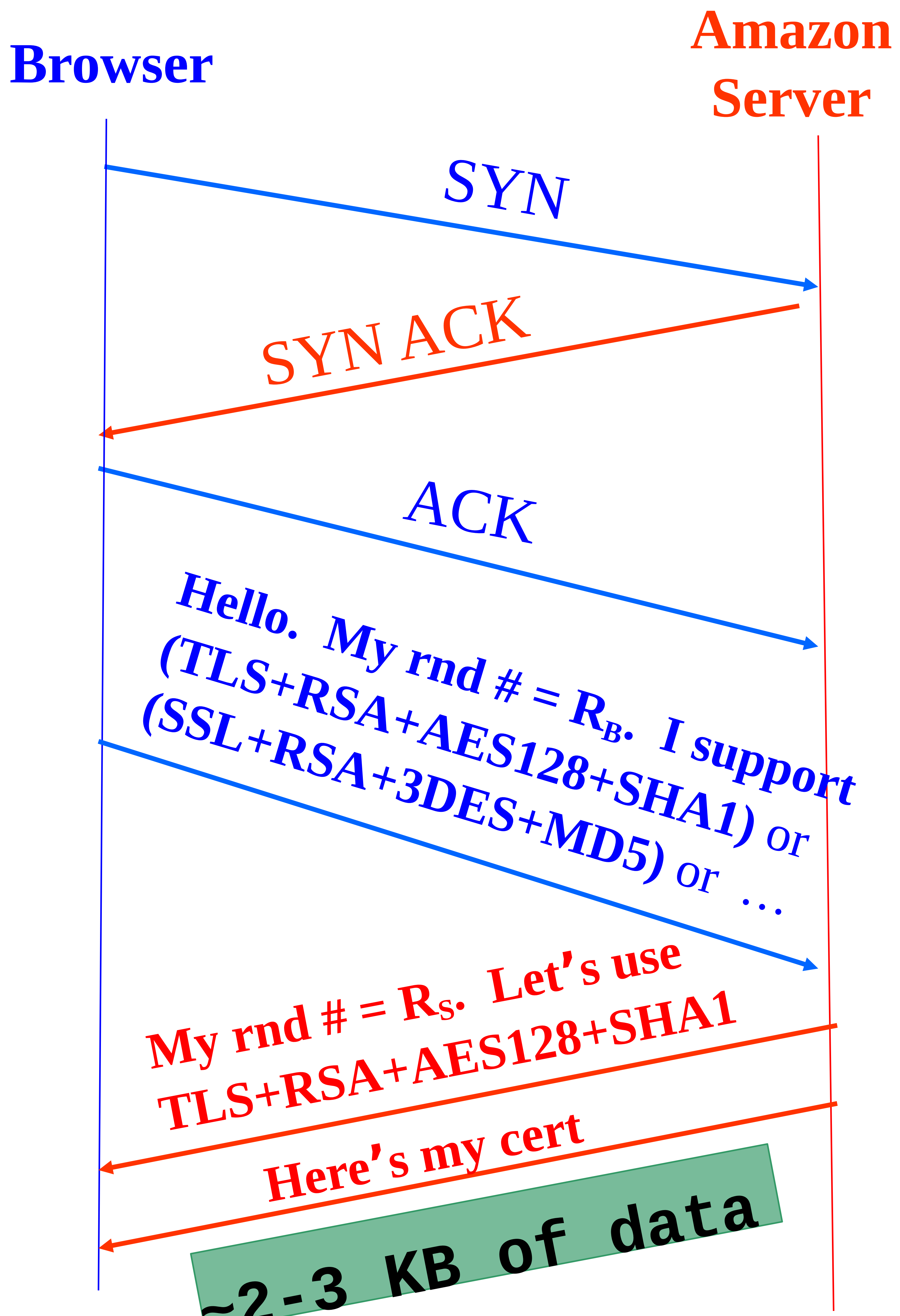


Announcements

- Project 2 due July 29
  - Start your implementation early!
  - Autograder is up
- Complete Mid-Summer Survey
- HW2 due August 1

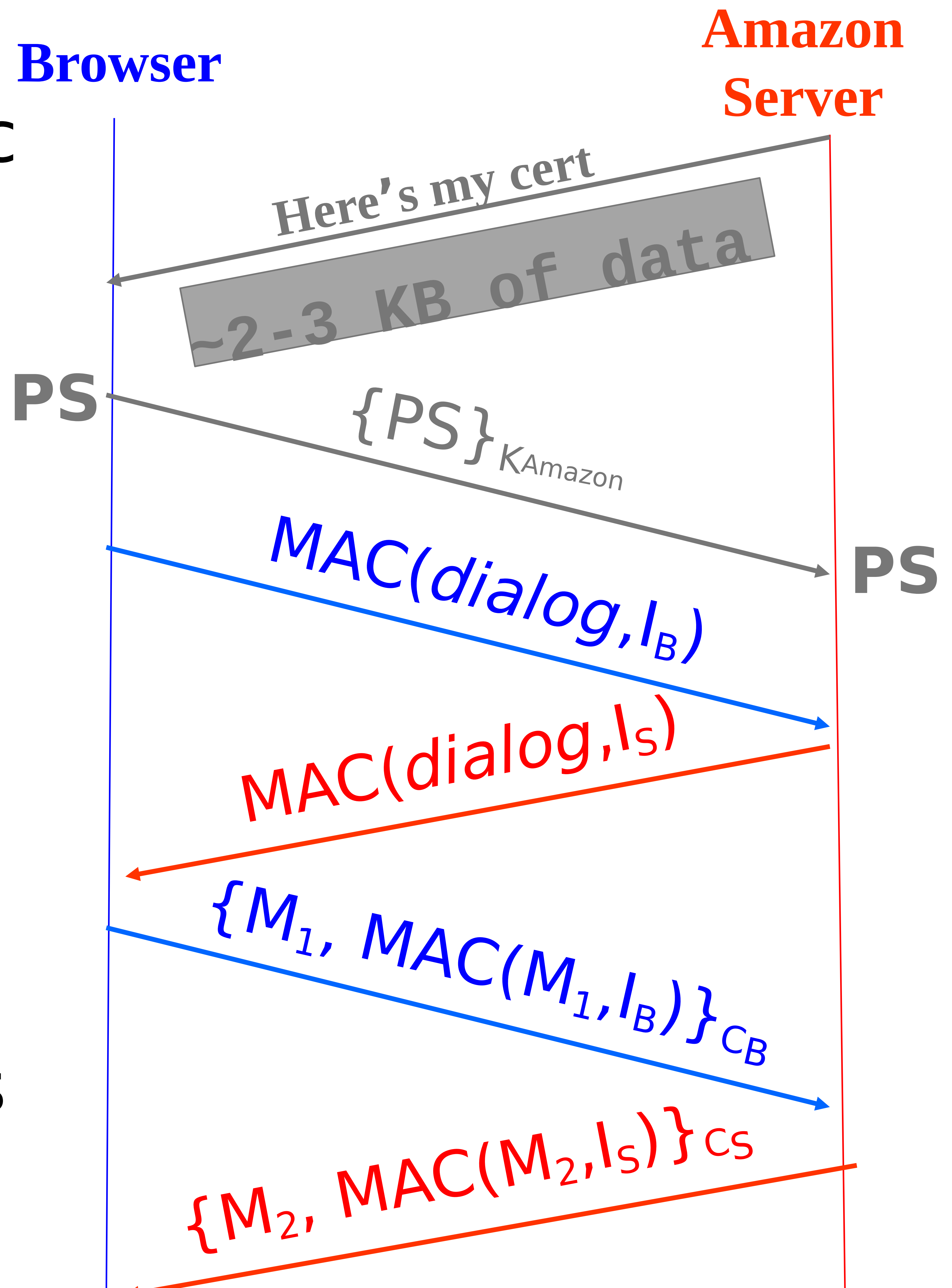
Reminder: HTTPS Connection (SSL / TLS)

- Browser (client) connects via TCP to Amazon's HTTPS server
- Client picks 256-bit random number R_B, sends over list of crypto protocols it supports
- Server picks 256-bit random number R_S, selects protocols to use for this session
- Server sends over its certificate
  - (all of this is in the clear)
- Client now ***validates*** cert


Browser
Amazon Server
SYN
SYN ACK
ACK
Hello. My rnd # = R_B. I support (TLS+RSA+AES128+SHA1) or (SSL+RSA+3DES+MD5) or …
My rnd # = R_S. Let's use TLS+RSA+AES128+SHA1
Here's my cert
~2-3 KB of data

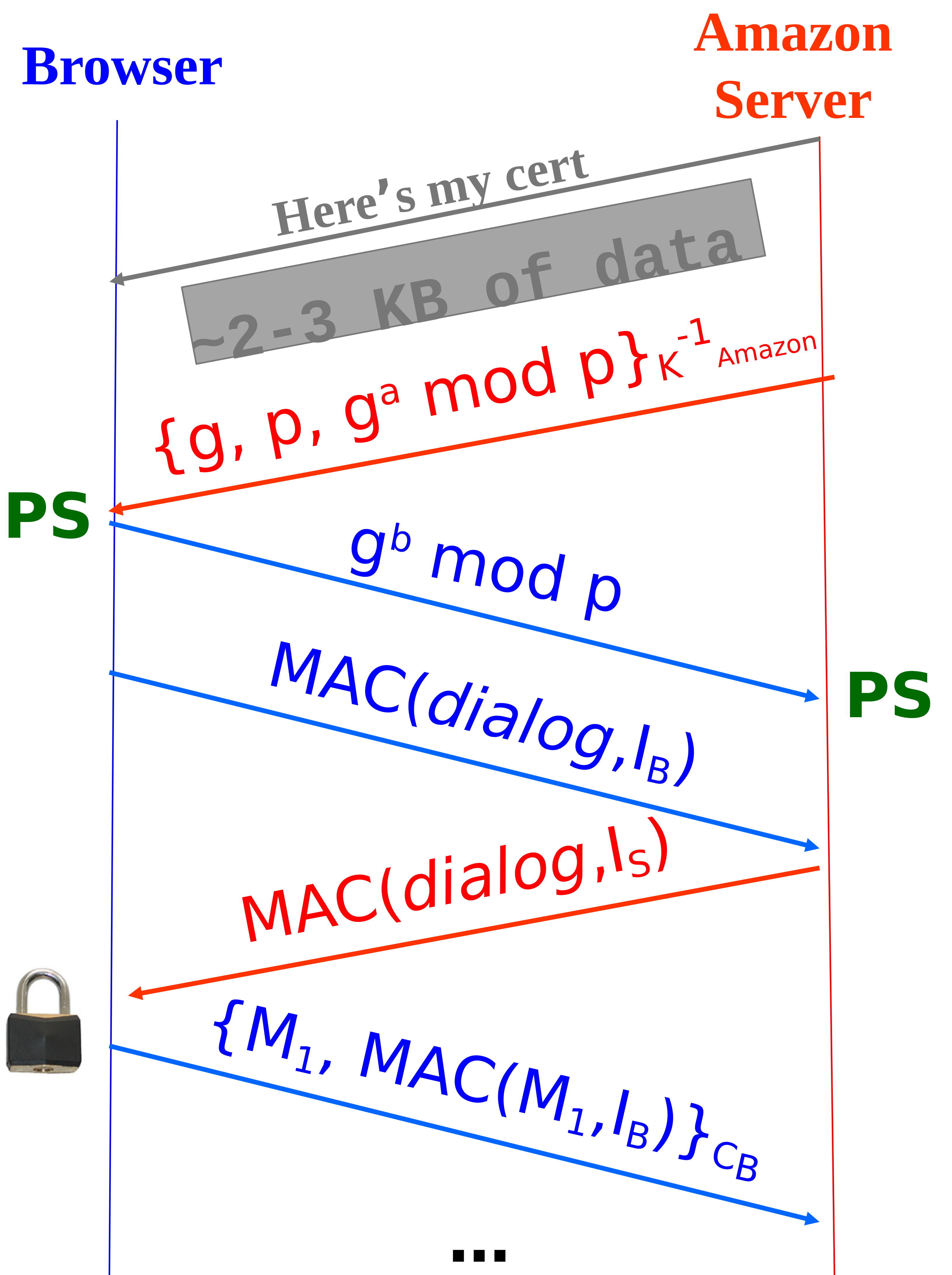
HTTPS Connection (SSL / TLS), cont.

- For RSA, browser constructs "Premaster Secret" PS
- Browser sends PS encrypted using Amazon's public RSA key KAmazon
- Using PS, R_B, and R_S, browser & server derive symm. cipher keys (C_B, C_S) & MAC integrity keys (I_B, I_S)
  - One pair to use in each direction
- Browser & server exchange MACs computed over entire dialog so far
- If good MAC, Browser displa🔒
- All subsequent communication encrypted w/ symmetric cipher (e.g., AES128) cipher keys, MACs
  - Sequence #'s thwart replay attacks



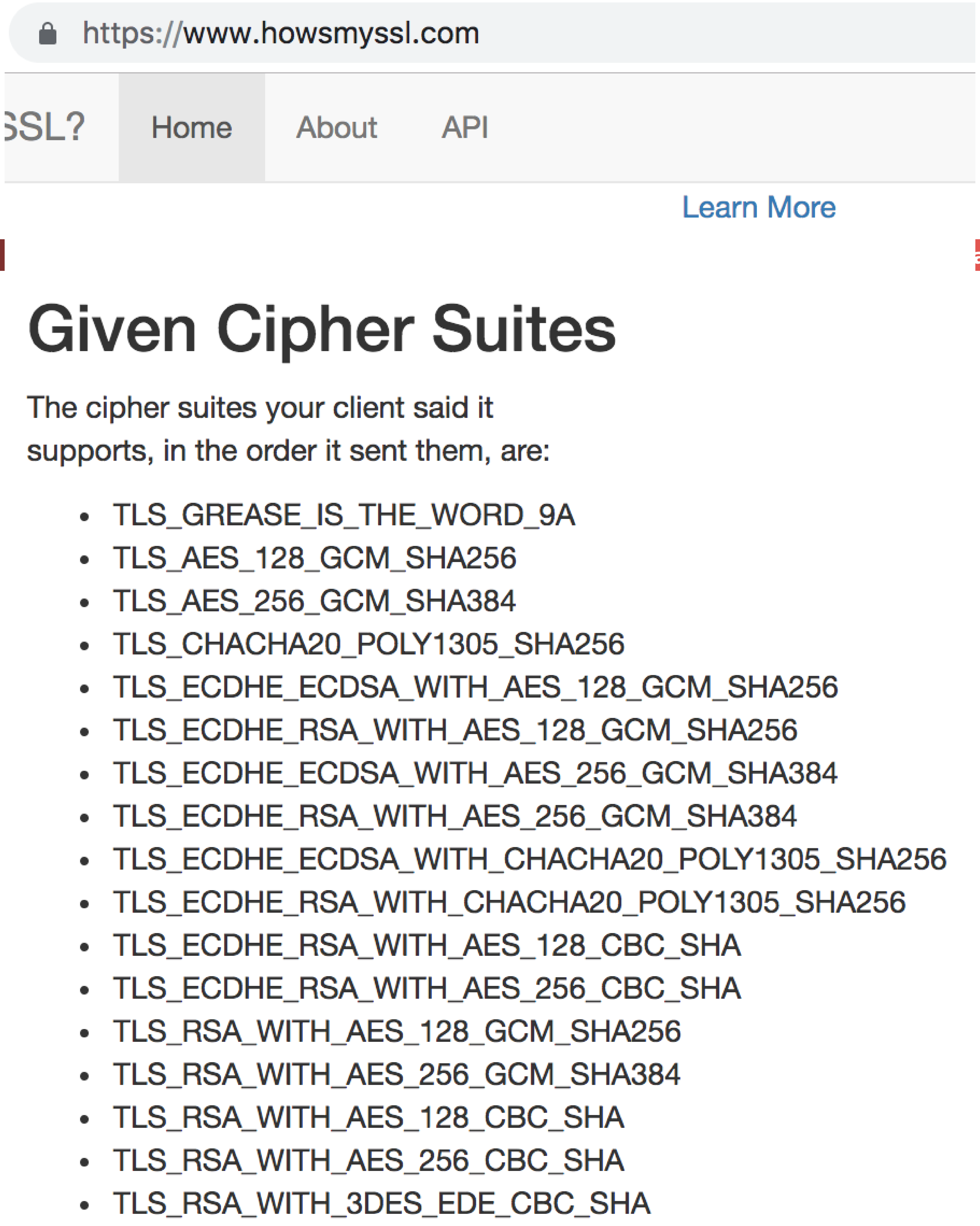
Alternative: Ephemeral Key Exchange via Diffie-Hellman

- For Diffie-Hellman (DHE), server generates random a, sends public parameters and g^a mod p
  - Signed with server's private key
- Browser verifies signature
- Browser generates random b, computes PS = g^{ab} mod p, sends g^b mod p to server
- Server also computes PS = g^{ab} mod p
- Remainder is as before: from PS, R_B, and R_S, browser & server derive symm. cipher keys (C_B, C_S) and MAC integrity keys (I_B, I_S), etc…



Cipher Suite Negotiation

- Chrome's cipher-suite information
  - Client sends to the server
  - Server then choses which one it wants
    - It ***should*** pick the common mode that both prefer based on order
- First is a dummy to keep servers honest
- Then its the bulk encryption only options
- Then key exchanges w encryption mode
  - Description is key exchange, signature (if necessary), and then bulk cipher & hash



Given Cipher Suites

The cipher suites your client said it supports, in the order it sent them, are:

- TLS_GREASE_IS_THE_WORD_9A
- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA

Why R_b and R_s?

- Both R_b and R_s act to affect the keys... Why?
  - Keys = $F(R_b \mid\mid R_s \mid\mid PS)$
- Needed to prevent a ***replay attack***
  - Attacker captures the handshake from either the client or server and replays it...
- If the other side choses a different R the next time...
  - The replay attack fails.
- But you ***don't need to check*** for reuse by the other side..
  - Just make sure you don't reuse it on your side!

And Sabotaged pRNGs...

- Let us assume the server is using DHE...
  - If an attacker can know ***a***, they have all the information needed to decrypt the traffic:
    - Since PS = g^{ab}, and can see g^b.
- TLS spews a lot of "random" numbers publicly as well
  - Nonces in the crypto, R_s, etc...
- If the server uses a bad pRNG which is both sabotaged and doesn't have ***rollback resistance***...
  - Dual_EC DRBG where you know the secret used to create the generator...
  - ANSI X9.31: An AES based one with a secret key...
- Attacker sees the handshake, sees subsequent PRNG calls, works ***backwards*** to get the secret
  - Attack of the week: DUHK
  - https://blog.cryptographyengineering.com/2017/10/23/attack-of-the-week-duhk/

Forward Secrecy Modes...

- The real benefit from DHE/ECDHE "forward secret" modes
  - Reminder: Forward Secrecy: Even if the attacker later compromises the server's private key, the attacker can't compromise previous traffic
- It makes it far more difficult to use even ***after*** an attacker compromises the server's private key
  - Attacker has to be a full MitM:
    Do the handshake to the client and a separate one for the server

End-to-End ⇒ Powerful Protections

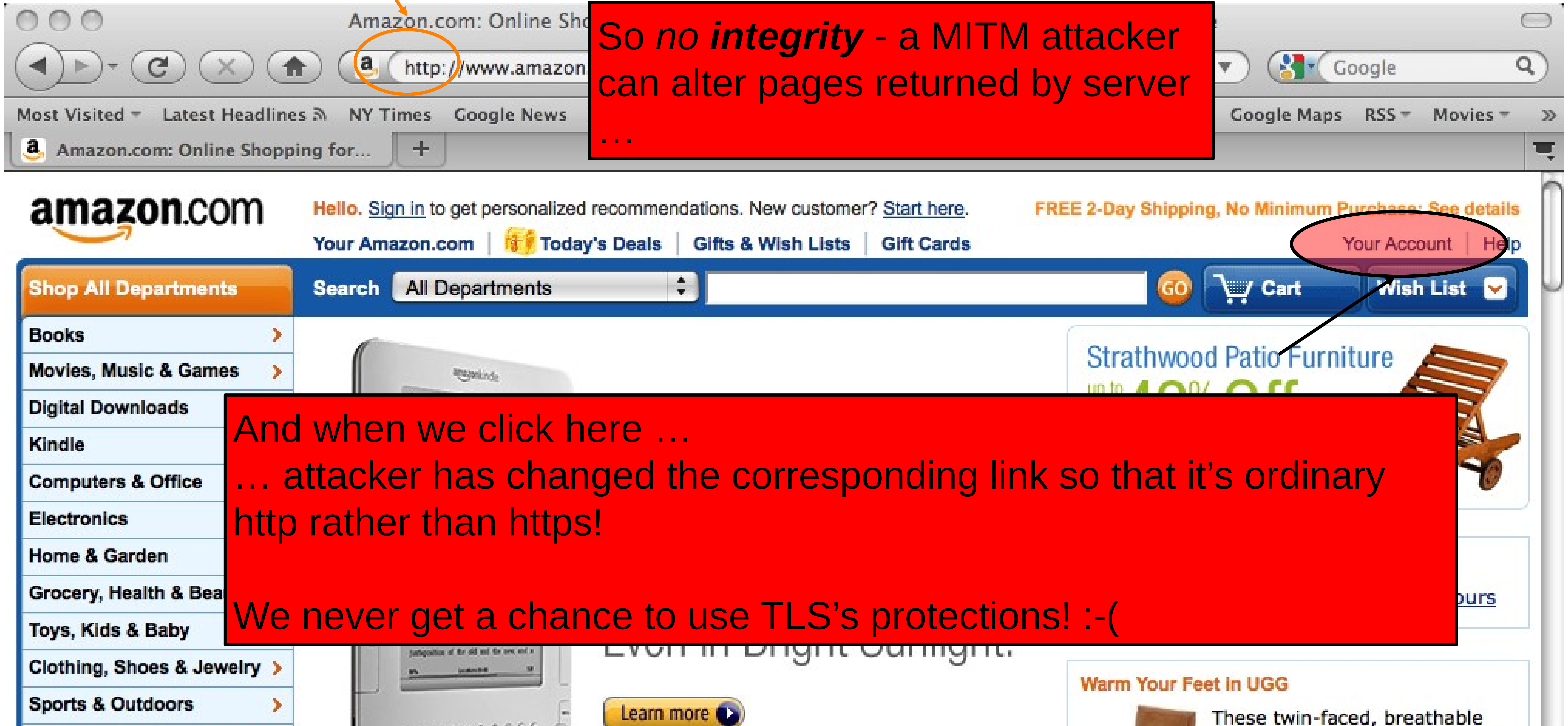
- Attacker runs a sniffer to capture our WiFi session?
  - But: encrypted communication is unreadable
    - Attacker doesn't learn contents, but learns metadata (browsing history)!
- DNS cache poisoning?
  - Client goes to wrong server
  - But: detects impersonation
    - No problem!
- Attacker hijacks our connection, injects new traffic
  - But: data receiver rejects it due to failed integrity check since all communication has a mac on it
    - No problem!
- Only thing a ***full man-in-the-middle*** attacker can do is inject RSTs, inject invalid packets, or drop packets: limited to DoS

SSL/TLS Problem: Revocation

- A site screws up and an attacker steals the private key associated with a certificate, what now?
  - Certificates have a timestamp and are only good for a specified time
    - But this time is measured in years!?!?
- Two mitigations:
  - Certificate revocation lists
    - Your browser occasionally calls back to get a list of "no longer accepted" certificates
  - OSCP
    - Online Certificate Status Protocol: https://en.wikipedia.org/wiki/Online_Certificate_Status_Protocol

“sslstrip”
(Amazon FINALLY fixed this recently)

Regular web surfing: http: URL



So *no **integrity*** - a MITM attacker can alter pages returned by server …

And when we click here …

… attacker has changed the corresponding link so that it’s ordinary http rather than https!

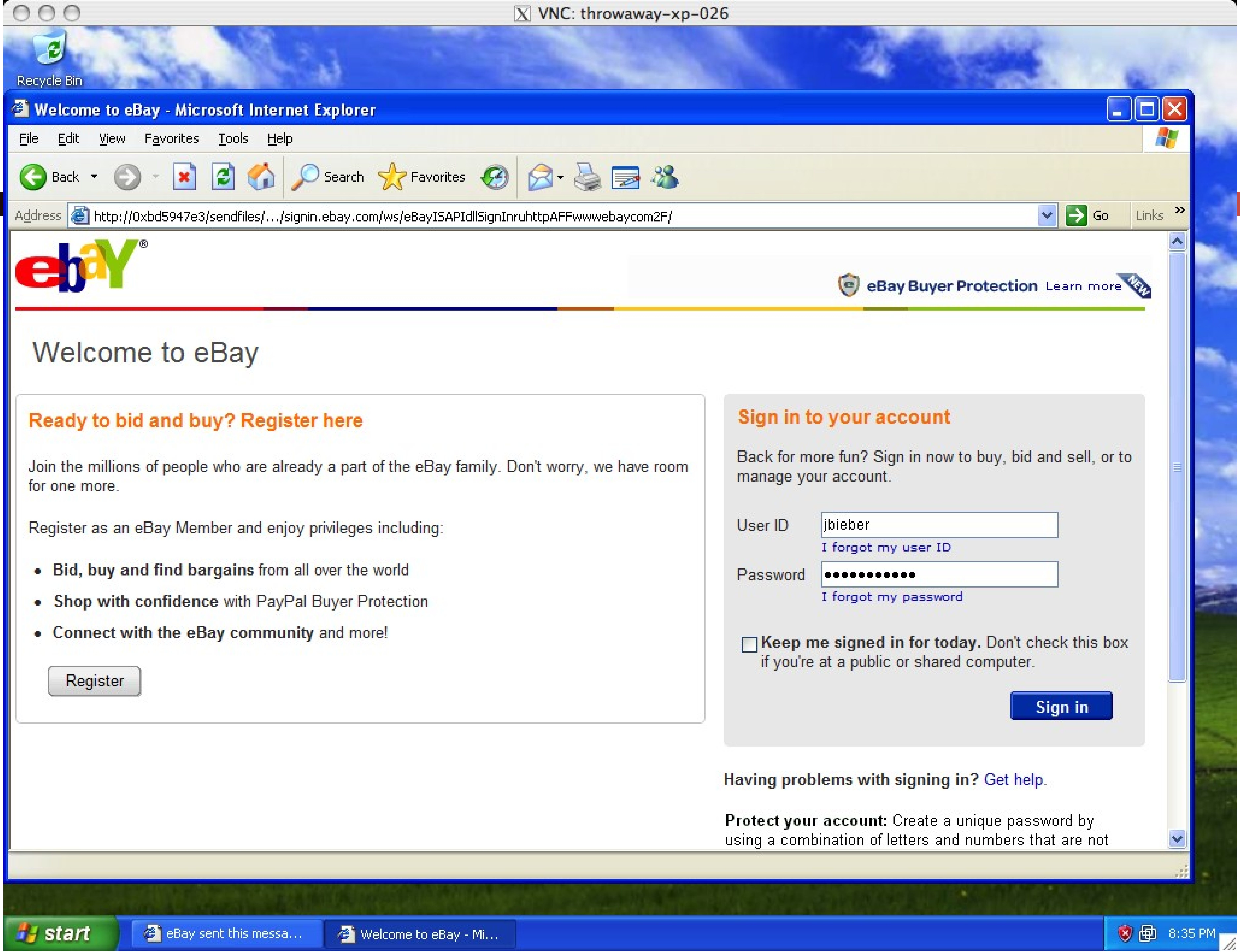
We never get a chance to use TLS’s protections! :-(

SSL / TLS Limitations, cont.

- Problems that SSL / TLS does not take care of ?
- Censorship
- SQL injection / XSS / server-side coding/logic flaws
- Vulnerabilities introduced by server inconsistencies
- Browser and server bugs
- Bad passwords

TLS/SSL Trust Issues

- User has to make correct trust decisions ...



http://0xbd5947e3/sendfiles/.../signin.ebay.com/ws/eBayISAPIdllSignInruhttpAFFwwwebaycom2F/

eBay Buyer Protection Learn more

Welcome to eBay

Ready to bid and buy? Register here

Join the millions of people who are already a part of the eBay family. Don't worry, we have room for one more.

Register as an eBay Member and enjoy privileges including:

- **Bid, buy and find bargains** from all over the world
- **Shop with confidence** with PayPal Buyer Protection
- **Connect with the eBay community** and more!

Register

Sign in to your account

Back for more fun? Sign in now to buy, bid and sell, or to manage your account.

User ID: jbieber

I forgot my user ID

Password: ●●●●●●●●●●●

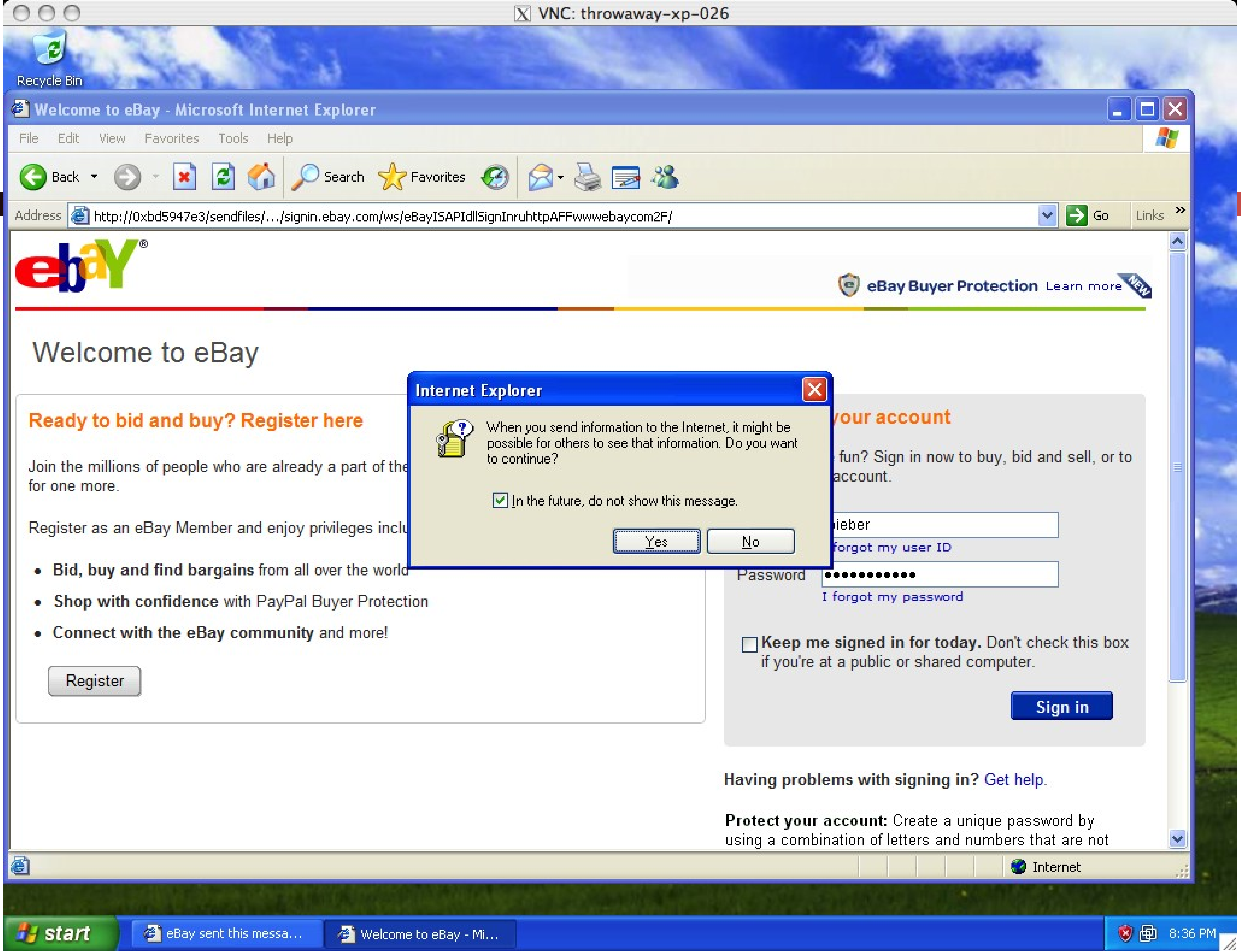
I forgot my password

☐ **Keep me signed in for today.** Don't check this box if you're at a public or shared computer.

Sign in

Having problems with signing in? Get help.

Protect your account: Create a unique password by using a combination of letters and numbers that are not



http://0xbd5947e3/sendfiles/.../signin.ebay.com/ws/eBayISAPIdllSignInruhttpAFFwwwebaycom2F/

Welcome to eBay

Ready to bid and buy? Register here

Join the millions of people who are already a part of the
for one more.

Register as an eBay Member and enjoy privileges inclu

- **Bid, buy and find bargains** from all over the world
- **Shop with confidence** with PayPal Buyer Protection
- **Connect with the eBay community** and more!

Register

Internet Explorer

When you send information to the Internet, it might be possible for others to see that information. Do you want to continue?

☑ In the future, do not show this message.

Yes No

your account

fun? Sign in now to buy, bid and sell, or to
account.

ieber

forgot my user ID

Password ●●●●●●●●●●●

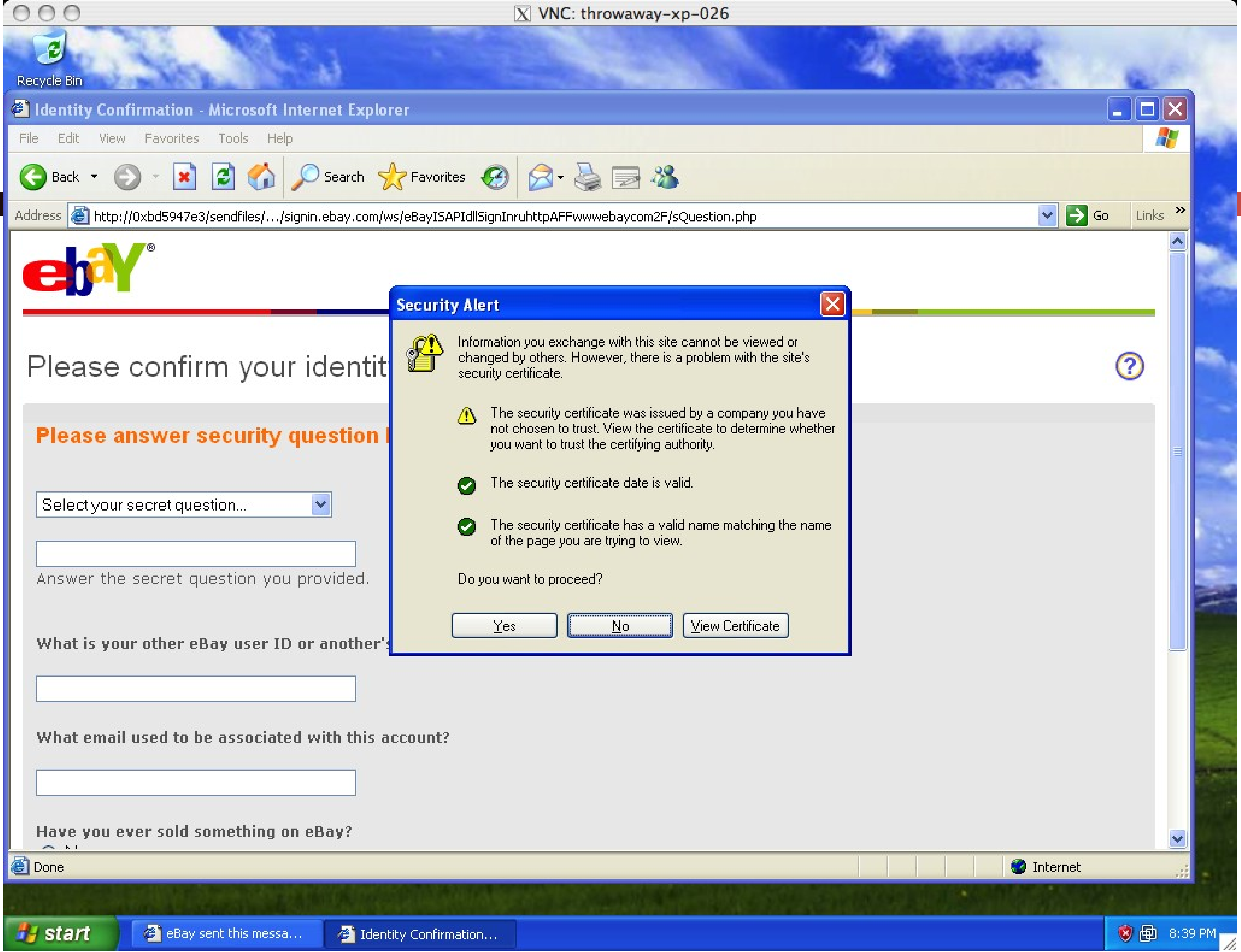
I forgot my password

☐ **Keep me signed in for today.** Don't check this box if you're at a public or shared computer.

Sign in

Having problems with signing in? Get help.

Protect your account: Create a unique password by using a combination of letters and numbers that are not



http://0xbd5947e3/sendfiles/.../signin.ebay.com/ws/eBayISAPIdllSignInruhttpAFFwwwebaycom2F/sQuestion.php

Please confirm your identit

Please answer security question

Select your secret question...

Answer the secret question you provided.

What is your other eBay user ID or another's

What email used to be associated with this account?

Have you ever sold something on eBay?

Security Alert

Information you exchange with this site cannot be viewed or changed by others. However, there is a problem with the site's security certificate.

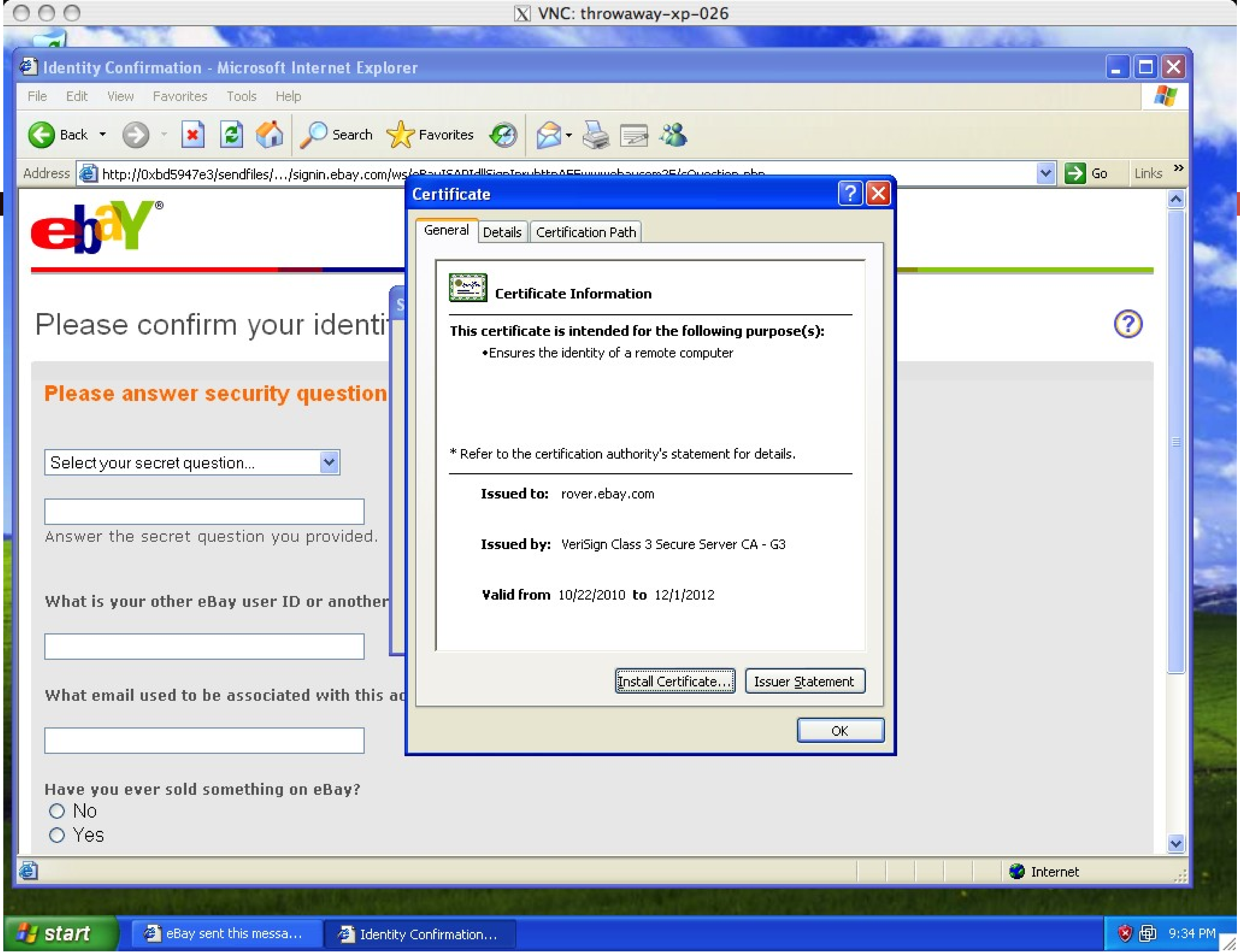
⚠ The security certificate was issued by a company you have not chosen to trust. View the certificate to determine whether you want to trust the certifying authority.

✓ The security certificate date is valid.

✓ The security certificate has a valid name matching the name of the page you are trying to view.

Do you want to proceed?

Yes | No | View Certificate



http://0xbd5947e3/sendfiles/…/signin.ebay.com/ws/

Please confirm your identi

Please answer security question

Select your secret question...

Answer the secret question you provided.

What is your other eBay user ID or another

What email used to be associated with this ac

Have you ever sold something on eBay?

○ No

○ Yes

Certificate Information

This certificate is intended for the following purpose(s):

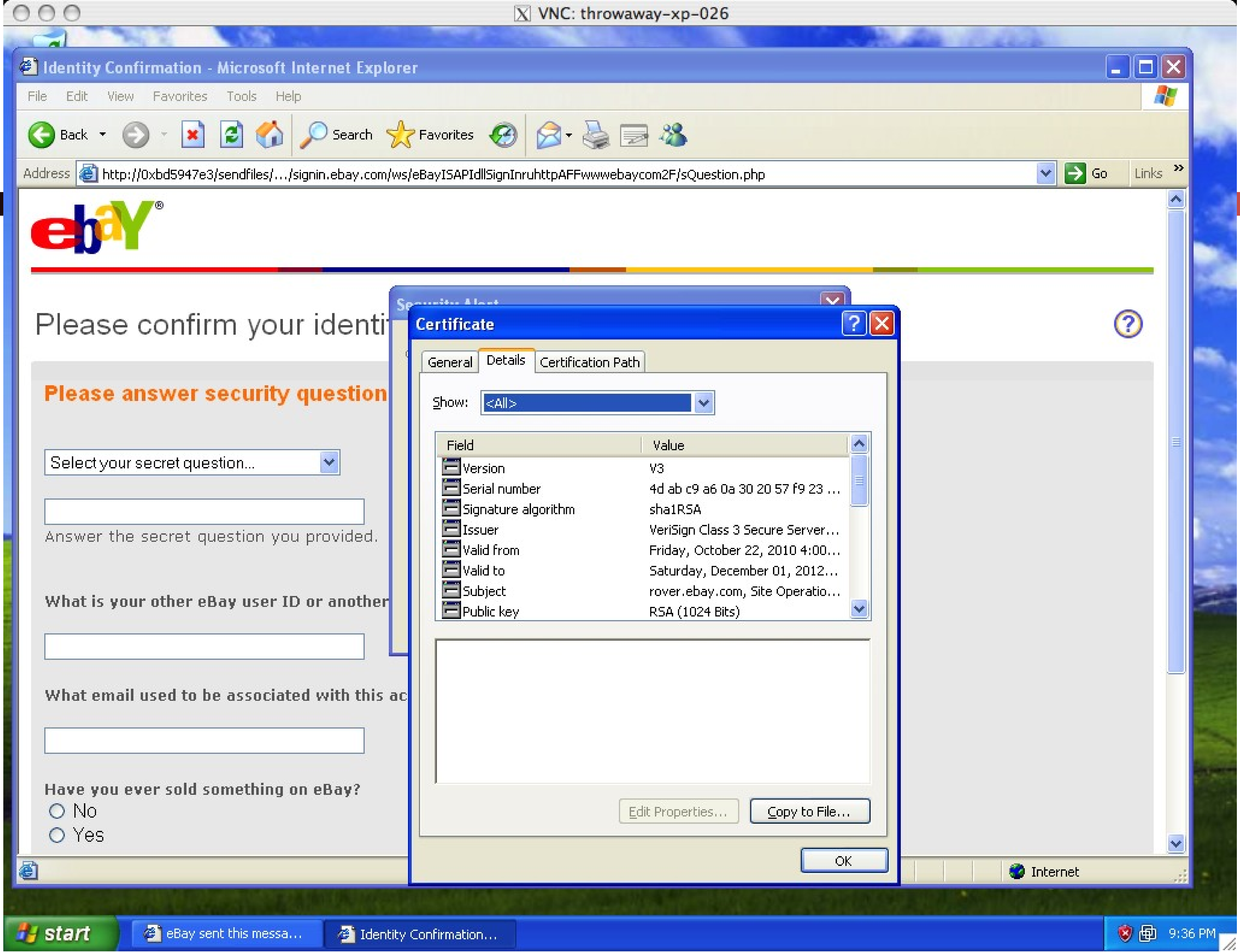
- Ensures the identity of a remote computer

* Refer to the certification authority's statement for details.

Issued to: rover.ebay.com

Issued by: VeriSign Class 3 Secure Server CA - G3

Valid from 10/22/2010 **to** 12/1/2012



http://0xbd5947e3/sendfiles/.../signin.ebay.com/ws/eBayISAPIdllSignInruhttpAFFwwwebaycom2F/sQuestion.php

Please confirm your identi

Please answer security question

Select your secret question...

Answer the secret question you provided.

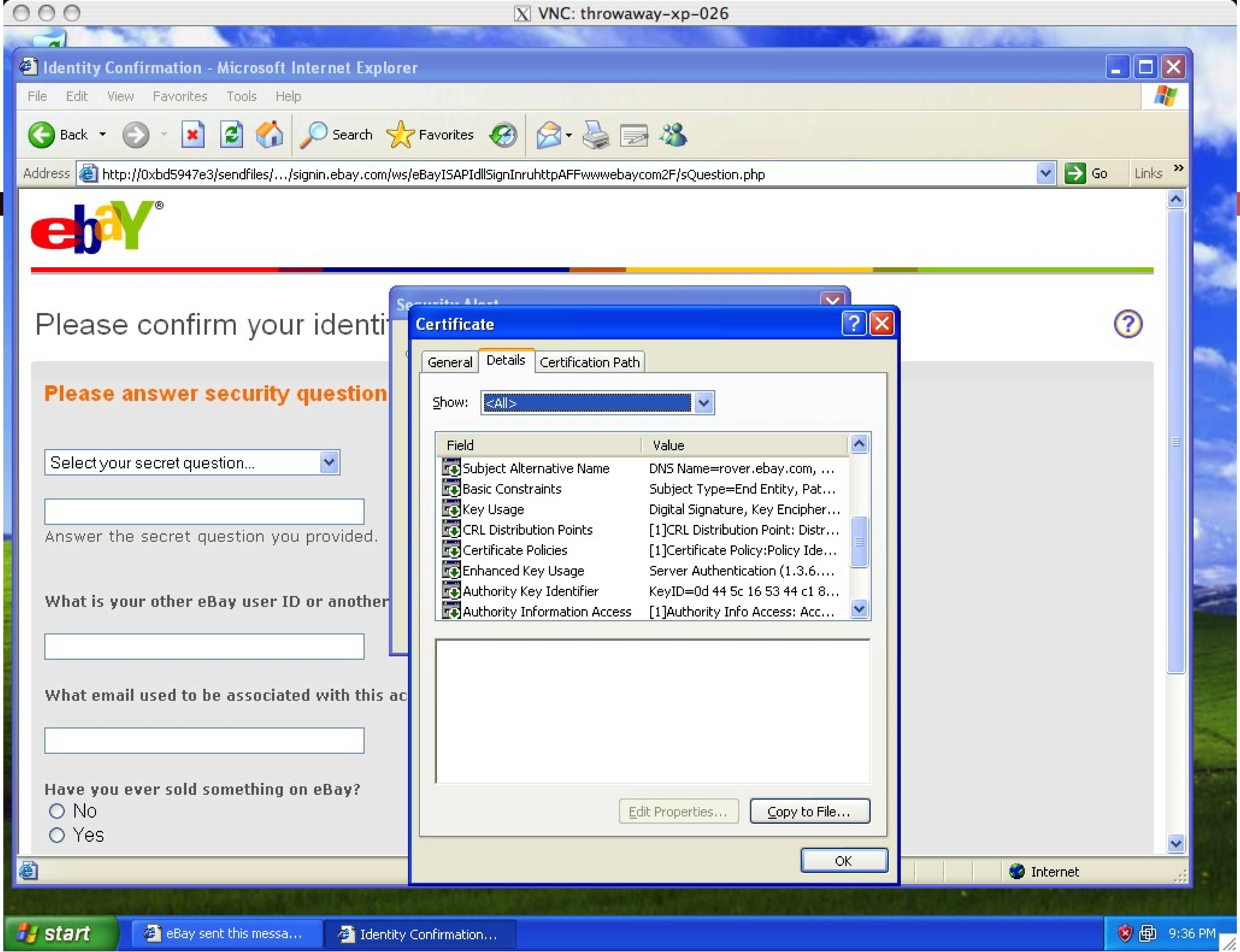
What is your other eBay user ID or another

What email used to be associated with this ac

Have you ever sold something on eBay?
○ No
○ Yes

Show: <All>

Field	Value
Version	V3
Serial number	4d ab c9 a6 0a 30 20 57 f9 23 ...
Signature algorithm	sha1RSA
Issuer	VeriSign Class 3 Secure Server...
Valid from	Friday, October 22, 2010 4:00...
Valid to	Saturday, December 01, 2012...
Subject	rover.ebay.com, Site Operatio...
Public key	RSA (1024 Bits)



VNC: throwaway-xp-026

Identity Confirmation - Microsoft Internet Explorer

Address: http://0xbd5947e3/sendfiles/…/signin.ebay.com/ws/eBayISAPIdllSignInruhttpAFFwwwebaycom2F/sQuestion.php

Please confirm your identi

Please answer security question

Select your secret question...

Answer the secret question you provided.

What is your other eBay user ID or another

What email used to be associated with this ac

Have you ever sold something on eBay?

○ No

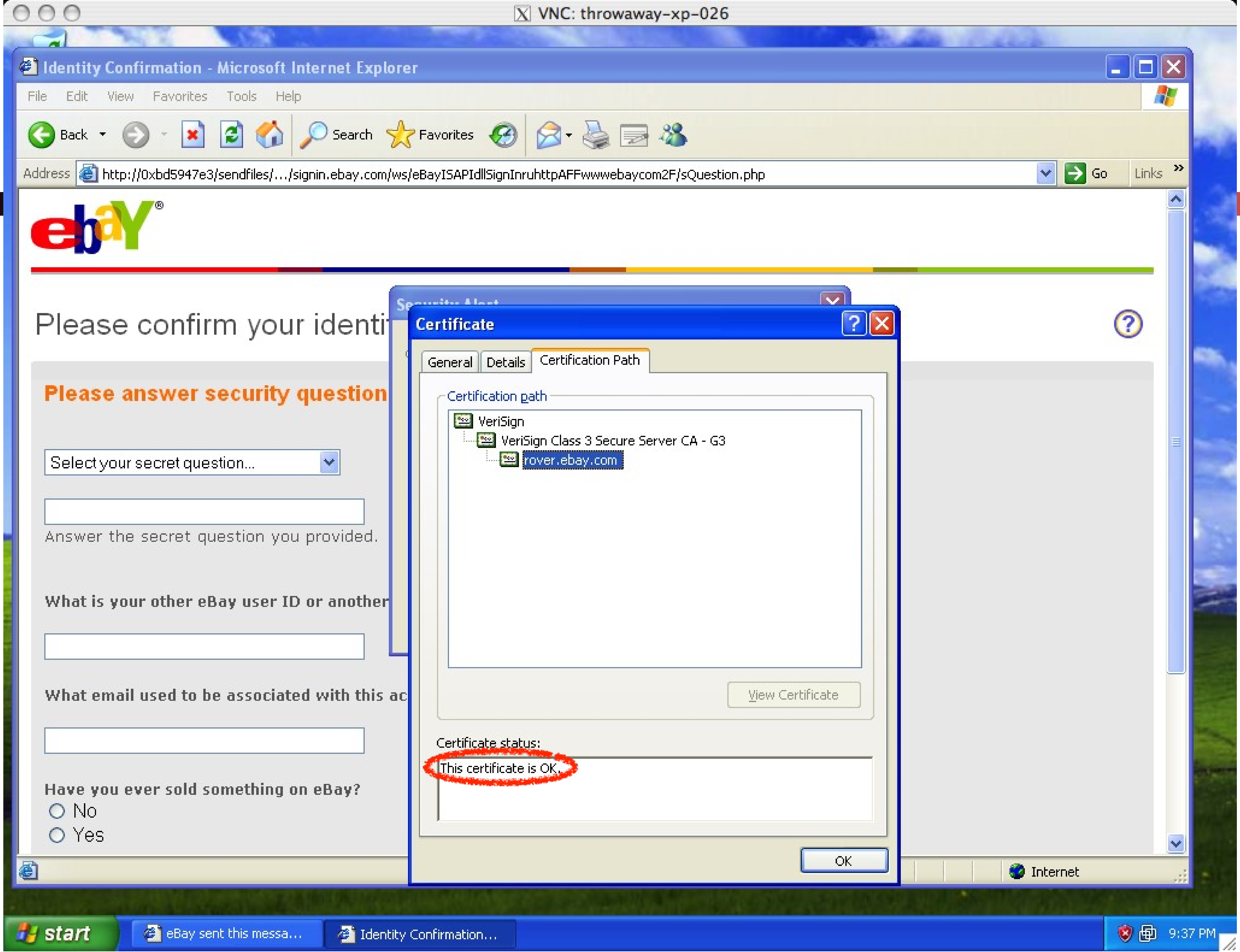
○ Yes

Security Alert

Certificate

Show: <All>

Field	Value
Subject Alternative Name	DNS Name=rover.ebay.com, ...
Basic Constraints	Subject Type=End Entity, Pat...
Key Usage	Digital Signature, Key Encipher...
CRL Distribution Points	[1]CRL Distribution Point: Distr...
Certificate Policies	[1]Certificate Policy:Policy Ide...
Enhanced Key Usage	Server Authentication (1.3.6....
Authority Key Identifier	KeyID=0d 44 5c 16 53 44 c1 8...
Authority Information Access	[1]Authority Info Access: Acc...



http://0xbd5947e3/sendfiles/.../signin.ebay.com/ws/eBayISAPIdllSignInruhttpAFFwwwebaycom2F/sQuestion.php

Please confirm your identi

Please answer security question

Select your secret question...

Answer the secret question you provided.

What is your other eBay user ID or another

What email used to be associated with this ac

Have you ever sold something on eBay?

○ No

○ Yes

Certificate

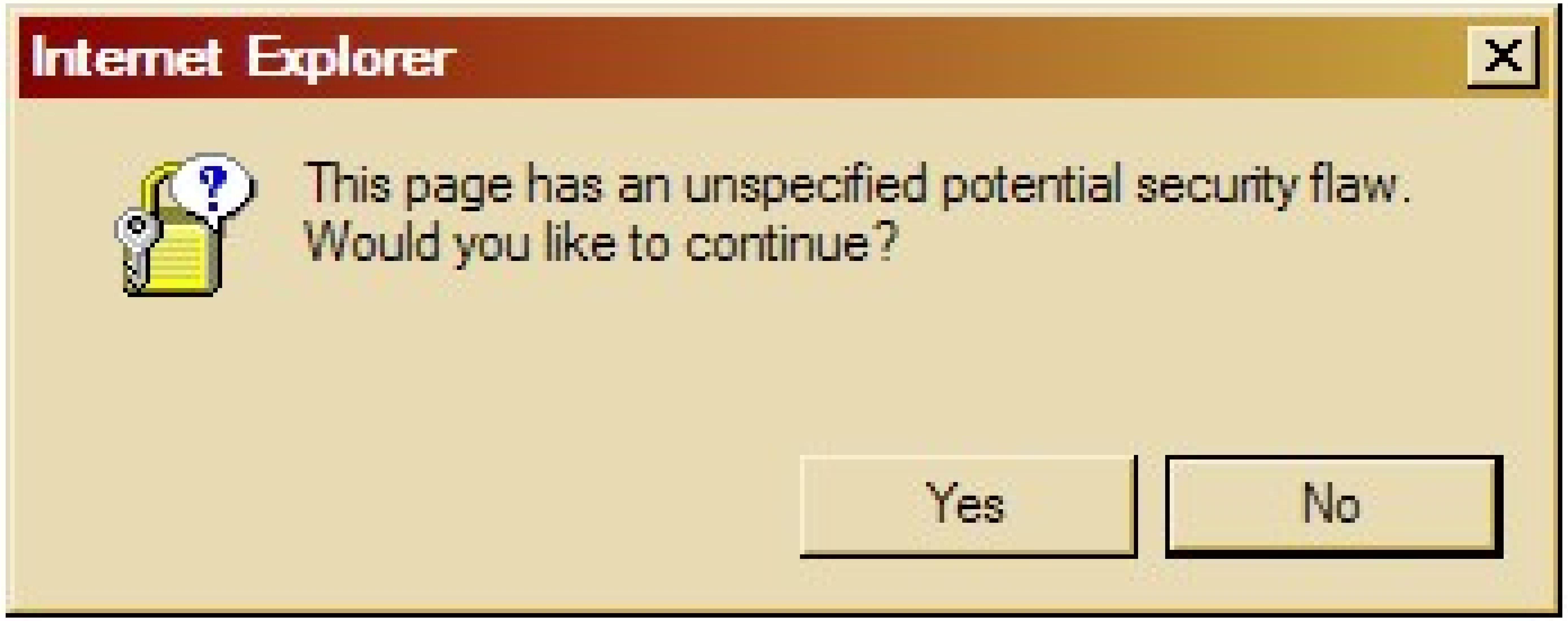
Certification path

- VeriSign
  - VeriSign Class 3 Secure Server CA - G3
    - rover.ebay.com

Certificate status:

This certificate is OK.

The equivalent as seen by most Internet users:



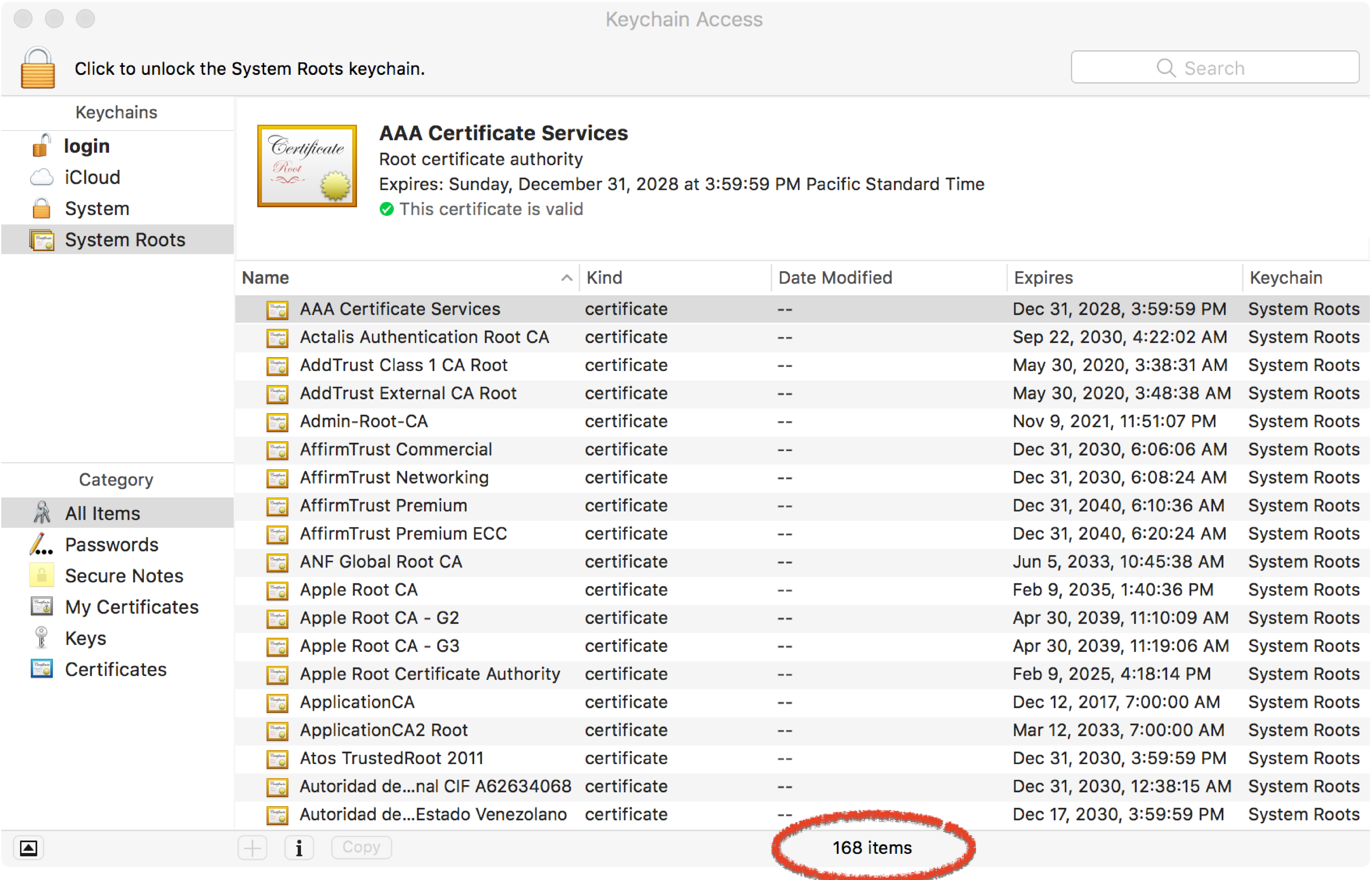
Internet Explorer

This page has an unspecified potential security flaw. Would you like to continue?

(note: an actual Windows error message!)

TLS/SSL Trust Issues, cont.

- *“Commercial certificate authorities protect you from anyone from whom they are unwilling to take money.”*
  - Matt Blaze, circa 2001



Click to unlock the System Roots keychain.

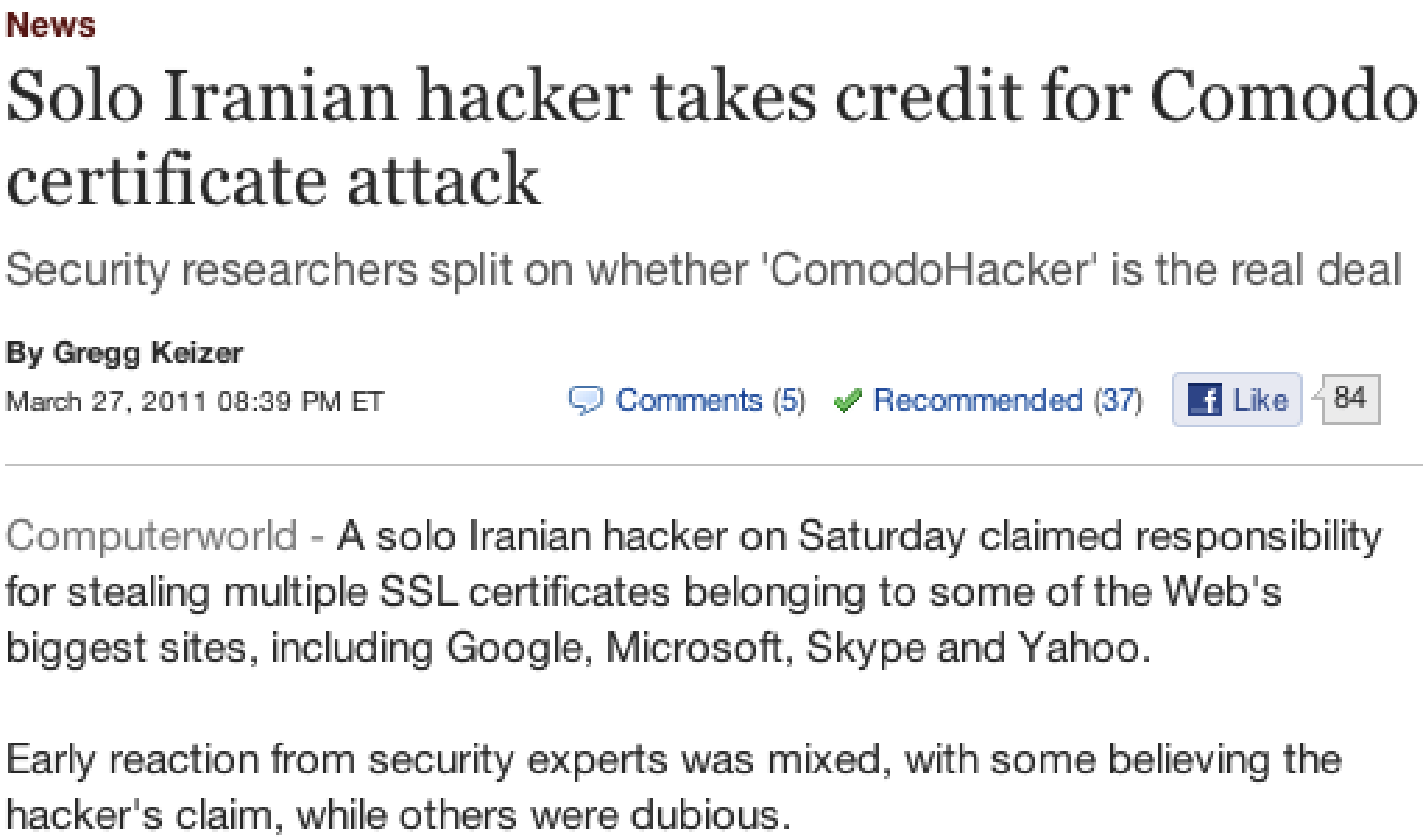
AAA Certificate Services
Root certificate authority
Expires: Sunday, December 31, 2028 at 3:59:59 PM Pacific Standard Time
✓ This certificate is valid

Name	Kind	Date Modified	Expires	Keychain
AAA Certificate Services	certificate	--	Dec 31, 2028, 3:59:59 PM	System Roots
Actalis Authentication Root CA	certificate	--	Sep 22, 2030, 4:22:02 AM	System Roots
AddTrust Class 1 CA Root	certificate	--	May 30, 2020, 3:38:31 AM	System Roots
AddTrust External CA Root	certificate	--	May 30, 2020, 3:48:38 AM	System Roots
Admin-Root-CA	certificate	--	Nov 9, 2021, 11:51:07 PM	System Roots
AffirmTrust Commercial	certificate	--	Dec 31, 2030, 6:06:06 AM	System Roots
AffirmTrust Networking	certificate	--	Dec 31, 2030, 6:08:24 AM	System Roots
AffirmTrust Premium	certificate	--	Dec 31, 2040, 6:10:36 AM	System Roots
AffirmTrust Premium ECC	certificate	--	Dec 31, 2040, 6:20:24 AM	System Roots
ANF Global Root CA	certificate	--	Jun 5, 2033, 10:45:38 AM	System Roots
Apple Root CA	certificate	--	Feb 9, 2035, 1:40:36 PM	System Roots
Apple Root CA - G2	certificate	--	Apr 30, 2039, 11:10:09 AM	System Roots
Apple Root CA - G3	certificate	--	Apr 30, 2039, 11:19:06 AM	System Roots
Apple Root Certificate Authority	certificate	--	Feb 9, 2025, 4:18:14 PM	System Roots
ApplicationCA	certificate	--	Dec 12, 2017, 7:00:00 AM	System Roots
ApplicationCA2 Root	certificate	--	Mar 12, 2033, 7:00:00 AM	System Roots
Atos TrustedRoot 2011	certificate	--	Dec 31, 2030, 3:59:59 PM	System Roots
Autoridad de...nal CIF A62634068	certificate	--	Dec 31, 2030, 12:38:15 AM	System Roots
Autoridad de...Estado Venezolano	certificate	--	Dec 17, 2030, 3:59:59 PM	System Roots

168 items

TLS/SSL Trust Issues

- *“Commercial certificate authorities protect you from anyone from whom they are unwilling to take money.”*
  - Matt Blaze, circa 2001
- So how many CAs do we have to worry about, anyway?
- Of course, it’s not just their greed that matters …



News

Solo Iranian hacker takes credit for Comodo certificate attack

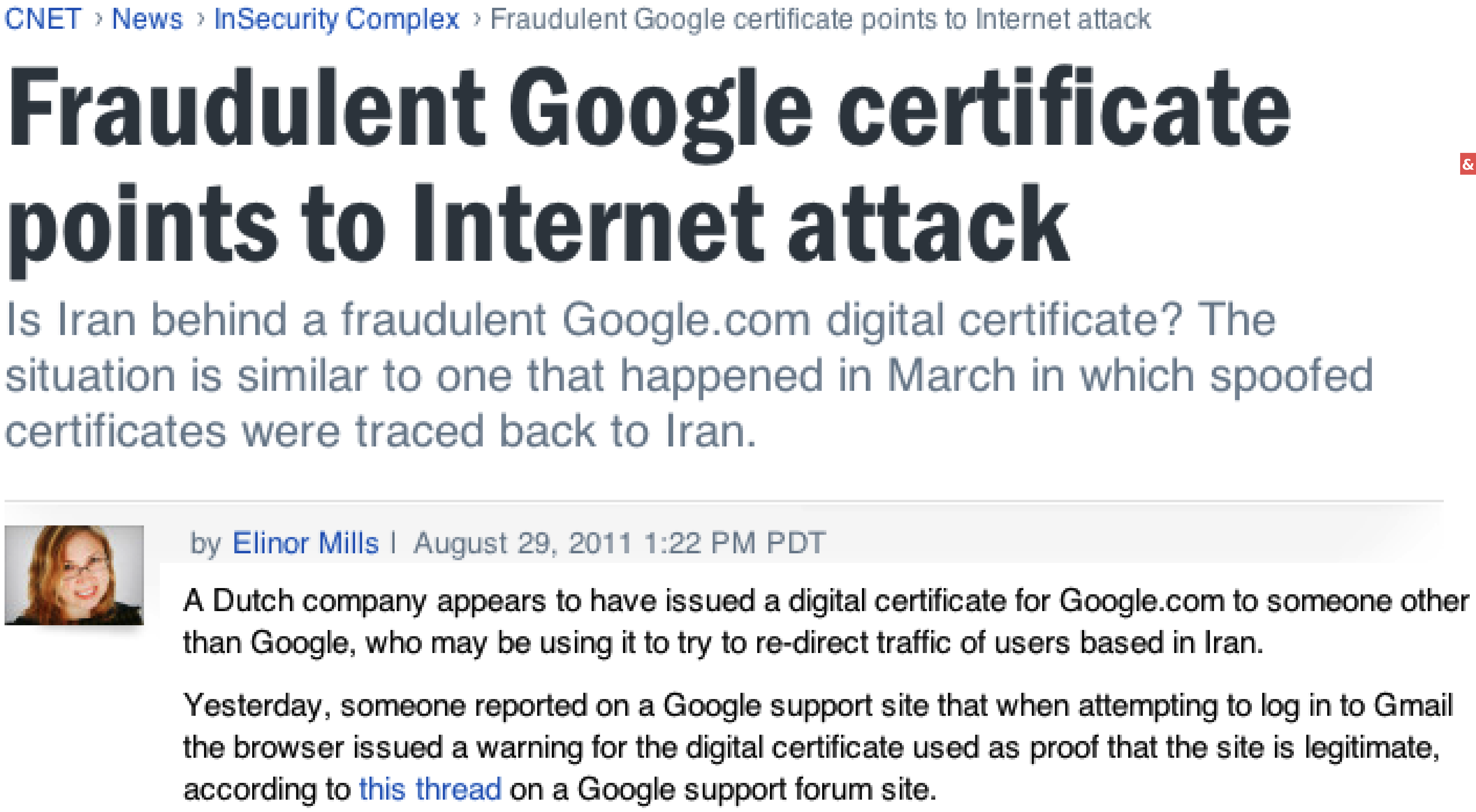
Security researchers split on whether 'ComodoHacker' is the real deal

By Gregg Keizer

March 27, 2011 08:39 PM ET

Computerworld - A solo Iranian hacker on Saturday claimed responsibility for stealing multiple SSL certificates belonging to some of the Web's biggest sites, including Google, Microsoft, Skype and Yahoo.

Early reaction from security experts was mixed, with some believing the hacker's claim, while others were dubious.



CNET › News › InSecurity Complex › Fraudulent Google certificate points to Internet attack

Fraudulent Google certificate points to Internet attack

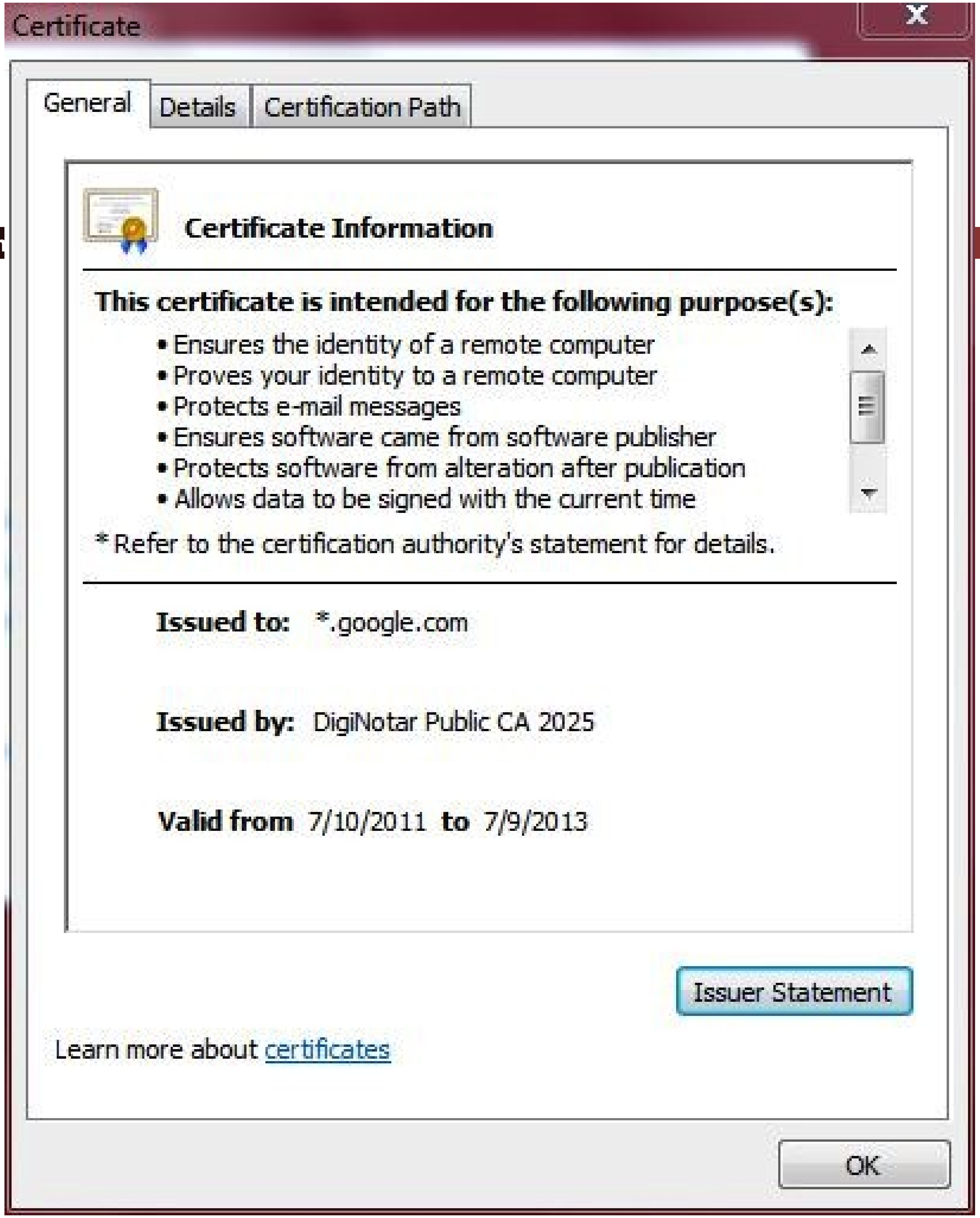
Is Iran behind a fraudulent Google.com digital certificate? The situation is similar to one that happened in March in which spoofed certificates were traced back to Iran.

by Elinor Mills | August 29, 2011 1:22 PM PDT

A Dutch company appears to have issued a digital certificate for Google.com to someone other than Google, who may be using it to try to re-direct traffic of users based in Iran.

Yesterday, someone reported on a Google support site that when attempting to log in to Gmail the browser issued a warning for the digital certificate used as proof that the site is legitimate, according to this thread on a Google support forum site.

& Jawale



Certificate Information

This certificate is intended for the following purpose(s):

- Ensures the identity of a remote computer
- Proves your identity to a remote computer
- Protects e-mail messages
- Ensures software came from software publisher
- Protects software from alteration after publication
- Allows data to be signed with the current time

* Refer to the certification authority's statement for details.

Issued to: *.google.com

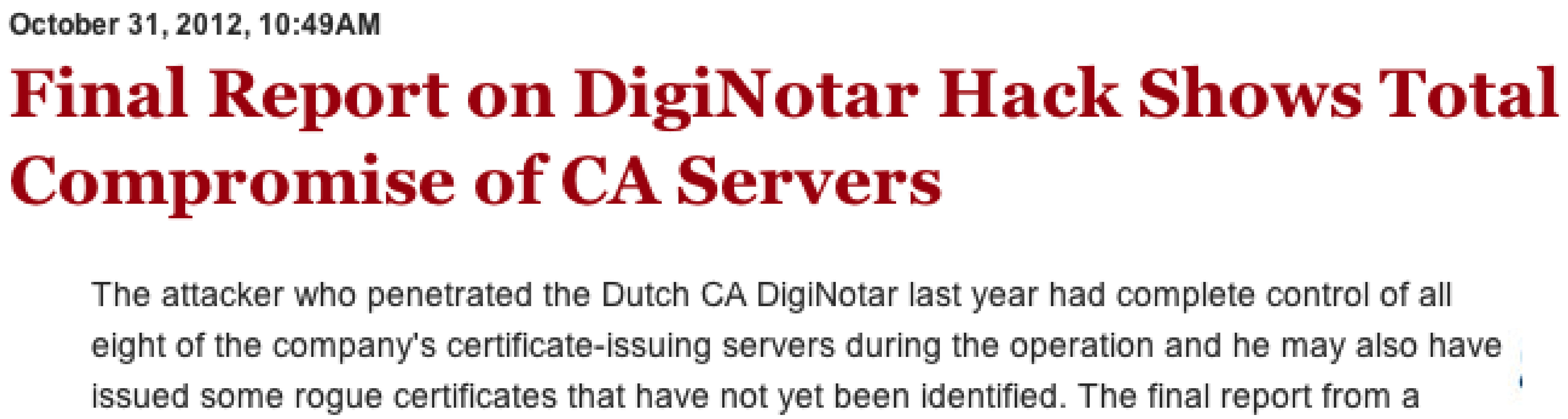
Issued by: DigiNotar Public CA 2025

Valid from 7/10/2011 **to** 7/9/2013

Learn more about certificates

This appears to be a fully valid cert using normal browser validation rules.

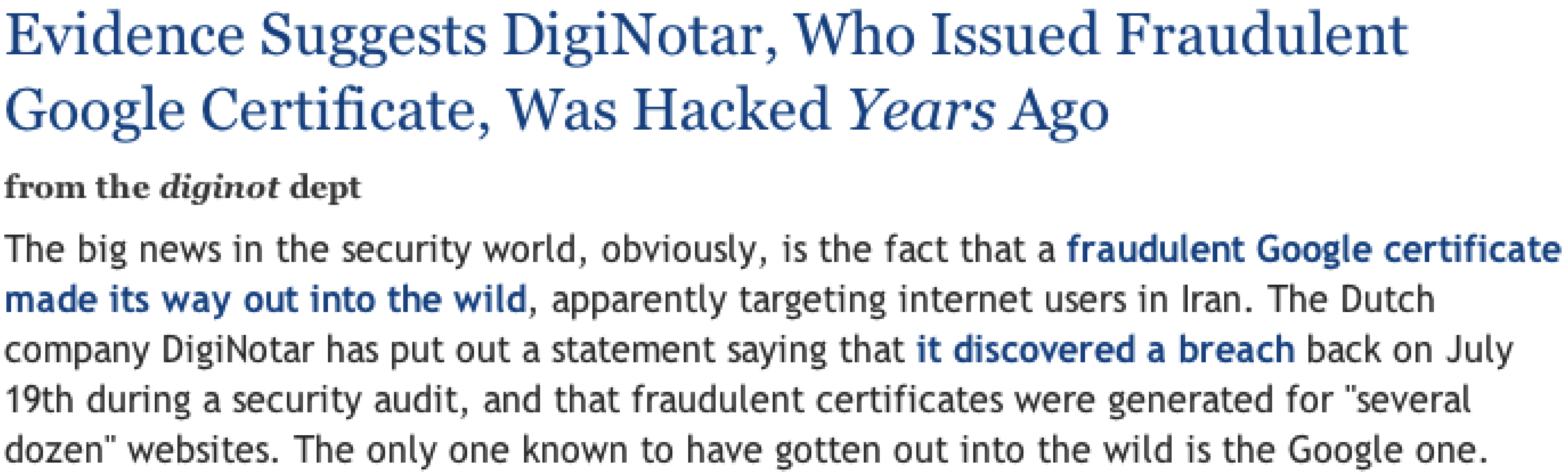
Only detected by Chrome due to its introduction of cert “pinning” – requiring that certs for certain domains must be signed by specific CAs rather than any generally trusted CA



October 31, 2012, 10:49AM

Final Report on DigiNotar Hack Shows Total Compromise of CA Servers

The attacker who penetrated the Dutch CA DigiNotar last year had complete control of all eight of the company's certificate-issuing servers during the operation and he may also have issued some rogue certificates that have not yet been identified. The final report from a



Evidence Suggests DigiNotar, Who Issued Fraudulent Google Certificate, Was Hacked *Years* Ago

from the *diginot* dept

The big news in the security world, obviously, is the fact that a fraudulent Google certificate made its way out into the wild, apparently targeting internet users in Iran. The Dutch company DigiNotar has put out a statement saying that it discovered a breach back on July 19th during a security audit, and that fraudulent certificates were generated for "several dozen" websites. The only one known to have gotten out into the wild is the Google one.

The DigiNotar Fallout

- The result was the “CA Death Sentence”:
  - Web browsers removed it from the trusted root certificate store
- This happened again with “WoSign”
  - A Chinese CA
- WoSign would allow an interesting attack
  - If I controlled RafaelTupynamba.github.com…
  - WoSign would allow me to create a certificate for *.github.com!?!?
  - And a bunch of other shady shenanigans

TLS/SSL Trust Issues

- “Commercial certificate authorities protect you from anyone from whom they are unwilling to take money.”
  - Matt Blaze, circa 2001
- So how many CAs do we have to worry about, anyway?
- Of course, it’s not just their greed that matters …
- And it’s not just their diligence & security that matters…
  - *“A decade ago, I observed that commercial certificate authorities protect you from anyone from whom they are unwilling to take money. That turns out to be wrong; they don't even do that much.”* - Matt Blaze, circa 2010

So the Modern Solution: Invoke Ronald Reagan, “Trust, but Verify”

- Static Certificate Pinning:
  The chrome browser has a list of certificates or certificate authorities that it trusts for given sites
  - Now creating a fake certificate requires attacking a ***particular*** CA
- HPKP Certificate Pinning:
  The web server provides hashes of certificates that should be trusted
  - This is “Leap of Faith”: The first time you assume it is honest but you will catch future changes
- Transparency mechanisms:
  - Public logs provided by certificate authorities
  - Browser extensions (EFF’s TLS observatory)
  - Backbone monitors (ICSI’s TLS notary)

And Making It Cheap: LetsEncrypt...

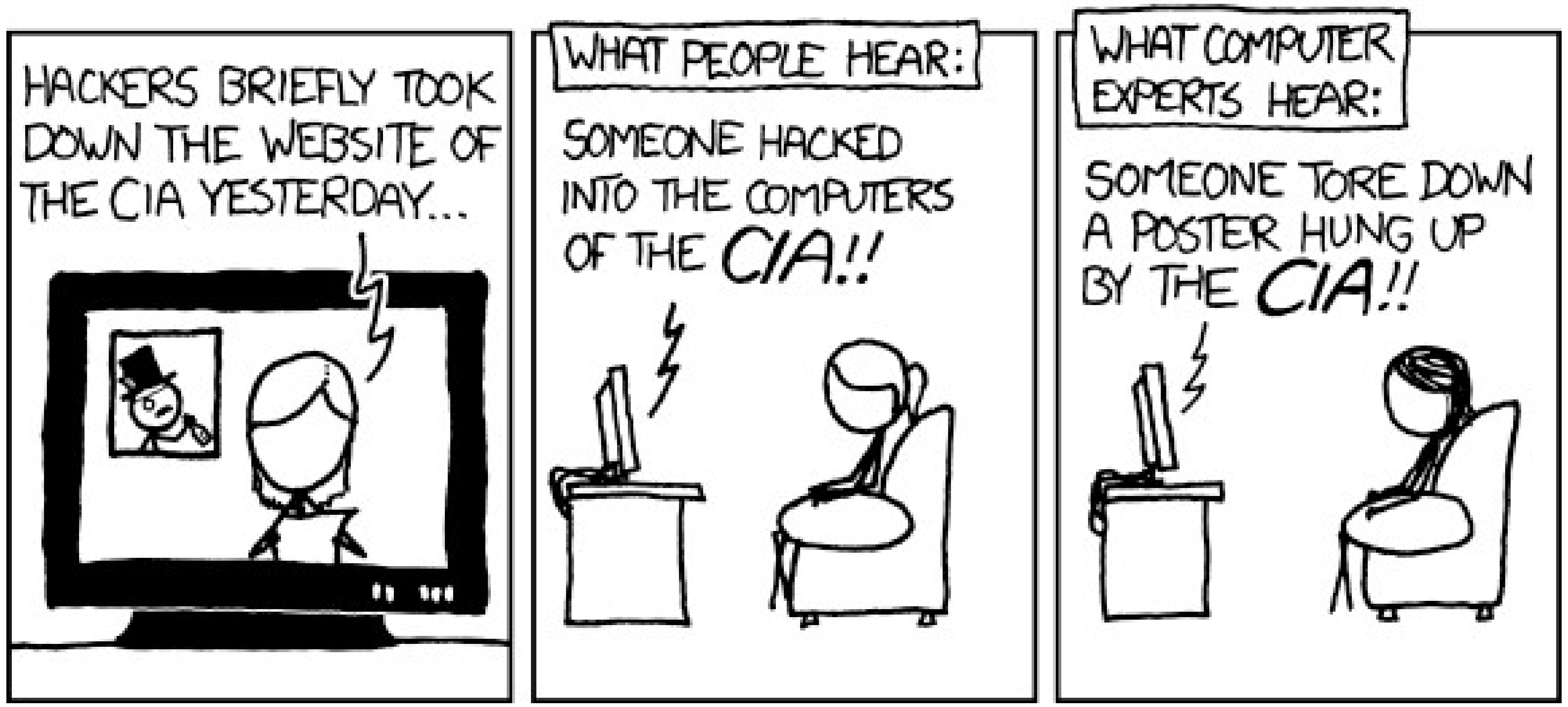
- Coupled to the depreciation of unencrypted HTTP...
  - Need to be able to have HTTPS be just about the same complexity...
- Idea: Make it easy to "prove" you own a web site:
  - Can you write an arbitrary cookie at an arbitrary location?
- Build ***automated*** infrastructure to do this
  - Script to create a private key
  - Generate a certificate signing request
  - PKI authority says "here's a file, put it on the server"
  - Script puts it on the server

Break
Random fact about me...

- I’ve been to a lot of Math Olympiads
  - Traveled to 6 countries
- In the International Mathematical Olympiad, I received my medal from Princess Letizia of Spain (now Queen of Spain)

Theme of This Lecture In Song: 50 Whys to Stop A Server...

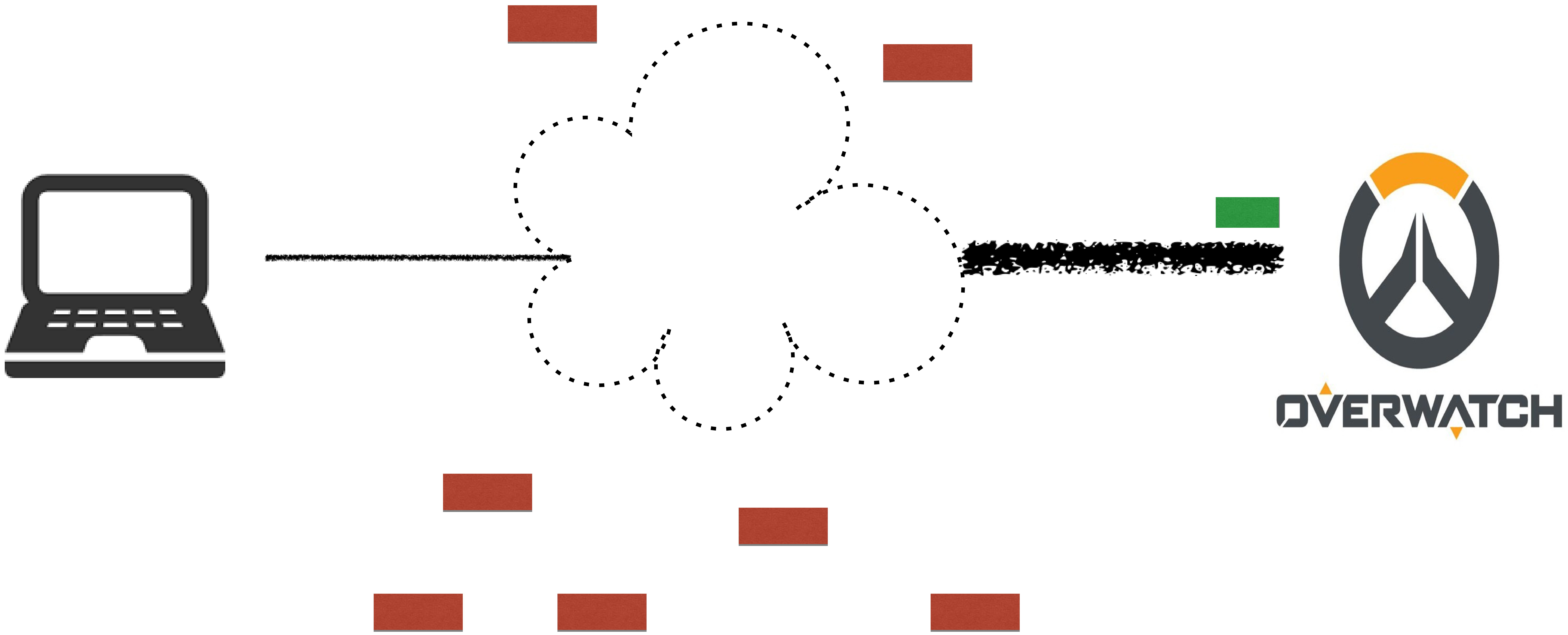
- You are a bad guy...
  - And you want to stop some server from being ***available***
- Why? You name it...
  - Because its hard for someone to frag you in an online game if you "boot" him from the network
  - Because people will pay up to stop the attack
  - Because it conveys a political message
  - Get paid for by others


HACKERS BRIEFLY TOOK DOWN THE WEBSITE OF THE CIA YESTERDAY...
WHAT PEOPLE HEAR:
SOMEONE HACKED INTO THE COMPUTERS OF THE CIA!!
WHAT COMPUTER EXPERTS HEAR:
SOMEONE TORE DOWN A POSTER HUNG UP BY THE CIA!!

The Easy DoS on a System: Resource Consumption...

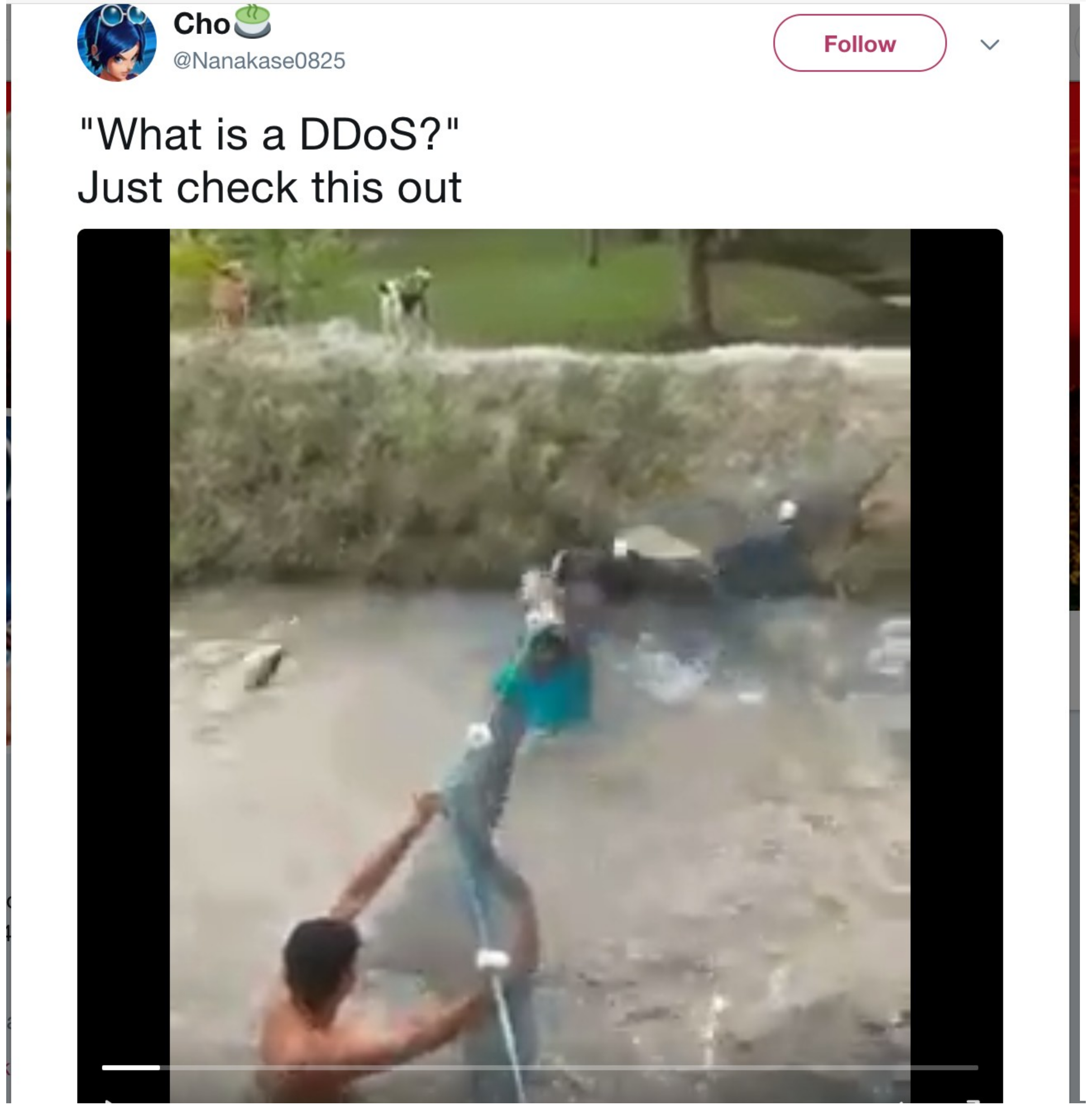
- Bad Dude has an account on your computer...
  - And wants to disrupt your work on Project 2...
- He runs this simple program:
  - while(1):
    - Write random junk to random files
      - (uses disk space, thrashes the disk)
    - Allocate a bunch of RAM and write to it
      - (uses memory)
    - fork()
      - (creates more processes to run)
- Only defense is some form of quota or limits:
  The system itself ***must*** enforce some isolation

The Network DOS

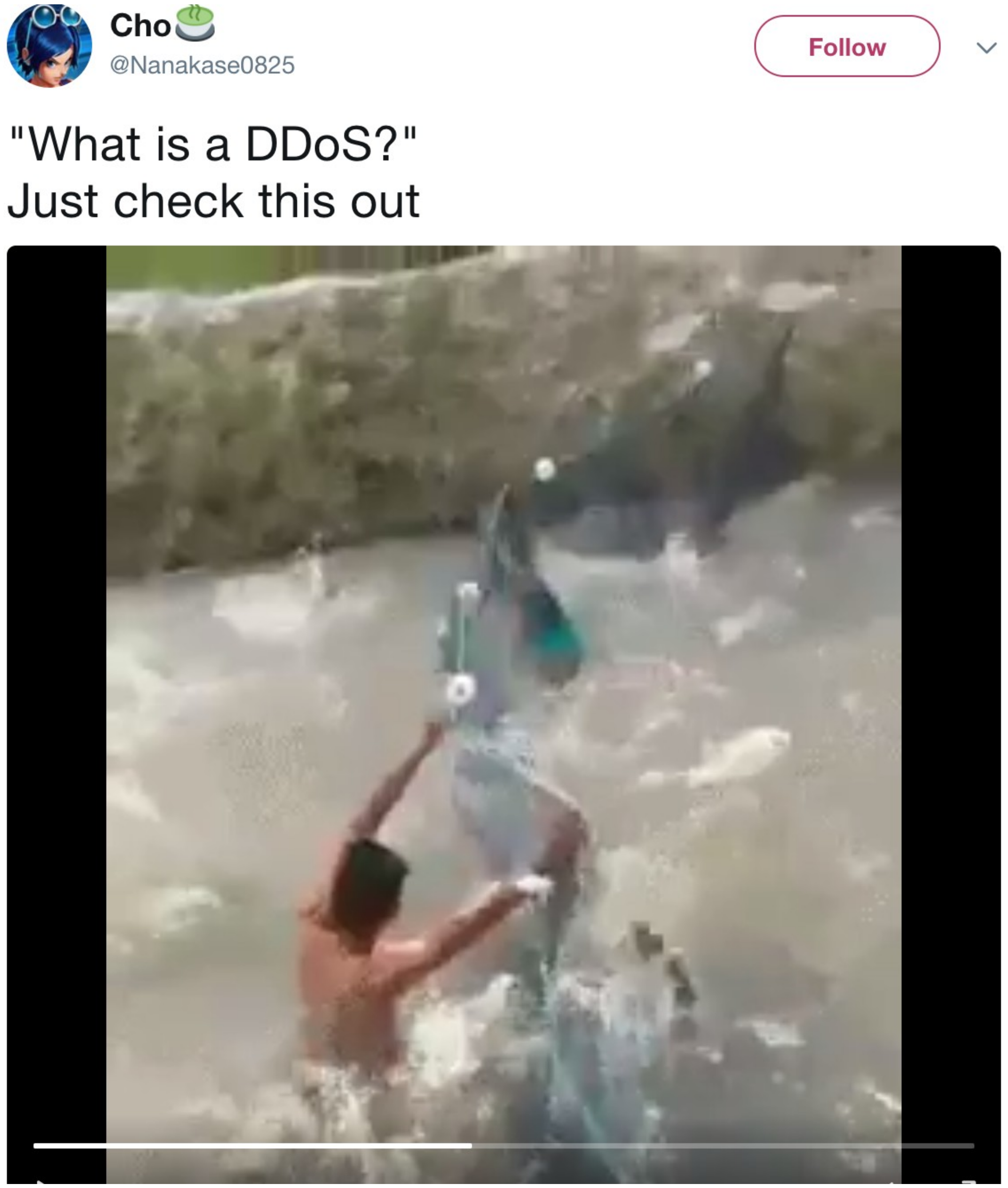


Or, another visual explanation…

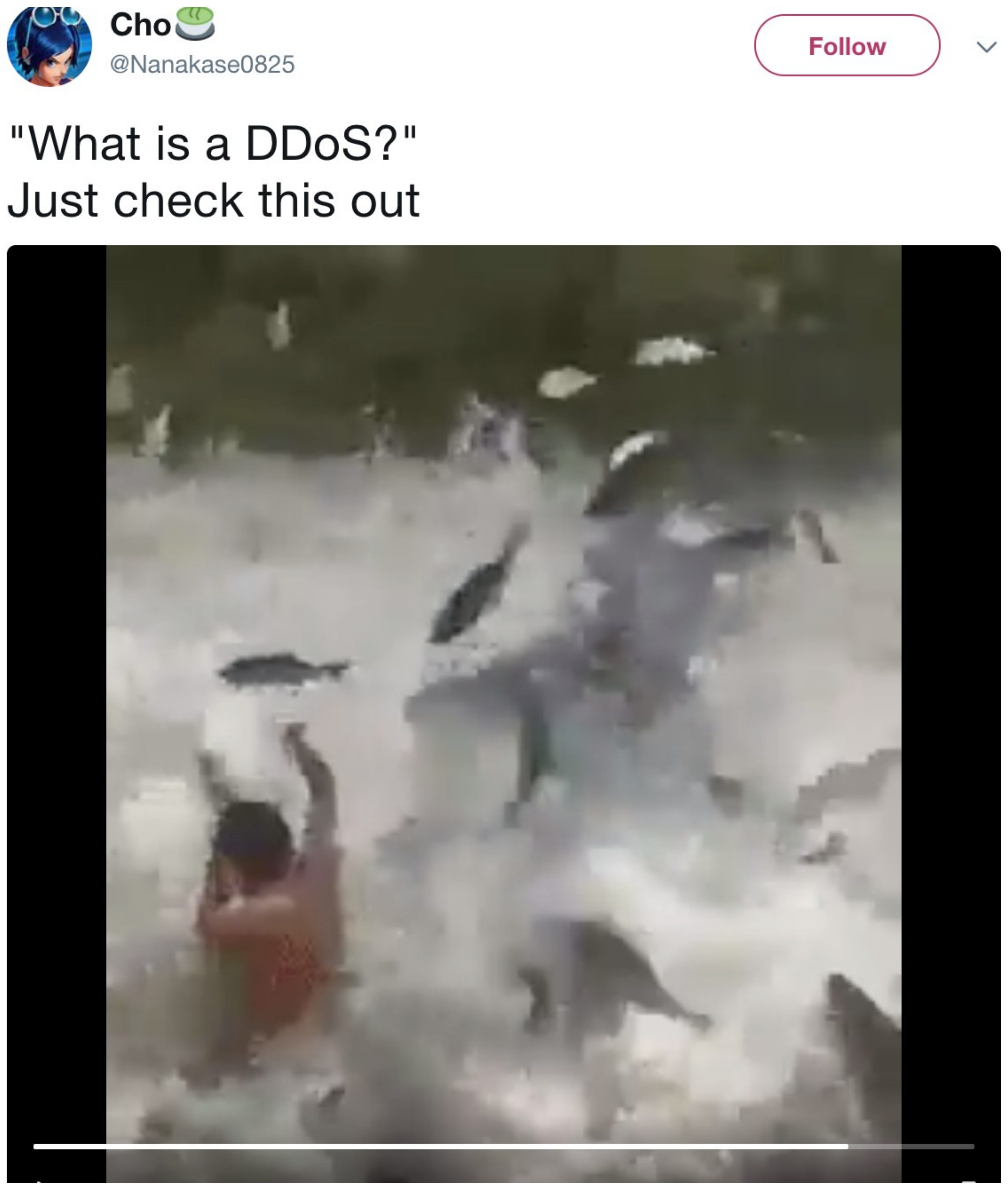
- https://twitter.com/kokonoe0825/status/789536739887112192



"What is a DDoS?"
Just check this out



"What is a DDoS?"
Just check this out



"What is a DDoS?"
Just check this out

DoS & Networks

- How could you DoS a target’s Internet access?
    - Send a zillion packets at them
    - Internet lacks ***isolation*** between traffic of different users!
- What resources does attacker need to pull this off?
    - At least as much sending capacity (***bandwidth***) as the bottleneck link of the target’s Internet connection
        - Attacker sends maximum-sized packets
    - Or: overwhelm the rate at which the bottleneck router can process packets
        - Attacker sends minimum-sized packets!
            - (in order to maximize the packet arrival rate)

Defending Against Network DoS

- Suppose an attacker has access to a beefy system with high-speed Internet access (a “big pipe”).
- They pump out packets towards the target at a very high rate.
- What might the target do to defend against the onslaught?
  - Install a network filter to discard any packets that arrive with attacker’s IP address as their source
    - E.g., **`drop * 66.31.33.7:* -> *:*`**
    - Or it can leverage any other pattern in the flooding traffic that’s not in benign traffic
  - Attacker’s IP address = means of identifying misbehaving user

Filtering Sounds Pretty Easy …

- … but DoS filters can be easily evaded:
  - Make traffic appear as though it's from many hosts
    - Spoof the source address so it can't be used to filter
      - Just pick a random 32-bit number of each packet sent
    - How does a defender filter this?
      - They don't!
      - Best they can hope for is that operators around the world implement anti-spoofing mechanisms (today about 75% do)
  - Use many hosts to send traffic rather than just one
    - Distributed Denial-of-Service = DDoS ("dee-doss")
    - Requires defender to install complex filters
    - How many hosts is "enough" for the attacker?
      - Today they are very cheap to acquire … :-(

It’s Not A “Level Playing Field”

- When defending resources from exhaustion, need to beware of asymmetries, where attackers can consume victim resources with little comparable effort
  - Makes DoS easier to launch
  - Defense costs much more than attack
- Particularly dangerous form of asymmetry: amplification
  - Attacker leverages system’s own structure to pump up the load they induce on a resource

Amplification

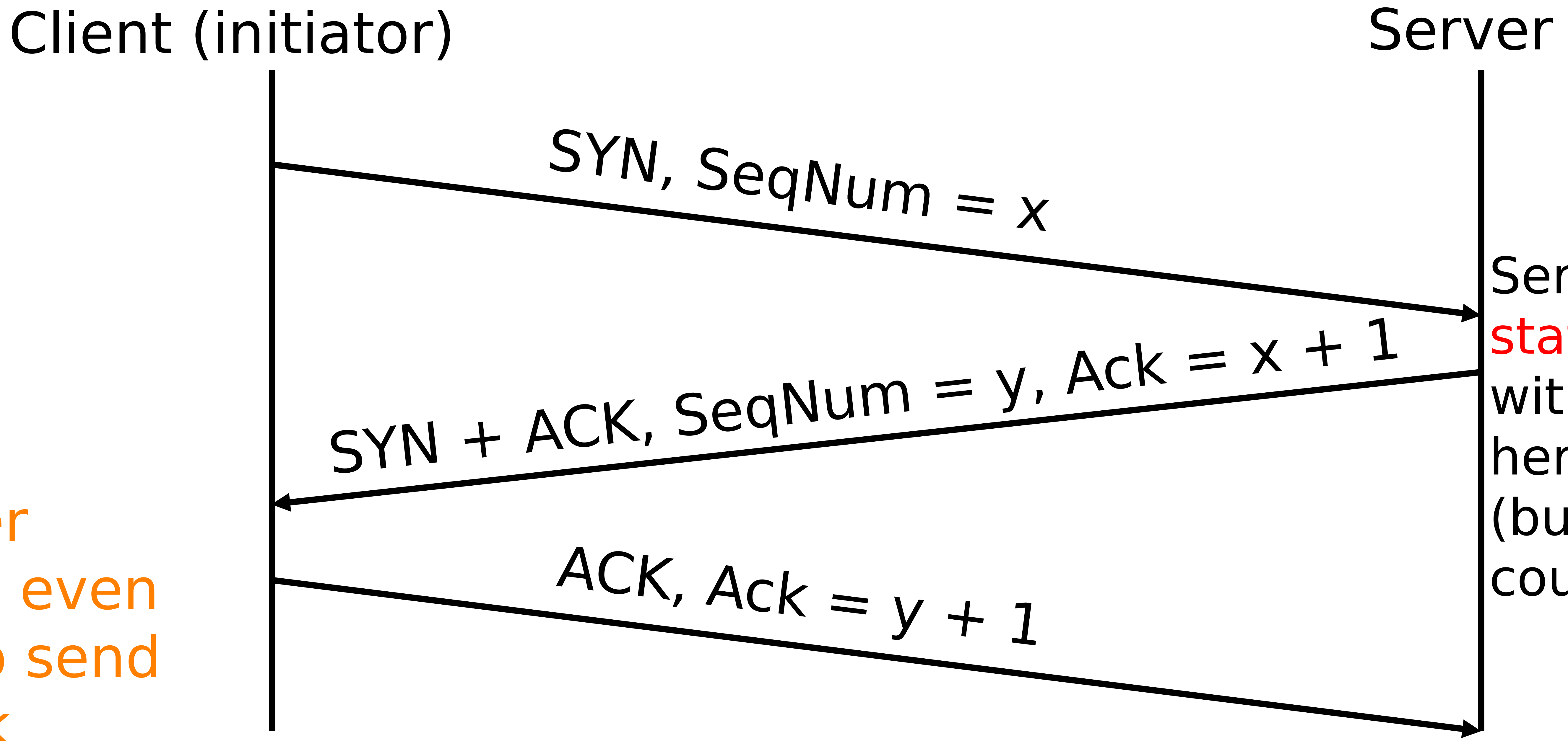
- Example of amplification: DNS lookups
  - Reply is generally much bigger than request
    - Since it includes a copy of the reply, plus answers etc.
  - Attacker spoofs DNS request to a patsy DNS server, seemingly from the target
    - Small attacker packet yields large flooding packet
    - Doesn't increase # of packets, but total volume
- Note #1: these examples involve blind spoofing
  - So for network-layer flooding, generally only works for UDP-based protocols (can't establish a TCP connection)
- Note #2: victim doesn't see spoofed source addresses
  - Addresses are those of actual intermediary systems

Botnets

- If an attacker can control a ***lot*** of systems
  - They gain a huge amount of bandwidth
    - Modern DOS attacks approach 1 Terabit-per-second with direct connections
  - And it becomes very hard to filter them out
    - How do you specify 1M machines you want to ignore?
- You control these "bots" in a "botnet"
  - So you can issue commands that cause all these systems to do what you want
- This is what took down dyn DNS (and with it Twitter, Reddit, etc...) two years ago: A botnet composed primarily of compromised cameras and DVRs:
  - The Miraj botnet

Transport-Level Denial-of-Service

- Recall TCP's 3-way connection establishment handshake
  - Goal: agree on initial sequence numbers

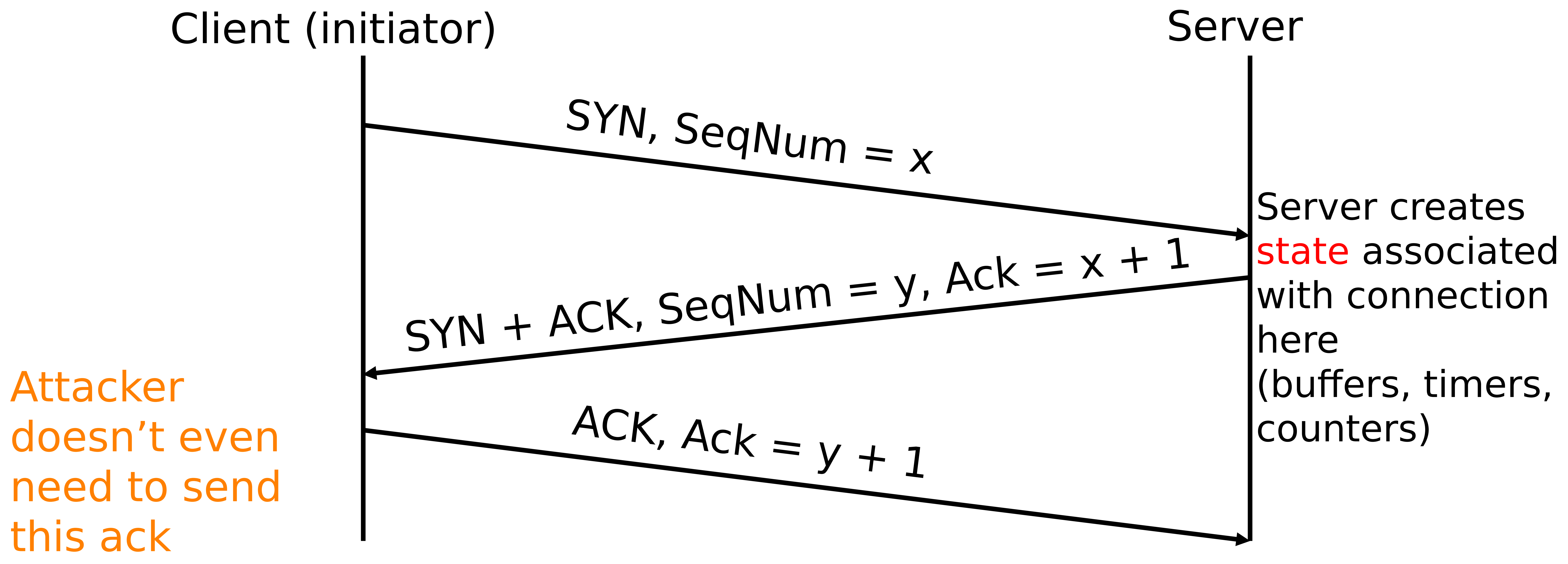


Server creates state associated with connection here (buffers, timers, counters)

Attacker doesn't even need to send this ack

Transport-Level Denial-of-Service

- Recall TCP's 3-way connection establishment handshake
  - Goal: agree on initial sequence numbers
- So a single SYN from an attacker suffices to force the server to spend some memory



Server creates state associated with connection here (buffers, timers, counters)

Attacker doesn't even need to send this ack

TCP SYN Flooding

- Attacker targets memory rather than network capacity
- Every (unique) SYN that the attacker sends burdens the target
- What should target do when it has no more memory for a new connection?
- No good answer!
  - Refuse new connection?
    - Legit new users can’t access service
  - Evict old connections to make room?
    - Legit old users get kicked off

TCP SYN Flooding Defenses

- How can the target defend itself?

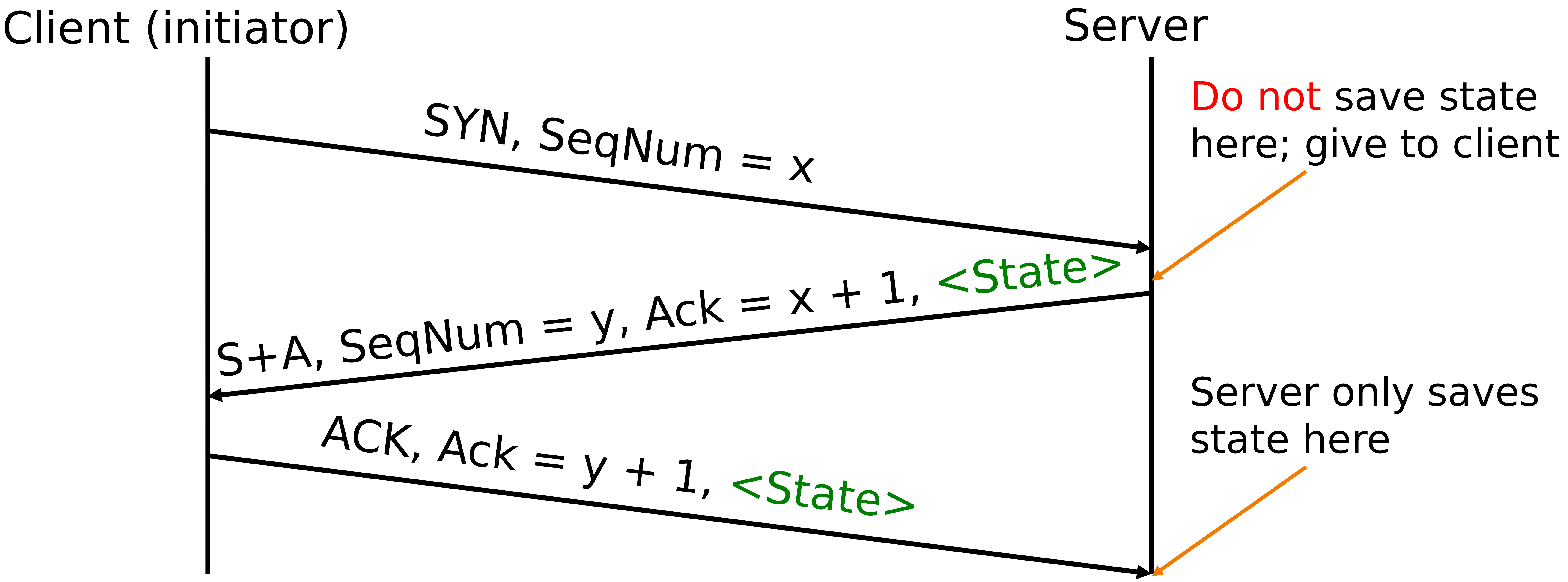
- Approach #1: make sure they have tons of memory!
  - How much is enough?
  - Depends on resources attacker can bring to bear (threat model), which might be hard to know
- Back of the envelope:
  - If we need to hold 10kB for 1 minute: to exhaust 1GB, an attacker needs…
    - 100k packets/minute, or a bit more than 1,000 packets per second

TCP SYN Flooding Defenses

- Approach #2: identify bad actors & refuse their connections
  - Hard because only way to identify them is based on IP address
    - We can’t for example require them to send a password because doing so requires we have an established connection!
  - For a public Internet service, who knows which addresses customers might come from?
  - Plus: attacker can spoof addresses since they don’t need to complete TCP 3-way handshake
- Approach #3: don’t keep state! (“SYN cookies”; only works for spoofed SYN flooding)

SYN Flooding Defense: Idealized

- Server: when SYN arrives, rather than keeping state locally, send it to the client …
- Client needs to return the state in order to established connection


Client (initiator)
Server
SYN, SeqNum = x
S+A, SeqNum = y, Ack = x + 1, <State>
ACK, Ack = y + 1, <State>
Do not save state here; give to client
Server only saves state here

SYN Flooding Defense: Idealized

- Server: when SYN arrives, rather than keeping state locally, send i
- Client needs t stablished connection

Client (

t save state
give to client

r only saves
here

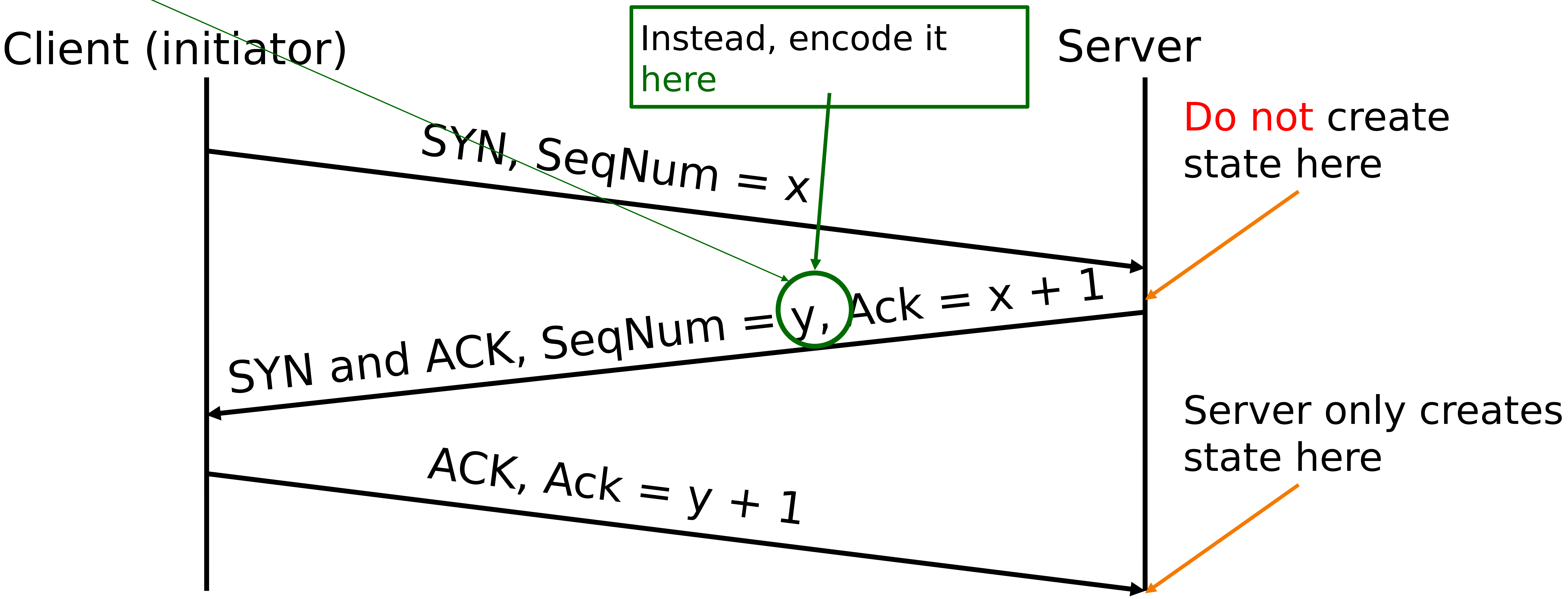
Problem: the world isn’t so ideal!

TCP doesn’t include an easy way to add a new <State> field like this.

Is there any way to get the same functionality without having to change TCP clients?

Practical Defense: SYN Cookies

- Server: when SYN arrives, encode connection state entirely within SYN-ACK's sequence # y
  - y = encoding of necessary state, using server secret
- When ACK of SYN-ACK arrives, server only creates state if value of y from it agrees w/ secret



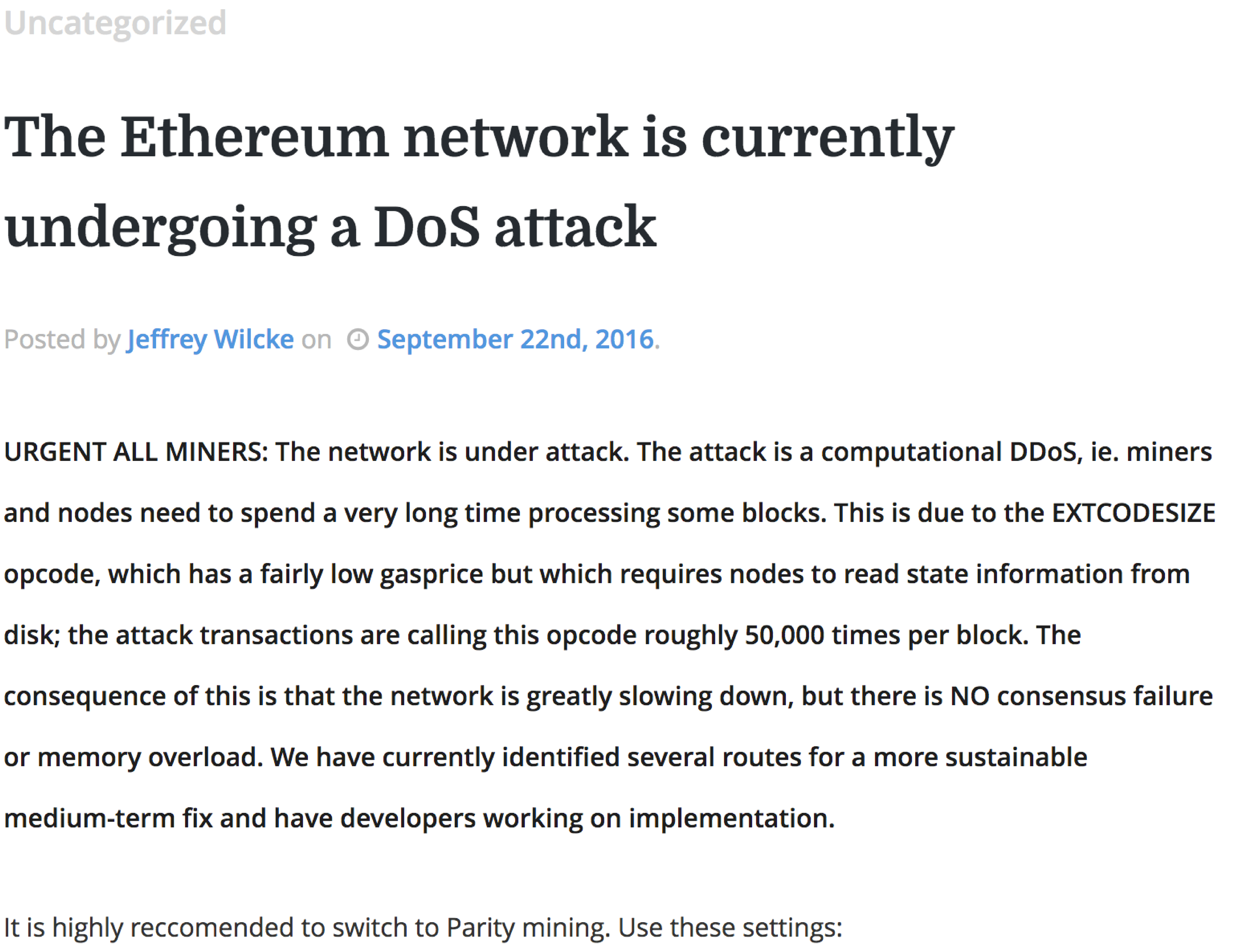
SYN Cookies: Discussion

- Illustrates general strategy: rather than holding state, encode it so that it is returned when needed
- For SYN cookies, attacker must complete 3-way handshake in order to burden server
  - Can’t use spoofed source addresses
- Note #1: strategy requires that you have enough bits to encode all the state
  - (This is just barely the case for SYN cookies)
  - You can think of a SYN cookie as a truncated MAC of the sender IP/port/sequence
- Note #2: if it’s expensive to generate or check the cookie, then it’s not a win

Application-Layer DoS

- Rather than exhausting network or memory resources, attacker can overwhelm a service’s processing capacity

- There are many ways to do so, often at little expense to attacker compared to target (asymmetry)



Uncategorized

The Ethereum network is currently undergoing a DoS attack

Posted by Jeffrey Wilcke on September 22nd, 2016.

URGENT ALL MINERS: The network is under attack. The attack is a computational DDoS, ie. miners and nodes need to spend a very long time processing some blocks. This is due to the EXTCODESIZE opcode, which has a fairly low gasprice but which requires nodes to read state information from disk; the attack transactions are calling this opcode roughly 50,000 times per block. The consequence of this is that the network is greatly slowing down, but there is NO consensus failure or memory overload. We have currently identified several routes for a more sustainable medium-term fix and have developers working on implementation.

It is highly reccomended to switch to Parity mining. Use these settings:

Algorithmic complexity attacks

- Attacker can try to trigger worst-case complexity of algorithms / data structures
- Example: You have a hash table.
  Expected time: O(1). Worst-case: O(n).
- Attacker picks inputs that cause hash collisions.
  Time per lookup: O(n).
  Total time to do n operations: O(n^2).
- Solution? Use algorithms with good worst-case running time.
  - E.g., using ***b*** bits of HMAC ensures that $P[h_k(x)=h_k(y)] = .5^b$, so hash collisions will be rare.
    - If the attacker doesn't know the key that is

Application-Layer DoS

- Defenses against such attacks?
- Approach #1: Only let legit users issue expensive requests
  - Relies on being able to identify/authenticate them
  - Note: that this itself might be expensive!
- Approach #2: Force legit users to "burn" cash
  - This is what a captcha really is!
- Approach #3: massive over-provisioning ($$$)
  - Or pay for someone else who massively over provisions for everyone:
    A content delivery network

DoS Defense in General Terms

- Defending against program flaws requires:
  - Careful design and coding/testing/review
  - Consideration of behavior of defense mechanisms
    - E.g. buffer overflow detector that when triggered halts execution to prevent code injection ⇒ denial-of-service
- Defending resources from exhaustion can be really hard. Requires:
  - Isolation and scheduling mechanisms
    - Keep adversary’s consumption from affecting others
  - Reliable identification of different users
  - Or just a ton of $$$$