# Attacking HTTPS

EECS 388: Intro to Security

University of Michigan

# 5PM Today in 1500 EECS: Adrienne Porter Felt on HTTPS

**Will give special remarks and Q/A mid-way thru lecture at 5PM if the AV gods smile upon us.**

# Review: Implementation Attacks

## OpenSSL Heartbleed Vulnerability

In April 2014, OpenSSL disclosed a catastrophic bug in their implementation of the TLS Heartbeat Extension

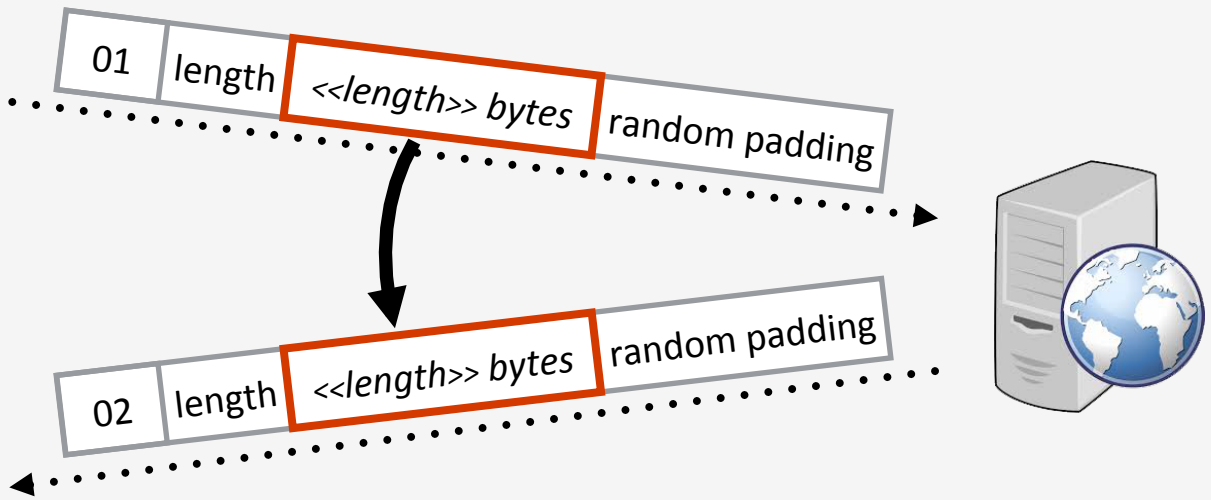
Vulnerability allowed attackers to dump private cryptographic keys, logins, and other private user data

Potentially affected any service that used OpenSSL for TLS—web, mail, messaging and database servers

An estimated **24-55%** of HTTPS websites vulnerable

## Review: Attacking Implementations

# TLS Heartbeat Extension



# Review: Attacking Implementations

**Apple Goto Fail** (Feb. 2014)

Apple SSL libraries skipped certificate checking for almost a year due to stray goto statement

```
    if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;

        err = sslRawVerify(ctx,
                           ctx->peerPubKey,
                           dataToSign,                          /*
                           dataToSignLen,                  /* plaintex
                           signature,
                           signatureLen);
        if(err) {
                sslErrorLog("SSLDecodeSignedServerKeyExchange: sslR
                    "returned %d\n", (int)err);
                goto fail;
        }

fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
```

**Mozilla BERsek** (Oct. 2014)

Bug in verifying cert signatures, allowed spoofing certs, probably since the beginning…!

## Moar: Attacking Implementations

### **Null Prefix Attack**, 2009

(x.509 uses Pascal-style strings, most browsers use C strings;
what if a common name contains "\0"?)

`gmail.com\0.badguy.com`

CA validates `badguy.com` (Pascal string)
Browser saw `gmail.com` (C string)

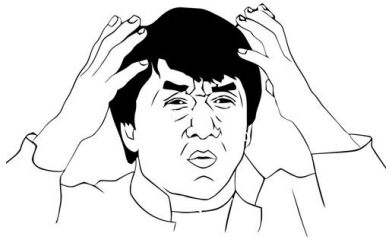
## Moar: Attacking Implementations: Crappy Primes

**Mining Ps and Qs**, USENIX Security 2012

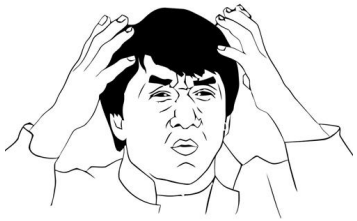
5% of TLS servers and 9% of SSH servers shared key material!

Worse, able to compute private keys for 0.5% of TLS servers and 1% of SSH servers due to inadequate PRNG on servers

WHY???



## Moar: Attacking Implementations: Crappy Primes



**Mining Ps and Qs**, USENIX Security 2012

Why can compute private keys of TLS & SSH servers??

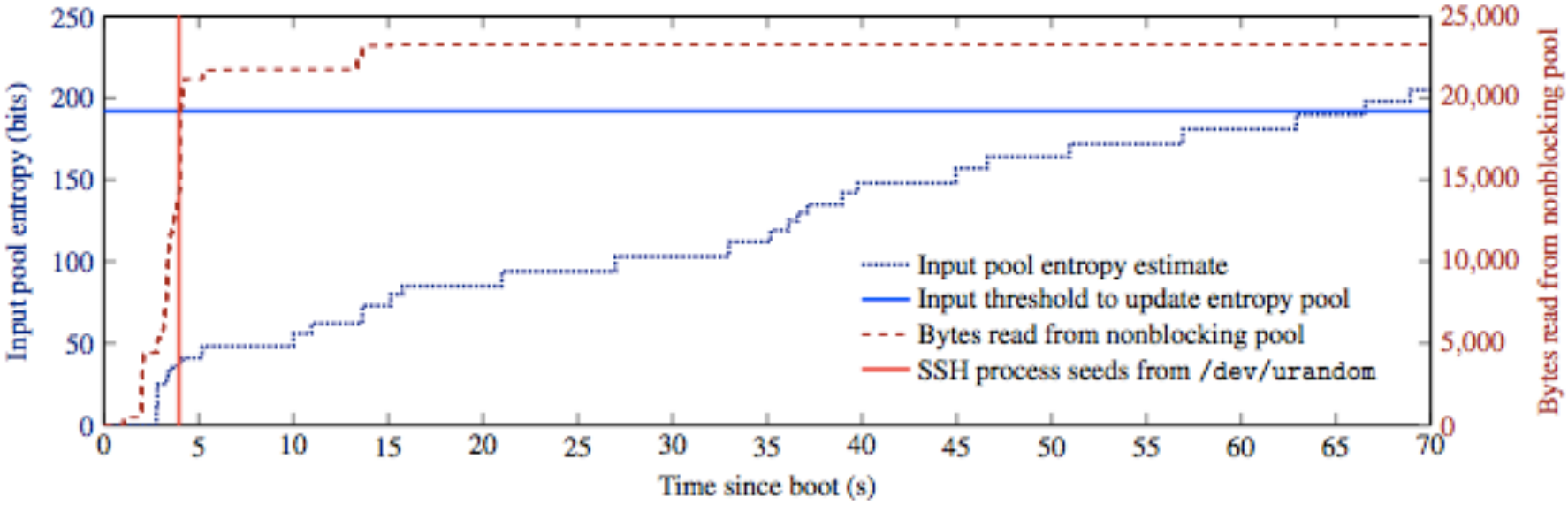
RSA: Public modulus n = p*q

Linux PRNG /dev/urandom is low quality at boot up
Servers tend to create keys when they first boot up…

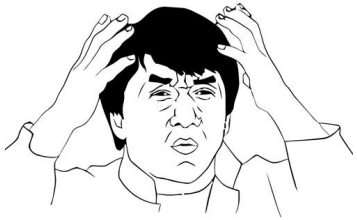
## Moar: Attacking Implementations: Crappy Primes

**Mining Ps and Qs**, USENIX Security 2012

/dev/urandom vs. /dev/random



# GROUP EXERCISE



**Mining Ps and Qs**, USENIX Security 2012

Generate p (PRNG still low entropy)
Do other stuff
Generate q (PRNG finally has enough entropy)

RSA: Public modulus n = p*q,
Server 1: n1 = **p***q1 Server 2: n2 = **p***q2

Uh oh: Can compute p, q efficiently!

# Attacking the TLS Protocol

## RC4-related Attacks

Biases in RC4 stream cipher can be used to leak cookies and passwords.

## CBC-related Attacks

**BEAST** and **POODLE** exploit weaknesses in TLS's use of CBC to leak data.

## Compression-related Attacks

TLS and HTTP both implement compression, but TLS exposes the length of the plaintext. The combination can leak data.

Example: Attacker uses JavaScript to cause user's browser to visit crafted URLs on a target site. If URLs match the user's secret cookie, the length of the TLS data will be shorter. Use this and many requests to iteratively leak the cookie.

## Export-related Attacks

**Netscape PRNG, FREAK**, **Logjam**, and **DROWN** exploit weaknesses in 1990s-era "export-grade" cryptography. Allowed MITM attacks against ~30% of popular modern sites.

## Moar: Attacking Implementations: Crappy Primes I

# Netscape PRNG

Predictable SSL keys based on time of day

Dr. Dobb's Journal, January 1996

https://people.eecs.berkeley.edu/~daw/papers/ddj-netscape.html

Seeded PRNG with time of day, predictable from TCP/IP headers

Encryption keys weakened to just 40 bits (EXPORT)

25 seconds on a calculator to guess SSL key material

# Why Does "EXPORT" Grade Crypto Even Exist?

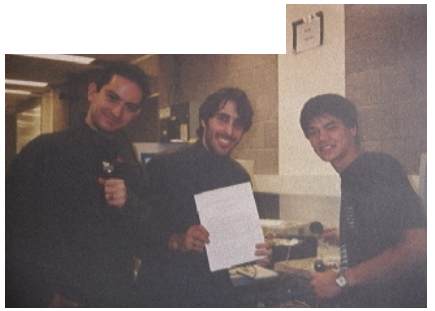

BUSINESS DAY

## *Court Hears Appeal in Encryption Case*

By JOHN MARKOFF DEC. 9, 1997

In a case that could have a profound effect on the future of electronic commerce and banking, a three-judge panel of the Court of Appeals for the Ninth Circuit here heard the Government's appeal of a December 1996 decision in which Judge Marilyn Hall Patel of Federal District Court ruled that Government attempts to control the export of encryption software were unconstitutional.

Her ruling came in a suit filed in February 1995 by Daniel J. Bernstein, then a graduate student at the University of California at Berkeley, after State Department officials said he would be required to register as a munitions dealer and secure an arms-trading license to export an electronic version of a short encryption program he had written called Snuffle.



# The FREAK Attack

Attacker

1. ClientHello:
"I'd like to use RSA…"

2. ClientHello:
"I only support export-grade RSA"

≈9% of popular HTTPS sites still allowed 512-bit export-grade RSA, Apache reuses export RSA key pair until server reboots!

3. "OK, here's my 512-bit temporary RSA key"
— Signed, *Server*

4. Buggy browsers (≈ all but Firefox) accept export-grade key *even though they didn't ask for it,* use to encrypt rest of the session

Typical servers reuse the temporary RSA key for hours or days.
Attacker factors it to get private key (~2 hours on EC2 for $100).

5. Attacker uses factored 512-bit key to steal or alter connection's data

# The Logjam Attack


Similar to FREAK, but targets export-grade Diffie-Hellman instead of RSA.

Protocol flaw rather than browser bug: affected practically any browser.

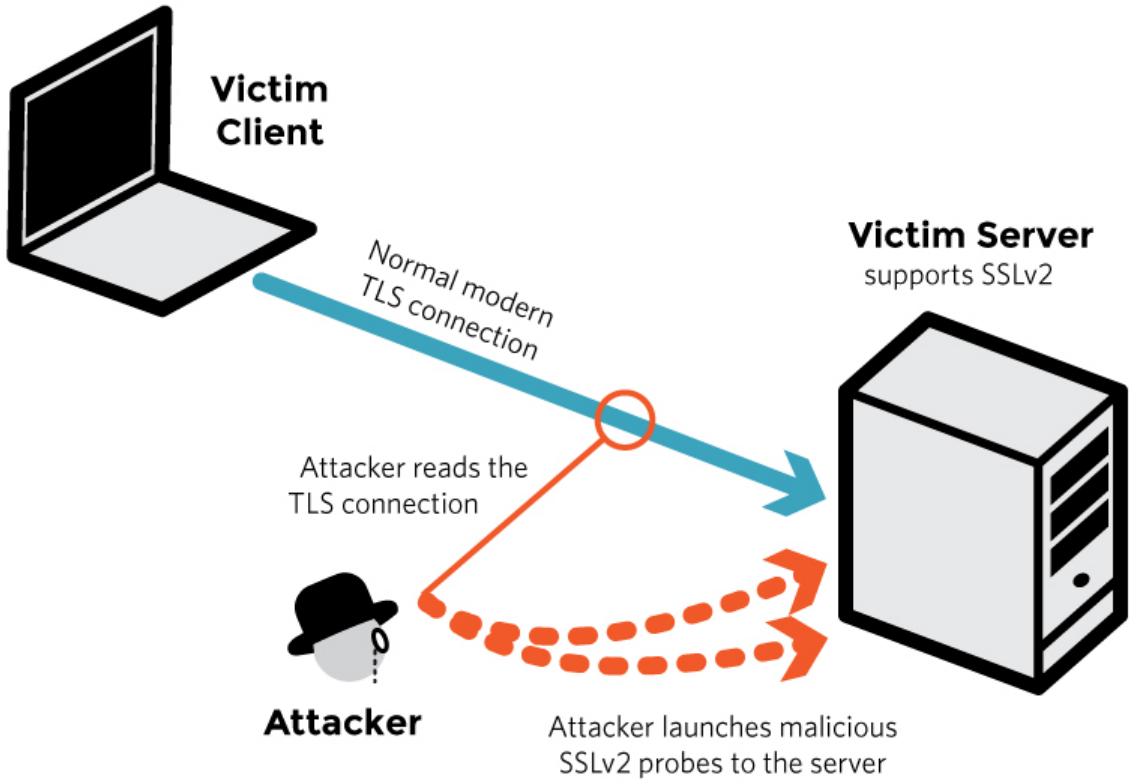
Targets hard-coded 512-bit DH primes. Big up-front computation allows attacker to break every subsequent connection from the server.

≈9% of popular HTTPS sites still allowed 512-bit export-grade Diffie-Hellman

# DROWN Attacking TLS Protocol

## DROWN Attack

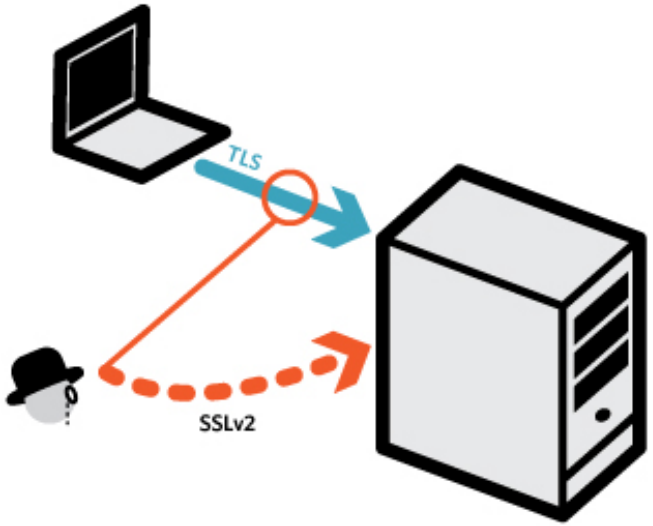
(March 2015) Bleichenbacher padding-oracle attack exploits servers that support obsolete SSL 2.0 to attack connections that use modern TLS.

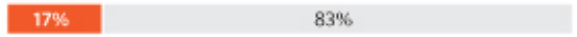


# DROWN: Attacking TLS Protocol

**A server is vulnerable to DROWN if:**

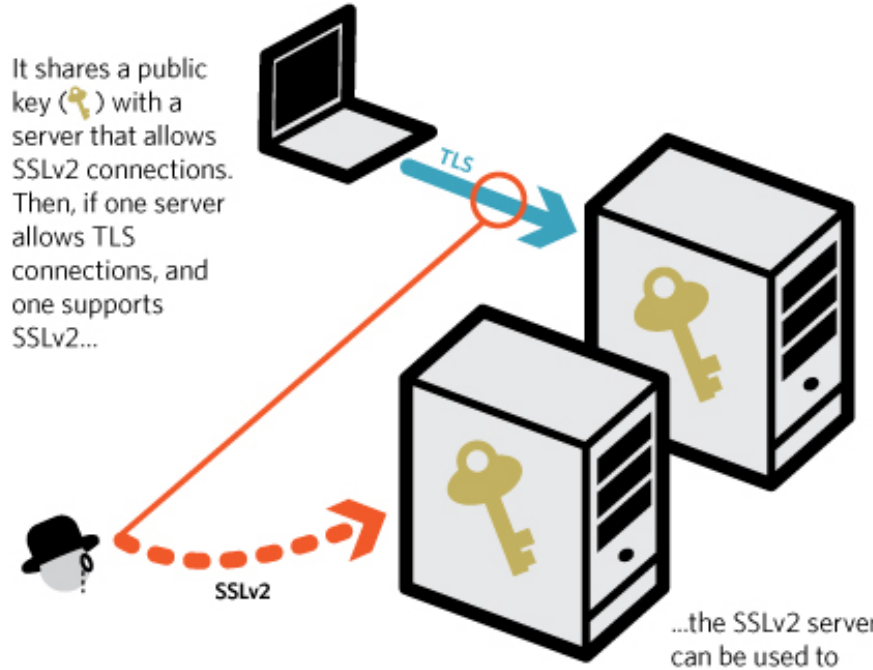
It allows both TLS and SSLv2 connections



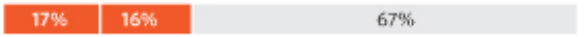




17% of HTTPS servers still allow SSLv2 connections

It shares a public key ( ) with a server that allows SSLv2 connections. Then, if one server allows TLS connections, and one supports SSLv2...




...the SSLv2 server can be used to attack the TLS server




When taking key reuse into account, an additional 16% of HTTPS servers are vulnerable, putting 33% of HTTPS servers at risk

# Where the Wild Warnings Are: Root Causes of Chrome HTTPS Certificate Errors

Mustafa Emre Acer
Google Inc.

Emily Stark
Google Inc.

Adrienne Porter Felt
Google Inc.

Sascha Fahl
Leibniz University Hannover

Radhika Bhargava
Purdue University

Bhanu Dev
International Institute of Information Technology Hyderabad

Matt Braithwaite
Google Inc.

Ryan Sleevi
Google Inc.

Parisa Tabriz
Google Inc.

# Takeaways

**Use HTTPS!** Despite attacks, it's so much better than nothing.

Be careful of bad server defaults and configuration gotchas. Test you sites using tools like SSLLabs.

TLS and the CA ecosystem will keep breaking. Use them, but don't rely on them exclusively, and plan to respond quickly.

Many problems involve fooling users into doing the wrong thing. We'll be safer when HTTPS eventually becomes the default.