The Bad-Attitude Guide to Computer Security

Keith Winstein

Stanford University
https://cs.stanford.edu/~keithw

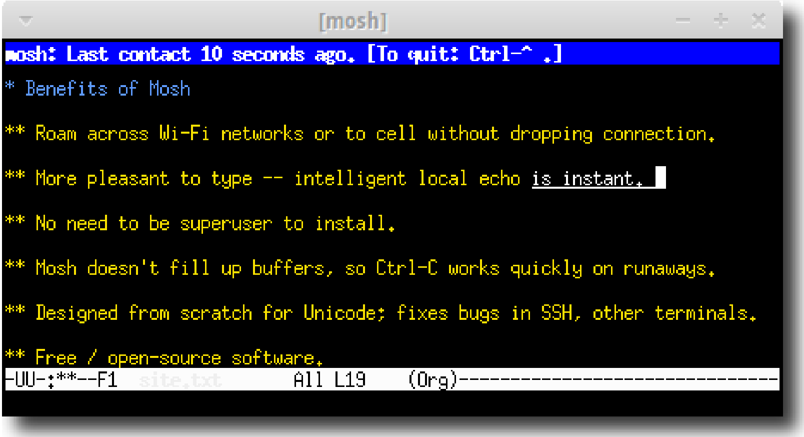
Bad (-attitude?) advice

1.
2.
3.
4.
5.
6.
7.
8.

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2.
3.
4.
5.
6.
7.
8.

Mosh (mobile shell)



```
mosh: Last contact 10 seconds ago. [To quit: Ctrl-^ .]
* Benefits of Mosh

** Roam across Wi-Fi networks or to cell without dropping connection.

** More pleasant to type -- intelligent local echo is instant.

** No need to be superuser to install.

** Mosh doesn't fill up buffers, so Ctrl-C works quickly on runaways.

** Designed from scratch for Unicode; fixes bugs in SSH, other terminals.

** Free / open-source software.
-UU-:**--F1  site.txt        All L19     (Org)-----------------------------
```

Mosh deployment

- Impetus: SSH for bad Wi-Fi
  - + intermittent connectivity
  - + roaming
  - + local echo
  - + security against forged RST
- First release: 2012
- Today: appx. 2–20 million users

Mosh protocol

- Every datagram wrapped with AES-OCB
- Every datagram represents idempotent operation

- **No** TLS, **no** DTLS, **no** public-key crypto
- **No** timestamps, **no** replay cache, **no** daemon
- **No** cipher negotiation, **no** file IO, **no** root

- Roaming: server replies to source address of highest-numbered authentic incoming datagram

Hacker News

“One man band by the looks of it. . . Implements its own private crypto protocol (has it been vetted for replay attacks? padding attacks? [insert 20 years of perplexing bugs confounding the greatest minds in computer science]?)”

Slashdot

“Welcome to Yet Another Protocol Devised By Academics Who Have Not Been Near a Real Network in Twenty Years, If Ever.”

Twitter

Dan Kaminsky: "Mosh being outside of SSH Transport makes academic perf code unauthenticated. . . Love MoSH, would love it much more if it operated inside SSH's channel"

Q: "any particular reason? Since quick recovery from packetloss is one of its main goals, UDP+OCB is needed."

Kaminsky: "It's *tricky* to build new secure channels. Look at DTLS's long and painful dev cycle."

Twitter (cont.)

Moxie Marlinspike: "I dunno, from a semantic sec perspective, it'd be hard to do worse than SSH. It's in many ways worse than TLS."

Kaminsky: "Do you suspect it has BEAST-style bugs waiting to be found?"

Moxie: "Already found. The CBC ciphersuites are totally off limits now because of chosen *ciphertext* attacks. Much wrse. . . Just like TLS. Bad protocol that keeps squeaking by in some circumstances. Slowly painting itself into a corner."

Security holes in Mosh's lifetime

TLS:

- `goto fail` (Secure Transport)
- GnuTLS verify (GnuTLS)
- Heartbleed (OpenSSL)
- Lucky Thirteen
- BEAST
- CRIME
- POODLE
- FREAK
- Logjam
- 2013 RC4 attacks

SSH:

- memory corruption attack
- X11 trust race condition
- weak tty permissions
- password limit circumvention
- root password auth bug
- unfinished roaming feature allows private key extraction
- command injection to xauth

Mosh:

- (no security holes that we know about, so far)

The lesson

- Committees are the worst.
- Small projects have a huge advantage.
- Bugs are caused by **features**. Fewer features, fewer bugs.
- Feynman-ish dictum: *You're not as good as the best contributor to a big project, but you're probably better than the average contributor.*
- For security, the **worst** contributor may be what matters.

Advice that I really (mostly) believe

After 20 years of committee design, SSL/TLS and its implementations are so hairy and so buggy that for a particular focused task, “doing your own” may sometimes be the more reasonable path, even if TLS would do the job.

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2.
3.
4.
5.
6.
7.
8.

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2.
3.
4.
5.
6.
7.
8.

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3.
4.
5.
6.
7.
8.

Hacker News

“Anyone doing security work in C in 2016 is in my opinion committing malpractice and putting user’s at risk because their ego’s can’t take not fiddling bits by hand.”

Security holes prevented by memory safety

TLS:

- `goto fail` (Secure Transport)
- GnuTLS verify (GnuTLS)
- Heartbleed (OpenSSL)
- Lucky Thirteen
- BEAST
- CRIME
- POODLE
- FREAK
- Logjam
- 2013 RC4 attacks

SSH:

- memory corruption attack
- X11 trust race condition
- weak tty permissions
- password limit circumvention
- root password auth bug
- roaming stub allows private key extraction
- command injection to xauth

Security holes prevented by memory safety

TLS:

- `goto fail` (Secure Transport) **not prevented**
- GnuTLS verify (GnuTLS) **not prevented**
- Heartbleed (OpenSSL) **prevented**
- Lucky Thirteen **not prevented**
- BEAST **not prevented**
- CRIME **not prevented**
- POODLE **not prevented**
- FREAK **not prevented**
- Logjam **not prevented**
- 2013 RC4 attacks **not prevented**

SSH:

- memory corruption attack **prevented**
- X11 trust race condition **not prevented**
- weak tty permissions **not prevented**
- password limit circumvention **not prevented**
- root password auth bug **not prevented**
- roaming stub allows private key extraction **prevented**
- command injection to xauth **not prevented**

Popular memory-safe languages are way too powerful.

- What you say: “memory-safe”
- What you mean: “Haskell”
- What people hear: “JavaScript, Python, or Ruby on Rails”

- Any language with `eval` is apparently very tempted to use it.
- Java security track record is also not great.

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3.
4.
5.
6.
7.
8.

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3.
4.
5.
6.
7.
8.

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3. HTTPS is bad.
4.
5.
6.
7.
8.

HTTPS's sacred promise

HTTPS’s sacred promise

There exists a company from a list of 173 companies, and that company at one point attested that the `WHOIS` contact email for the domain name in my URL bar belonged to the same person who now controls the server I’m talking to (unless its cert has been stolen).

HTTPS's promise is pretty lame.

- Really should be verifying the file author, not just identity of the server maintainer.
- After download, no way to authenticate a file, no record of who attested to server's identity, and no proof.
- Breaks even voluntary caching and virus-scanning.
- Guarantee provided by *any* of 173 semi-savory companies.

Pinning to the rescue

Browser Vendor: Use an HSTS header to pin a *particular* server cert!

Site: That sounds like TOFU. What if a CA issues an evil cert before the user first visits us?

Vendor: Ask Google, Apple, Microsoft, and Mozilla to hardcode your server cert directly into the browser!

Pinning II

Site: We did it. Except, some of our users are behind a firewall that seems to still be MITMing their sessions.

Vendor: The pin is only honored for "public" root CAs. If the user installs a "private" root CA, that will override the pin, even if you get your cert hardcoded into the browser.

Site: Why would the browser *knowingly* allow a man-in-the-middle attack?

Vendor: Lots of companies use virus-scanning middleboxes, and the only way to do that with HTTPS is to completely MITM the sessions. Honoring the pin may be the right thing, but our browser will be perceived as broken and we'd lose market share. We'd only do this after our competitors had already done it.

Pinning III

Site: When disregarding the pin, at least display a broken padlock and a warning: "Your session is being eavesdropped on by a private authority. Click here to disable."

Vendor: If we did that, we'd have to display the warning for all eternity because those resources will land in the cache and permanently corrupt it. So it wouldn't be a very useful warning.

Site: If the cache is "permanently corrupted," you *should* display a warning for eternity!

Vendor: Then our browser would be perceived as broken and we'll lose market share. We'd only do this after the other vendors have.

Site: Can you at least make this an option for paranoid people?

Vendor: lol no

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3. HTTPS is bad.
4.
5.
6.
7.
8.

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3. HTTPS is bad.
4.
5.
6.
7.
8.

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3. HTTPS is bad.
4. Password hashing is bad.
5.
6.
7.
8.

Password hashing is bad.

- Password hashing is bad because it makes you think it's okay for users to send you their passwords.
- Users should not send you their passwords.
- Your site security should not depend on your enforcement of a password complexity requirement.
- You do not want a server compromise to expose anything that allows a bad guy to intercept or crack user passwords.
- **Better:** Delegate. Use public-key auth, or "Log in with Google" / "Log in with Facebook" / OpenID Connect.

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3. HTTPS is bad.
4. Password hashing is bad.
5.
6.
7.
8.

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3. HTTPS is bad.
4. Password hashing is bad.
5.
6.
7.
8.

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3. HTTPS is bad.
4. Password hashing is bad.
5. Forward secrecy is usually secret, and that’s bad.
6.
7.
8.

Forward secrecy defined

Forward secrecy

In a communication between multiple parties, there exists a "ratchet" at time t and a later "deletion event" at time u.

Forward secrecy: An exposure of secret material by a party after time u will not aid an eavesdropper in decoding ciphertext encoded before time t.

(Colloquially, t might be the end of a session, and u is when all parties have erased the key or anything that can derive the key that protected the session.)

The problem: no way to communicate when *u* occurs

- Websites are supposed to erase their key cache once a day.

- How can a client learn if this has happened? No way to ask.

- TLS 1.3 draft includes async key rotation, but no authenticated way to acknowledge the message.

- **My view:** if you care about PFS, you should want **authenticated** PFS. Any operation worth doing is worth confirming, including key rotation.

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3. HTTPS is bad.
4. Password hashing is bad.
5. Forward secrecy is usually secret, and that’s bad.
6.
7.
8.

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3. HTTPS is bad.
4. Password hashing is bad.
5. Forward secrecy is usually secret, and that’s bad.
6.
7.
8.

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3. HTTPS is bad.
4. Password hashing is bad.
5. Forward secrecy is usually secret, and that’s bad.
6. End-to-end security is bad, and key escrow is good.
7.
8.

End-to-end security is bad.

- I used to believe in end-to-end security.
- But, I used to think of myself as the endpoint.
- Now I own endpoints that communicate securely with their maker over Wi-Fi or their own LTE.
- **I deserve the right to listen in on what my own devices are saying about me.**

End-to-end security is bad (cont.).

- Today, end-to-end security means the endpoint is the *only* thing that can defend itself.
- Hard to provide defense-in-depth or even detect attacks if you can only see ciphertext.
- Every cheap device is a single point of failure.
- Manufacturers will not keep up with security patches for a $10 device.

Proposed research agenda

How can we build a firewall and auditor for encrypted communications in order to leave no single point of failure?

(Ex. approaches: Blindbox, delayed key release, read-only keys)

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3. HTTPS is bad.
4. Password hashing is bad.
5. Forward secrecy is usually secret, and that’s bad.
6. End-to-end security is bad, and key escrow is good.
7.
8.

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3. HTTPS is bad.
4. Password hashing is bad.
5. Forward secrecy is usually secret, and that’s bad.
6. End-to-end security is bad, and key escrow is good.
7.
8.

Bad (-attitude?) advice

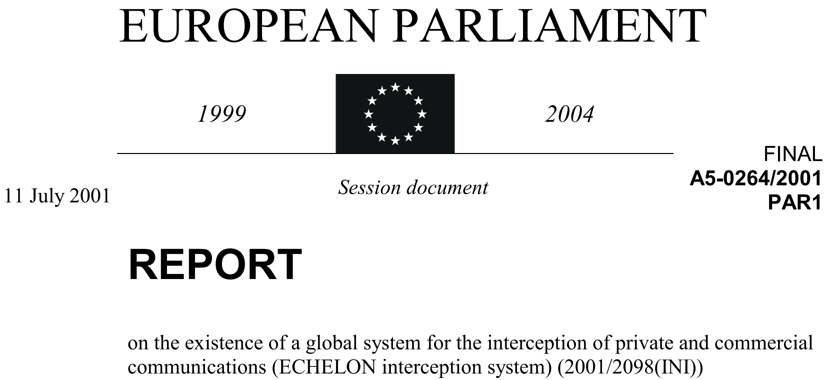
1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3. HTTPS is bad.
4. Password hashing is bad.
5. Forward secrecy is usually secret, and that’s bad.
6. End-to-end security is bad, and key escrow is good.
7. The Snowden docs shouldn’t have changed our behavior.
8.

GNU emacs, May 1987

```
;;; spook.el --- spook phrase utility for overloading the NSA line eater

;;  Just before sending mail, do M-x spook.
;;  A number of phrases will be inserted into your buffer, to help
;;  give your message that extra bit of attractiveness for automated
;;  keyword scanners.  Help defeat the NSA trunk trawler!
```

European Union, July 2001



EUROPEAN PARLIAMENT

1999 *2004*

FINAL
A5-0264/2001
PAR1

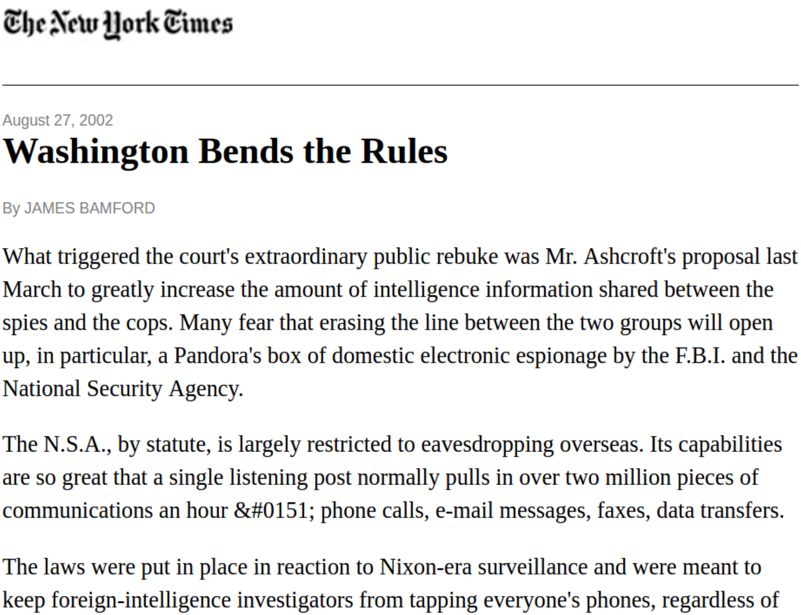
11 July 2001

Session document

REPORT

on the existence of a global system for the interception of private and commercial communications (ECHELON interception system) (2001/2098(INI))

Bush admin. wins first-ever FISA appeal, 2002



The New York Times

August 27, 2002

Washington Bends the Rules

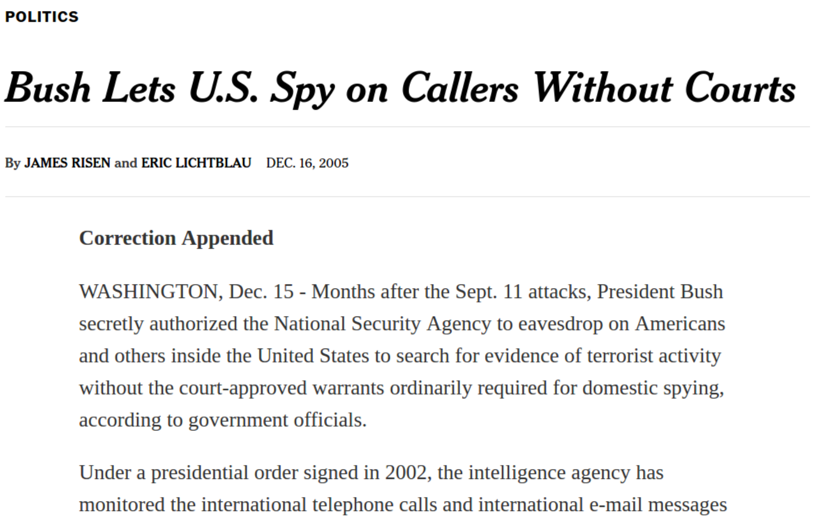
By JAMES BAMFORD

What triggered the court's extraordinary public rebuke was Mr. Ashcroft's proposal last March to greatly increase the amount of intelligence information shared between the spies and the cops. Many fear that erasing the line between the two groups will open up, in particular, a Pandora's box of domestic electronic espionage by the F.B.I. and the National Security Agency.

The N.S.A., by statute, is largely restricted to eavesdropping overseas. Its capabilities are so great that a single listening post normally pulls in over two million pieces of communications an hour &#0151; phone calls, e-mail messages, faxes, data transfers.

The laws were put in place in reaction to Nixon-era surveillance and were meant to keep foreign-intelligence investigators from tapping everyone's phones, regardless of

New York Times, December 2005



POLITICS

Bush Lets U.S. Spy on Callers Without Courts

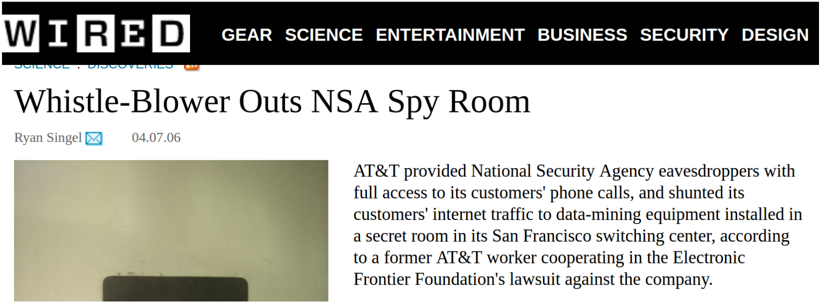
By JAMES RISEN and ERIC LICHTBLAU DEC. 16, 2005

Correction Appended

WASHINGTON, Dec. 15 - Months after the Sept. 11 attacks, President Bush secretly authorized the National Security Agency to eavesdrop on Americans and others inside the United States to search for evidence of terrorist activity without the court-approved warrants ordinarily required for domestic spying, according to government officials.

Under a presidential order signed in 2002, the intelligence agency has monitored the international telephone calls and international e-mail messages

AT&T whistleblower, April 2006

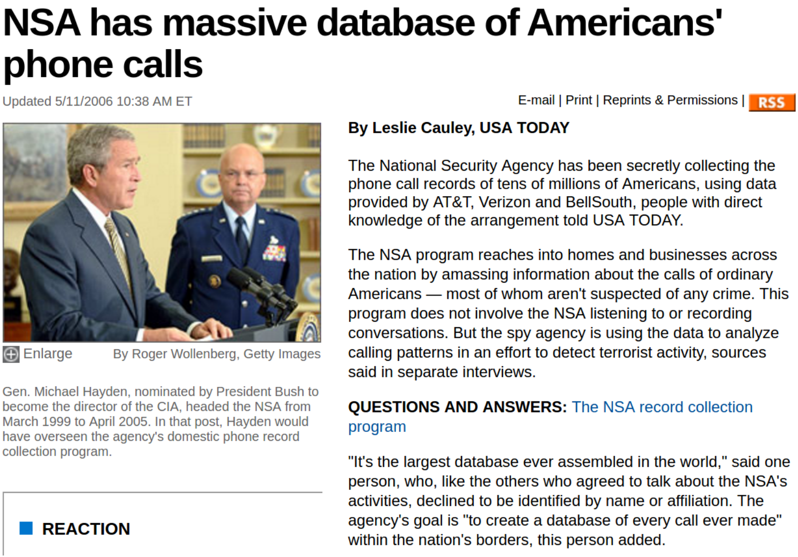


Whistle-Blower Outs NSA Spy Room

Ryan Singel 04.07.06

AT&T provided National Security Agency eavesdroppers with full access to its customers' phone calls, and shunted its customers' internet traffic to data-mining equipment installed in a secret room in its San Francisco switching center, according to a former AT&T worker cooperating in the Electronic Frontier Foundation's lawsuit against the company.

USA Today, May 2006



NSA has massive database of Americans' phone calls

Updated 5/11/2006 10:38 AM ET

By Roger Wollenberg, Getty Images

Gen. Michael Hayden, nominated by President Bush to become the director of the CIA, headed the NSA from March 1999 to April 2005. In that post, Hayden would have overseen the agency's domestic phone record collection program.

REACTION

By Leslie Cauley, USA TODAY

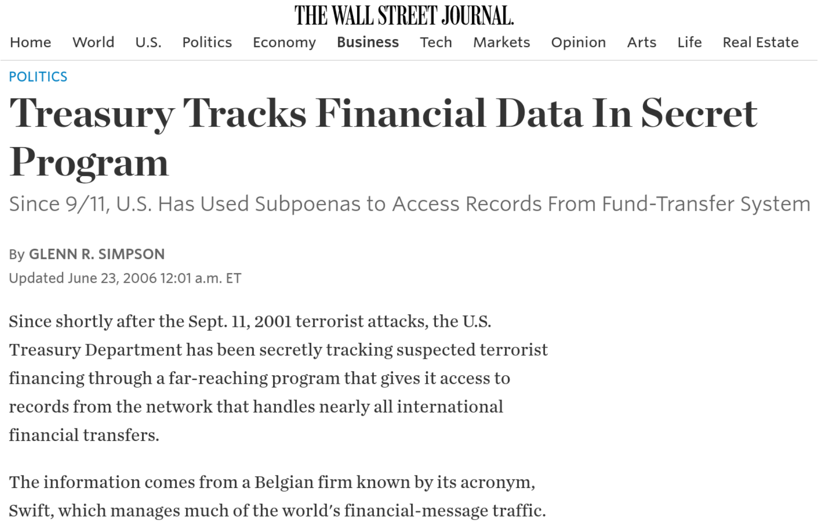
The National Security Agency has been secretly collecting the phone call records of tens of millions of Americans, using data provided by AT&T, Verizon and BellSouth, people with direct knowledge of the arrangement told USA TODAY.

The NSA program reaches into homes and businesses across the nation by amassing information about the calls of ordinary Americans — most of whom aren't suspected of any crime. This program does not involve the NSA listening to or recording conversations. But the spy agency is using the data to analyze calling patterns in an effort to detect terrorist activity, sources said in separate interviews.

QUESTIONS AND ANSWERS: The NSA record collection program

"It's the largest database ever assembled in the world," said one person, who, like the others who agreed to talk about the NSA's activities, declined to be identified by name or affiliation. The agency's goal is "to create a database of every call ever made" within the nation's borders, this person added.

Wall Street Journal, June 2006



Treasury Tracks Financial Data In Secret Program

Since 9/11, U.S. Has Used Subpoenas to Access Records From Fund-Transfer System

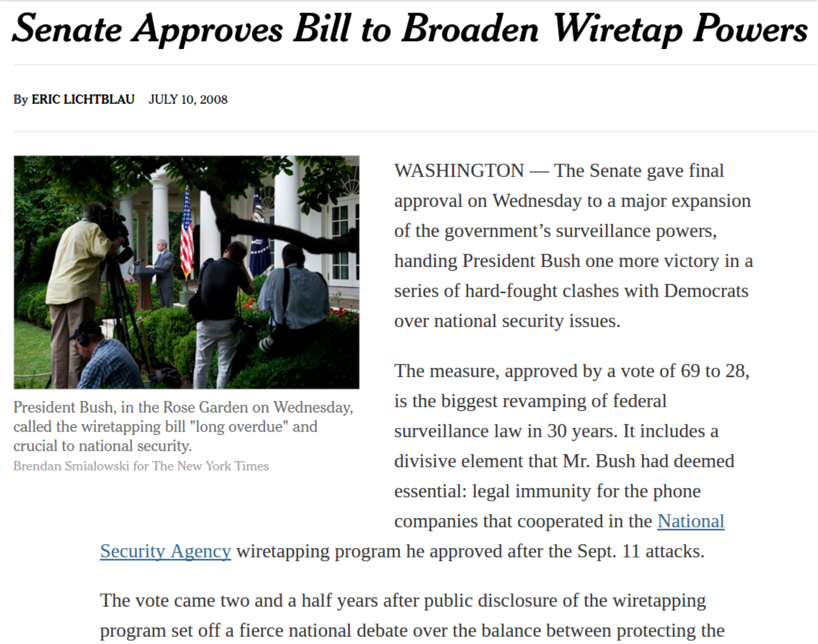
By **GLENN R. SIMPSON**

Updated June 23, 2006 12:01 a.m. ET

Since shortly after the Sept. 11, 2001 terrorist attacks, the U.S. Treasury Department has been secretly tracking suspected terrorist financing through a far-reaching program that gives it access to records from the network that handles nearly all international financial transfers.

The information comes from a Belgian firm known by its acronym, Swift, which manages much of the world's financial-message traffic.

FISA Amendments Act, July 2008



Senate Approves Bill to Broaden Wiretap Powers

By **ERIC LICHTBLAU** JULY 10, 2008

President Bush, in the Rose Garden on Wednesday, called the wiretapping bill "long overdue" and crucial to national security.

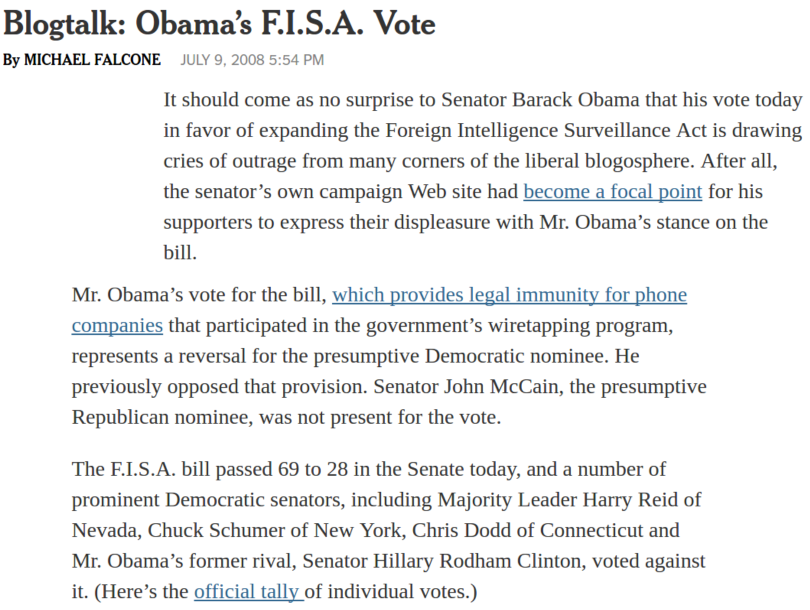
Brendan Smialowski for The New York Times

WASHINGTON — The Senate gave final approval on Wednesday to a major expansion of the government's surveillance powers, handing President Bush one more victory in a series of hard-fought clashes with Democrats over national security issues.

The measure, approved by a vote of 69 to 28, is the biggest revamping of federal surveillance law in 30 years. It includes a divisive element that Mr. Bush had deemed essential: legal immunity for the phone companies that cooperated in the National Security Agency wiretapping program he approved after the Sept. 11 attacks.

The vote came two and a half years after public disclosure of the wiretapping program set off a fierce national debate over the balance between protecting the

Obama supported FAA, Hillary Clinton opposed



Blogtalk: Obama's F.I.S.A. Vote

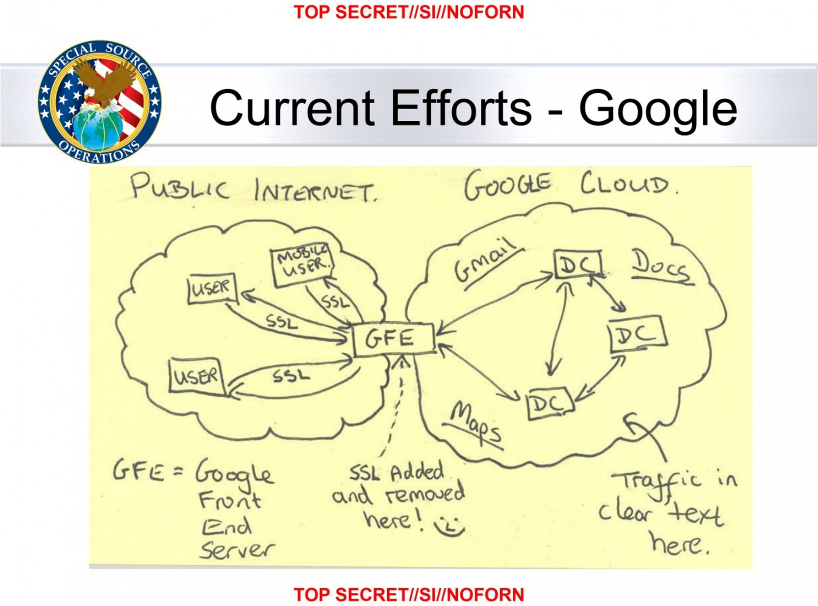
By MICHAEL FALCONE JULY 9, 2008 5:54 PM

> It should come as no surprise to Senator Barack Obama that his vote today in favor of expanding the Foreign Intelligence Surveillance Act is drawing cries of outrage from many corners of the liberal blogosphere. After all, the senator's own campaign Web site had become a focal point for his supporters to express their displeasure with Mr. Obama's stance on the bill.

Mr. Obama's vote for the bill, which provides legal immunity for phone companies that participated in the government's wiretapping program, represents a reversal for the presumptive Democratic nominee. He previously opposed that provision. Senator John McCain, the presumptive Republican nominee, was not present for the vote.

The F.I.S.A. bill passed 69 to 28 in the Senate today, and a number of prominent Democratic senators, including Majority Leader Harry Reid of Nevada, Chuck Schumer of New York, Chris Dodd of Connecticut and Mr. Obama's former rival, Senator Hillary Rodham Clinton, voted against it. (Here's the official tally of individual votes.)

Snowden documents, October 2013



TOP SECRET//SI//NOFORN

SPECIAL SOURCE OPERATIONS

Current Efforts - Google

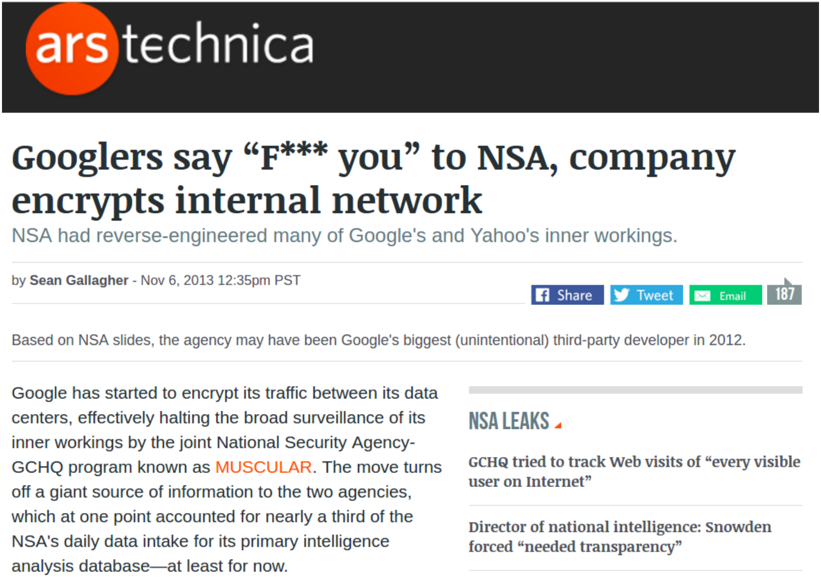
GFE = Google Front End Server

SSL Added and removed here! :)

Traffic in clear text here.

TOP SECRET//SI//NOFORN

“Google has started to encrypt,” November 2013



Googlers say “F*** you” to NSA, company encrypts internal network

NSA had reverse-engineered many of Google's and Yahoo's inner workings.

by **Sean Gallagher** - Nov 6, 2013 12:35pm PST

Based on NSA slides, the agency may have been Google's biggest (unintentional) third-party developer in 2012.

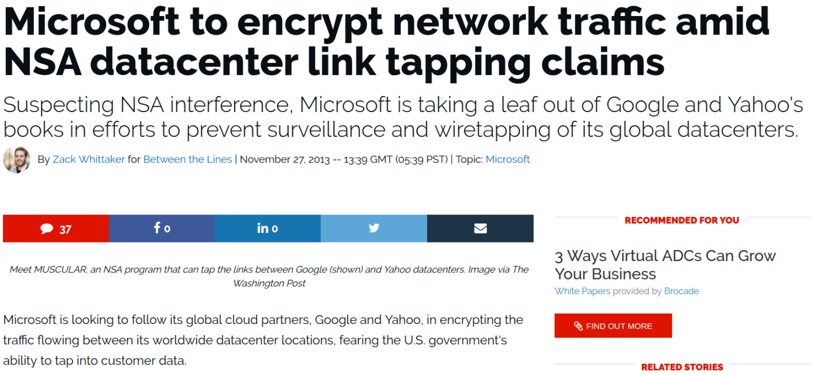
Google has started to encrypt its traffic between its data centers, effectively halting the broad surveillance of its inner workings by the joint National Security Agency-GCHQ program known as MUSCULAR. The move turns off a giant source of information to the two agencies, which at one point accounted for nearly a third of the NSA's daily data intake for its primary intelligence analysis database—at least for now.

NSA LEAKS

GCHQ tried to track Web visits of “every visible user on Internet”

Director of national intelligence: Snowden forced “needed transparency”

Microsoft looking to encrypt, November 2013



Microsoft to encrypt network traffic amid NSA datacenter link tapping claims

Suspecting NSA interference, Microsoft is taking a leaf out of Google and Yahoo's books in efforts to prevent surveillance and wiretapping of its global datacenters.

By Zack Whittaker for Between the Lines | November 27, 2013 -- 13:39 GMT (05:39 PST) | Topic: Microsoft

Meet MUSCULAR, an NSA program that can tap the links between Google (shown) and Yahoo datacenters. Image via The Washington Post

Microsoft is looking to follow its global cloud partners, Google and Yahoo, in encrypting the traffic flowing between its worldwide datacenter locations, fearing the U.S. government's ability to tap into customer data.

RELATED STORIES

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2. "Safe" languages aren't *that* safe.
3. HTTPS is bad.
4. Password hashing is bad.
5. Forward secrecy is usually secret, and that's bad.
6. End-to-end security is bad, and key escrow is good.
7. The Snowden docs shouldn't have changed our behavior.
8.

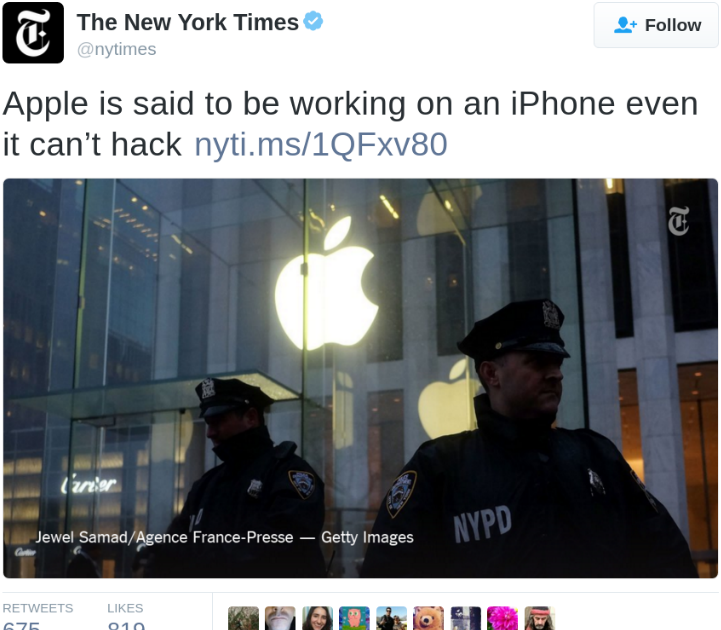
Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3. HTTPS is bad.
4. Password hashing is bad.
5. Forward secrecy is usually secret, and that’s bad.
6. End-to-end security is bad, and key escrow is good.
7. The Snowden docs shouldn’t have changed our behavior.
8.

Bad (-attitude?) advice

1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3. HTTPS is bad.
4. Password hashing is bad.
5. Forward secrecy is usually secret, and that’s bad.
6. End-to-end security is bad, and key escrow is good.
7. The Snowden docs shouldn’t have changed our behavior.
8. Auto-updating without consent is bad, because we might become bad.

nytimes.com, February 2016



The New York Times
@nytimes

Apple is said to be working on an iPhone even it can't hack nyti.ms/1QFxv80

Jewel Samad/Agence France-Presse — Getty Images

RETWEETS 675

LIKES 819

This is not something to be congratulated for.

- Nobody congratulates the makers of SSH, GPG, OpenSSL, Apache, or Mosh for making a system "even they can't hack." And they shouldn't.
- Good design = even the designer has no special access.
- When you retain the ability to auto-update user software without consent or public review, **you become part of the attack surface.**
- Honor the user's informed consent.

The Bad-Attitude Guide to Computer Security

1. **Do** build your own cryptographic protocol.
2. “Safe” languages aren’t *that* safe.
3. HTTPS is bad.
4. Password hashing is bad.
5. Forward secrecy is usually secret, and that’s bad.
6. End-to-end security is bad, and key escrow is good.
7. The Snowden docs shouldn’t have changed our behavior.
8. Auto-updating without consent is bad, because we might become bad.

Keith Winstein
`https://cs.stanford.edu/~keithw`