Newspeak: A Paradigm for Architectural Security

Steven M. Bellovin
http://www.cs.columbia.edu/~smb
Columbia University

September 15, 2008
We’re Doing Security Wrong

- Traditional approaches haven’t worked well in the past
- They aren’t working now
- It is extremely unlikely that they will work in the future
What Has Changed?

- We are much more reliant on (networked) computer systems
- Today’s systems are much more complex (and hence probably have many more bugs)
- We have many more active enemies: cyberthieves, hacktivists, foreign governments, etc.
- The threats have changed, but the vulnerabilities are the same
We're Doing Security Wrong
What Has Changed?

War Stories
CardSystems Solutions
What Went Wrong?
Analysis
SQL Injection Attacks
Industry Response
TJX
Careless Networking, Clever Crooks
One More Horror Story
A Sophisticated Attack

Analysis
A New Hope
Research Needs
Conclusions
June 20, 2005
Lost Credit Data Improperly Kept, Company Admits
By ERIC DASH

The chief of the credit card processing company whose computer system was penetrated by data thieves, exposing 40 million cardholders to a risk of fraud, acknowledged yesterday that the company should not have been retaining those records.

The official, John M. Perry, chief executive of CardSystems Solutions, indicated that the records known to have been stolen covered roughly 200,000
What Went Wrong?

Simple technical flaws!

They had been audited:

CardSystems underwent a Visa security audit in December 2003 and was certified by Visa in June 2004 as complying with Visa’s security rules.

At some point, the company misbehaved:

CardSystems acknowledged it had stored... cardholder names, account numbers, and security codes in violation of both MasterCard’s and Visa’s rules.
Analysis

- The audit didn’t do its job
- It missed SQL injection attacks!
- There was misfeasance by the company
- Their (technical) defenses failed. (An inside job?)
SQL Injection Attacks

(From http://xkcd.com/327/)
Industry Response

- CardSystems Solutions was effectively put out of business by the credit card companies
- Technical standards were tightened
- Did it do any good?
TJX breach could top 94 million accounts
Filings in case involving Visa cards alone as much as $83 million

By Mark Jewell

The Associated Press
TJX used 802.11 ("WiFi") with WEP, a known-weak technology

TJX is of the view that the intruder initially gained access to the system via the wireless local area networks (WLANS) at two stores in the United States.

At the time of the initial penetration, industry standards permitted WEP

Personal data (i.e., driver’s license numbers and social security numbers) was unnecessarily and improperly stored

Losses to TJX approached $200M...
March 18, 2008

NATIONAL BRIEFING | NEW ENGLAND

Maine: Security Breach at Supermarket Chain

By THE ASSOCIATED PRESS

The Hannaford Brothers supermarket chain announced a security breach that began Dec. 7 and led to thefts of customer credit and debit card numbers from more than 200 stores. Hannaford says the security breach affects all
A Sophisticated Attack

- Hannaford Bros. was fully compliant with all relevant standards
- The data was intercepted in transit over fiber optic networks
- How? Sniffing software was installed on hundreds of servers
- But — no end-to-end encryption
The companies involved were largely compliant with industry standards

Industry standards lag the state of the art

Wider use of encryption might seem an obvious answer. But in practice, encryption is unused at certain points in a data-processing chain because the computing power it requires can slow transactions.

“Would you like to sit at your gas pump for five minutes to get an authorization?” said Avivah Litan, a security analyst at Gartner Inc.
The attackers didn’t care about the systems
They wanted data
They wanted financially valuable data
The Attackers Are Knowledgeable

- They’ve attacked obscure industry segments
- They’ve penetrated uncommon software
- They’ve gone after bulk sources of data
- They’ve resold the stolen data to users
Root Causes

- Personnel misbehavior
- Insider attacks?
- Buggy code
- Poor encryption
- Encrypting the wrong thing
Traditional Defenses

- Background checks
  ⇒ Rarely done in the civilian sector

- “Evaluated” code (if we’re lucky)
  ⇒ Misses many bugs; rarely used

- Strong authentication
  Few attacks (except phishing) target authentication systems

- Use good crypto
  ⇒ Most people can’t evaluate the quality of crypto

- Firewalls
  ⇒ Often in the wrong place, and blocking the wrong things
This way lies madness
We will *never* have bug-free code
Complex systems will *always* have security bugs
Almost no one can afford the time and people needed to do things properly
We need a new approach
We're Doing Security Wrong
What Has Changed?

War Stories
Analysis

A New Hope
Design Principles
Classical Design
Wrong and Right
Why?
Data-Centric Approach
Newspeak
Newspeak 2.0
What if the Web Server is Hacked?
Multiple Conversations
Data Flow
A Prototype: Propylaeum
Sample Configuration
The Hard Parts
Why It Works

Research Needs

Conclusions
Design Principles

- Data-centric architecture, with strong protections around the important data
- Accept the inevitability of security holes
- Inherent resilience
- Inherent protection of *most* of the data
Classical Design

- Many applications, often less trusted
- Complex server application
- Back-end database(s) managed by the server application
- Firewalls protect the server
Wrong and Right

The Wrong Approach

Database → Web Server → Browser

The Right Approach

Database → Web Server → Browser
Why?

- The firewall in the first case is pointless: the big risk comes from the web server.
- If the web server falls, the database is completely exposed.
- Or: expose the web server *machine*, turn off all other services, and protect the database.
Data-Centric Approach

- The web server is a syntax translator
- A *simple* language is used between the web server and the database
- Encryption and authentication are from the end user to the database
- Syntax-directed checking of database inputs
“The purpose of Newspeak was not only to provide a medium of expression for the [proper] world-view . . . but to make all other modes of thought impossible.

. . .

“There would be many crimes and errors which it would be beyond [a person’s] power to commit, simply because they were nameless and therefore unimaginable.”

1984, George Orwell
No SQL injection — because SQL is only invoked on sanitized inputs

- No verb to dump the database
- No verb to read a credit card number
- The web server can only operate on accounts selected by end users
What if the Web Server is Hacked?

- Even without end-to-end encryption, only active accounts are at risk
- Most accounts aren’t active most of the time
- Use an IDS to detect web server compromise
Arrows show direction of information flow
Data Flow

- The user object places an order
- Credit card numbers are sent only to the billing system
- The order object supplies the total price
- It also updates the inventory
- The web server can do very little
A Prototype: Propylaeum

- The web server sends Javascript encryption code to the web browser
- All data is encrypted to the Propylaeum daemon
- It decrypts, authenticates, and filters the data
- A simple configuration file describes each data syntax via regular expressions
- Neither the web server nor the database server handle untrusted data
<?xml version="1.0" ?>
<propylaeum>

<variables>
<allowed-value varname="ISBN" regex="[0-9-]">
<allowed-value varname="NAME" regex="[A-Za-z0-9-]">
<allowed-value varname="CC" regex="[0-9 ]">
</variables>

<action name="BOOK_DETAIL">
<query>
</query>
</action>
</propylaeum>
The Hard Parts

- Minor issue: public key encryption in Javascript is slow (and no one implements shttp)
- Major issue: designing the dialect(s) of Newspeak
- Every application is different; middleware layers tend to be too powerful
Why It Works

- The complex logic isn’t trusted — it’s outside of the TCB
- The database isn’t exposed to untrusted, unfiltered inputs
- The filter daemon is small enough that (perhaps) we can get it right
Research Needs
Modularization

- What is the general solution for proper modularization of applications?
- How do we design the data flow?
- What are the tradeoffs between different design options?
Right now, Newspeak is a concept, not a language
- Application designers have found it hard to build their own instantiations
- We need a concrete, general design
Is there a simple way to specify sanitization?

More precisely, how do we make programmers use it?

Simple input types can be sanitized simply; complex types require a complex specification.
Development of proper tools has helped programmer productivity and program correctness in the past

We need Newspeak tools, too

But we don’t yet know what to build!
Conclusions
Looking Back at Tradition

- Designs of the past were host-centric
- The OS was relied on to mediate all data transfer
- Security strength was measured by ACL power
- The network was a glorified form of remote login
- None of that is true today
Today’s Environment

- Network-centric
- Server computers run one application, in one protection domain
- We have no network-wide reference monitor
- Put another way, we need a network form of “least privilege”
- Any such monitor has to be application-specific
Conclusions

- Newspeak isn’t the only possible solution
- But — any robust solution will need to follow some of the same principles
- Modern TCBs are at the application level
- The OS has a secondary role; at best, it can provide strong isolation between components
- The danger points are the communications channels; those need to be strongly protected against bugs