We have to ***watch and listen to everything that people are doing*** *so that we can catch terrorists, drug dealers, pedophiles, and organized criminals. Some of this data is sent unencrypted through the Internet, or sent encrypted to a company that passes the data along to us, but we learn much more when we have* ***comprehensive direct access to hundreds of millions of disks and screens and microphones and cameras****.*

This talk explains how we've successfully manipulated the world's software ecosystem to ensure our continuing access to this wealth of data. This talk will not cover our efforts against encryption, and will not cover our hardware back doors.

Making sure
software stays insecure

Daniel J. Bernstein

University of Illinois at Chicago &
Technische Universiteit Eindhoven

Some important clarifications:

1. "*We*" doesn't include me.

I want secure software.

Some important clarifications:

1. "*We*" doesn't include me.
I want secure software.

2. Their actions violate
fundamental human rights.

Some important clarifications:

1. *"We"* doesn't include me.
I want secure software.

2. Their actions violate
fundamental human rights.

3. I don't have evidence that
they've deliberately manipulated
the software ecosystem.

Some important clarifications:

1. *"We"* doesn't include me.
I want secure software.

2. Their actions violate
fundamental human rights.

3. I don't have evidence that
they've deliberately manipulated
the software ecosystem.

This talk is actually
a thought experiment:
how *could* an attacker manipulate
the ecosystem for insecurity?

Distract managers, sysadmins, etc.

Identify activities that *can't* produce secure software but that can nevertheless be marketed as "security".

Example: virus scanners.

Divert attention, funding, human resources, etc. into "security", away from actual security.

Distract managers, sysadmins, etc.

Identify activities that *can't* produce secure software but that can nevertheless be marketed as "security".

Example: virus scanners.

Divert attention, funding, human resources, etc. into "security", away from actual security.

People naturally do this. Attacker investment is magnified. Attack discovery is unlikely.

2014 NIST "Framework for improving critical infrastructure cybersecurity":

"Cybersecurity threats exploit the increased complexity and connectivity of critical infrastructure systems, placing the Nation's security, economy, and public safety and health at risk. . . .

2014 NIST "Framework for improving critical infrastructure cybersecurity":

"Cybersecurity threats exploit the increased complexity and connectivity of critical infrastructure systems, placing the Nation's security, economy, and public safety and health at risk. . . . The Framework focuses on using business drivers to guide cybersecurity activities and considering cybersecurity risks as part of the organization's risk management processes."

“This risk-based approach enables an organization to gauge resource estimates (e.g., staffing, funding) to achieve cybersecurity goals in a cost-effective, prioritized manner.”

"This risk-based approach enables an organization to gauge resource estimates (e.g., staffing, funding) to achieve cybersecurity goals in a cost-effective, prioritized manner."

- "Identify."
  e.g. inventory your PCs.

"This risk-based approach enables an organization to gauge resource estimates (e.g., staffing, funding) to achieve cybersecurity goals in a cost-effective, prioritized manner."

- "Identify."
  e.g. inventory your PCs.
- "Protect."
  e.g. inventory your humans.

“This risk-based approach enables an organization to gauge resource estimates (e.g., staffing, funding) to achieve cybersecurity goals in a cost-effective, prioritized manner.”

- “Identify.”
  e.g. inventory your PCs.
- “Protect.”
  e.g. inventory your humans.
- “Detect.”
  e.g. install an IDS.

“This risk-based approach enables an organization to gauge resource estimates (e.g., staffing, funding) to achieve cybersecurity goals in a cost-effective, prioritized manner.”

- “Identify.”
  e.g. inventory your PCs.
- “Protect.”
  e.g. inventory your humans.
- “Detect.”
  e.g. install an IDS.
- “Respond.”
  e.g. coordinate with CERT.

“This risk-based approach enables an organization to gauge resource estimates (e.g., staffing, funding) to achieve cybersecurity goals in a cost-effective, prioritized manner.”

- “Identify.”
  e.g. inventory your PCs.
- “Protect.”
  e.g. inventory your humans.
- “Detect.”
  e.g. install an IDS.
- “Respond.”
  e.g. coordinate with CERT.
- “Recover.”
  e.g. “Reputation is repaired.”

Categories inside “Protect”:

- “Access Control”.
- “Awareness and Training”.
- “Data Security”.
  e.g. inventory your data.
- “Information Protection Processes and Procedures”.
  e.g. inventory your OS versions.
- “Maintenance”.
- “Protective Technology”.
  e.g. review your audit logs.

Categories inside "Protect":

- "Access Control".
- "Awareness and Training".
- "Data Security".
  e.g. inventory your data.
- "Information Protection Processes and Procedures".
  e.g. inventory your OS versions.
- "Maintenance".
- "Protective Technology".
  e.g. review your audit logs.

Subcategories in Framework: 98.
. . . promoting secure software: 0.

Categories inside “Protect”:

- “Access Control”.
- “Awareness and Training”.
- “Data Security”.
  e.g. inventory your data.
- “Information Protection Processes and Procedures”.
  e.g. inventory your OS versions.
- “Maintenance”.
- “Protective Technology”.
  e.g. review your audit logs.

Subcategories in Framework: 98.
. . . promoting secure software: 0.

This is how the money is spent.

Distract users

e.g. “Download only trusted applications from reputable sources or marketplaces.”

e.g. “Be suspicious of unknown links or requests sent through email or text message.”

e.g. “Immediately report any suspect data or security breaches to your supervisor and/or authorities.”

e.g. “Ideally, you will have separate computers for work and personal use.”

Distract programmers

Example: automatic low-latency software “security” updates.

Distract programmers

Example: automatic low-latency software "security" updates.

Marketing: "security" is defined by *public security holes*.
Known hole in Product 2014.06?
Update now to Product 2014.07!

Distract programmers

Example: automatic low-latency software "security" updates.

Marketing: "security" is defined by *public security holes*.
Known hole in Product 2014.06?
Update now to Product 2014.07!

To help the marketing, publicize actual attacks that exploit public security holes.

Distract programmers

Example: automatic low-latency software "security" updates.

Marketing: "security" is defined by *public security holes*.
Known hole in Product 2014.06?
Update now to Product 2014.07!

To help the marketing, publicize actual attacks that exploit public security holes.

Reality: Product 2014.07 also has security holes that attackers are exploiting.

Distract researchers

Example:
When researcher finds attack showing that a system is insecure, create a competition for *the amount of damage*.

“You corrupted only one file?”

“How many users are affected?”

“Do you really expect an attacker to use 100 CPU cores for a month just to break this system?”

Distract researchers

Example:
When researcher finds attack showing that a system is insecure, create a competition for *the amount of damage*.

"You corrupted only one file?"

"How many users are affected?"

"Do you really expect an attacker to use 100 CPU cores for a month just to break this system?"

$\Rightarrow$ More attack papers!

Discourage security

Tell programmers that
"100% security is impossible"
so they shouldn't even try.

Discourage security

Tell programmers that
"100% security is impossible"
so they shouldn't even try.

Tell programmers that
"defining security is impossible"
so it can't be implemented.

Discourage security

Tell programmers that
"100% security is impossible"
so they shouldn't even try.

Tell programmers that
"defining security is impossible"
so it can't be implemented.

Hide/dismiss/mismeasure
security metric #1.

<u>Discourage security</u>

Tell programmers that
“100% security is impossible”
so they shouldn't even try.

Tell programmers that
“defining security is impossible”
so it can't be implemented.

Hide/dismiss/mismeasure
security metric #1.

Prioritize compatibility,
“standards”, speed, etc. e.g.:
“An HTTP server in the kernel
is critical for performance.”

What is security?

Integrity policy #1:
Whenever the computer
shows me a file,
it also tells me
the source of the file.

e.g. If Eve creates a file
and convinces the computer
to show me the file
as having source `Frank`
then this policy is violated.

I have a few other
security policies,
but this is my top priority.

The trusted computing base

1987: My first UNIX experience.
Low-cost terminals access
multi-user Ultrix computer.

Picture credit:
terminals.classiccmp.org/wiki/index.php/DEC_VT102

I log in to the Ultrix computer,
store files labeled `Dan`,
start processes labeled `Dan`.

Eve logs in,
stores files labeled `Eve`,
starts processes labeled `Eve`.

Frank logs in,
stores files labeled `Frank`,
starts processes labeled `Frank`.

Eve and Frank cannot
store files labeled `Dan`,
start processes labeled `Dan`.
(Of course, sysadmin can.)

How is this implemented?

OS kernel allocates disk space:

	system files
`Dan`	my files
`Eve`	Eve's files
`Frank`	Frank's files

OS kernel allocates RAM:

	kernel memory
`Dan`	my processes
`Eve`	Eve's processes
`Frank`	Frank's processes

CPU hardware enforces
memory protection:
a user process cannot
read or write files
or RAM in other processes
without permission from kernel.

Kernel enforces various rules.

When a process creates another
process or a file, kernel copies uid.

Process is allowed to read or write
any file with the same uid,
but not with different uid.

Assume the hardware works.
How do we verify that
Eve can't write Dan's files?

1. Check the code that
enforces these rules.

Assume the hardware works.
How do we verify that
Eve can't write Dan's files?

1. Check the code that enforces these rules.

2. Check the code that allocates disk space, RAM; and user-authentication code.

Assume the hardware works.
How do we verify that
Eve can't write Dan's files?

1. Check the code that
enforces these rules.

2. Check the code that
allocates disk space, RAM;
and user-authentication code.

3. Check all other kernel code.
Bugs anywhere in kernel
can override these rules.
Memory protection doesn't apply;
language (C) doesn't compensate.

The code we have to check is the
trusted computing base.
Security metric #1: TCB size.

Eve can't write Dan's files
unless there's a TCB bug.

Eve's actions: irrelevant.
Other software: irrelevant.
Millions of lines of code
that we *don't* have to check.

Do we need an audit log? No.
Keep computers separate? No.
Limit software Eve can run? No.

File sharing

So far have described
complete user isolation.

But users want to share
many of their files:
consider the Web, email, etc.

I want to be able
to mark a file I own
as readable to just me;
or also readable to Frank;
or to Eve+Frank;
or to a bigger group;
or to the general public.

Say Frank creates a file,
makes it readable to me.

I save a copy.

Later I look at the copy.

Remember integrity policy #1:
Whenever the computer
shows me a file,
it also tells me
the source of the file.
$\Rightarrow$ Computer has to tell me
that Frank was the source.

I *own* the copy
but Frank is the *source*.

Obvious implementation:

The OS kernel tracks
source for each file, process.

When my copying process
opens the file from Frank,
the OS kernel marks Frank
as a source for that process.

When process creates file,
the kernel copies source.

Typical OS kernels today
don't even try to do this.

More complicated example:
Eve and Frank create files,
make them readable to me.

I have a process that
reads the file from Eve,
reads the file from Frank,
creates an output file.

More complicated example:
Eve and Frank create files,
make them readable to me.

I have a process that
reads the file from Eve,
reads the file from Frank,
creates an output file.

Integrity policy #1 $\Rightarrow$
The OS kernel marks
both Frank and Eve
as sources for the process,
then sources for the file.

Web browsing

Frank posts `news-20140710`
on his web server.
My browser retrieves the file,
shows it to me.

Integrity policy #1 $\Rightarrow$
My computer tells me that
Frank was the source.

A modern browser tries
to enforce this policy.
But browser is a massive TCB,
very expensive to check,
full of critical bugs.

What if I instead
give Frank a file-upload
account on my computer?

Frank logs in,
stores a file `news-20140710`.
I start a process
that looks at the file.

If OS tracks sources
then it tells me that
Frank was the source.

Why should this be manual?

Browser creates process
that downloads `news-20140710`
from Frank's web server.

("Creating a process is slow."
—Oh, shut up already.)

OS automatically
adds URL as a source
for the process.

Process shows me the file.
OS tells me the URL.

Closing thoughts

Is the community
even *trying* to build
a software system
with a small TCB
that enforces integrity policy #1?

If software security is a failure,
does this mean that
security is impossible,
or does it mean that
the community isn't trying?