Secure Information Flow Analysis: A Science of Hacking?

Geoffrey Smith
School of Computer Science
Florida International University
Miami, Florida USA
"Snoop Software Gains Power and Raises Privacy Concerns"

New York Times, 10/10/03

Privacy and the Internet

Keystroke loggers monitor a target computer, recording all keystrokes (credit card numbers, passwords, ...). Said to be common on public terminals (airports, Kinko's, ...).

"Silent deploy" lets the logger be installed via e-mail!
Standard Techniques for Privacy

- Access Control: Controls who can directly access information.
- Encryption: Secures communication channels. Does not prevent the receiver from leaking the information.
- Digital Signatures: Tells you who wrote a program. Does not ensure that they are trustworthy.

3
Example: A Tax Return Applet

IRS

My computer

TrustMe site

billing info (encrypted)

tax return (encrypted)

4
Want to prevent the program from leaking any information from variables to variables.

Want to prevent the program from leaking any information from \( H \) to \( S \) to \( C \) to \( L \).

(Or a richer classification: \( \mathcal{L} > S > C > L \).

(high, private).

\( H \) •

(low, public) or

\( T \) •

Classify program variables as

We want an end-to-end privacy property.

Secure Information Flow
An explicit flow:

\[ \text{leak} = \text{secret} \]

Two permissible flows:

\[ \text{secret} = \text{leak} \]
\[ \text{leak} = 76318 \]

An implicit flow:

If \( \text{secret} \mod 2 = 0 \)

\[ \text{leak} = 0 \]
\[ \text{else} \]

\[ \text{leak} = 1 \]

Assume \( \text{secret} \) is \( H \) and \( \text{leak} \) is \( L \).

Some Hacking
AMulti-ThreadedExample

Assume trigger is initially 0.

Thread Odd:

if(secret%2==0)
while (trigger == 0)
if (secret % 2 == 1)
    leak = 0;
    trigger = 1;

Thread Even:

if(secret%2==1)
while (trigger == 0)
if (secret % 2 == 1)
    leak = 1;
    trigger = 1;

A Multi-Threaded Example
An Example Using Out-of-Bounds Array Indexing

Assume leak is initially 0.

Once all the threads terminate, or abort, leak contains a copy of the 10-bit secret.

Thread where:

\[ a[1] \rightarrow \text{secret} \rightarrow q \pmod{2} = 0 \]

\[ a[] = \text{new int}[1] \]

Thread q, where 0 < q < 10.

Assume leak is initially 0.
An Example Using Timing

Thread:
while (secret > 0)
    secret--;
    leak = 2;

Math.sqrt (2.0); leak = 1;

Assuming a probabilistic thread scheduler, there is a probabilistic flow:

Thread a:
while (secret < 0)
    secret--; 
    leak = 2;

Thread b:
An Example using Caching

int i, count, xs[4096], ys[4096];
for(count=0; count<100000; count++) {
    for (i = 0; i < 4096; i++)
        if (secret)
            xs[i]++;
        else
            ys[i]++;
    for (i = 0; i < 4096; i++)
        xs[i]++;
    for (i = 0; i < 4096; i++)
        ys[i]++;
    for (i = 0; i < 4096; i++)
        if (secret)
            xs[i]++;
    for (count = 0; count < 100000; count++)
        i; count, xs[4096], ys[4096],;}

On Sparc server goliath (with a 16K data cache), this takes about twice as long when secret is 0 as it takes when secret is 1!
Noninterference [Goguen and Meseguer 82]

Noninterference [Goguen and Meseguer 82]

What does it mean to say, "Program $P$ leaks no information from $H$"? It depends on what is $T$-observable.

But what should $(P', \eta) \stackrel{T}{\sim} (P', \eta)$ mean?

Noninterference Property: If $P \eta = \eta$, then $(P, \eta) \sim (P', \eta)$.

Idea: If we run $P$ twice, on initial memories that differ only on the values of variables to $T$ variables.

Def: Memories $\eta$ and $\eta'$ are $T$-equivalent, written $\eta \sim \eta'$, if and only if $\eta$ and $\eta'$ agree on the values of all $T$ variables.

Def difference.

What does it mean to say, "Program $P$ leaks no information from $H$"? It depends on what is $T$-observable. Shouldn't be able to tell the difference.

But what should $(P', \eta) \sim (P', \eta)$ mean?
nondeterministic or probabilistic.

A further complication is that with multi-threading the program may be

(semantics).

Especially nasty is that leaks may occur at a lower level than our formal

prevent leaks.

Of course, the more allowed $T$-observations there are, the harder it is to

\[ \ldots \bullet \]

- The number of page faults during the program's execution.
- The program's running time.
- Whether or not the program terminates.
- Whether or not the program aborts.
- The final values of $T$ variables.
- Possible $T$-observations:
Static Analyses for Secure Information Flow

Idea: Analyze programs statically before executing them. Programs that pass the analysis can be executed safely.

Denning's analysis guarantees a noninterference property. Denning's analysis can be formulated as a type system.

Pioneered by [Denning and Denning 77].

But because the analysis must be conservative, some safe programs will necessarily be rejected.

Volpano, Smith, and Irvine 96]:

Much work has followed: A recent survey [Sabelfeld and Myers 03] includes about 150 references.
Probabilistic Noninterference property

Changing the initial values of \( V \) variables does not change the joint distribution of possible final values of \( T \) variables.

Probabilistic Noninterference property

(But multi-threading makes abstract time observable internally!)

Limit \( T \) observations to the final values of \( T \) variables.

Focus on the multi-threaded case with a uniform probabilistic scheduler.

Probabilistic Noninterference
Classify and restrict expressions and commands.

A Type System for Probabilistic Noninterference

Classifications:

1. An expression $e$ is $H$ if it contains variables of type $\tau$; otherwise it is $L$.

2. A command $c$ is $\tau/\tau_2$ if it assigns only to variables of type $\tau_2$.

3. A command $c$ is $\tau$ cmd $u$ if it assigns only to variables of type $\tau$ (or lower).

Higher (or higher) and it terminates in exactly $n$ steps.

15
Restrictions:

1. An expression cannot be assigned to an variable.

(No explicit leaks.)

2. A guarded command with guard cannot assign to variables.

(No implicit leaks.)

3. A command whose running time depends on variables cannot be followed sequentially by a command that assigns to variables.

(No timing leaks.)

A very similar type system was developed independently by Boudol and Castellani (01).
while (secret == 0) { leak = 5; } while (secret == 0) { leak = 1; } while (leak == 0) { leak -= 1; } while (leak == 0) { H } illegal while (secret == 0) { leak = 1; } illegal while (leak == 0) { leak = 5; } illegal while (leak == 0) { leak = 6; } illegal while (secret == 0) { leak = 0; } illegal

Example Typhing
Multi-Threaded Programs as Markov Chains
If a well-typed command is run under two \( \ell \)-equivalent memories, it makes exactly the same assignments to \( \ell \) variables, at the same times.

Soundness of Type System
Probabilistic Bisimulation on Markov Chains

Given a Markov chain with state set $S$ and transition probabilities $p_{ab}$, for all $a, b \in S$, and an equivalence relation $\approx$ on $S$, Strong with respect to timing—from $a_1$ and $a_2$ we pass through the same equivalence classes of the same times.

\[ \begin{array}{c}
q_{a_1}p_{a_2} = q_{a_2}p_{a_1} \\
\end{array} \]

Definition: is a probabilistic bisimulation if

\[ a_1 \approx a_2 \text{ implies } q_{a_1} \approx \approx q_{a_2} \]

Kemeny, Snell 60, Larsen, Skon 91, Sabelfeld, Sands 00

Probabilistic Bisimulation on Markov Chains
is not a probabilistic bisimulation:

But, abstracting away from time, both \( a_1 \) and \( a_2 \) to \( B \) with probability \( 2/3 \).

\( a_2 \) goes to \( B \) (in one step) with probability \( 2/3 \). 

\( a_1 \) goes to \( B \) (in one step) with probability \( 1/3 \).
Let $A$ and $B$ be distinct equivalence classes, $a \in A$.

Definition: is a weak probabilistic bisimulation if

\[ p(a \rightarrow A, B) = (p(a, A') \cdot A') + p(a, B) \]

Appropriate if we are interested in the sequence of equivalence classes that are visited, but not how long the chain remains in each class.

Let $A$ and $B$ be distinct equivalence classes, $a \in A$.

Weak Probabilistic Bisimulation on Markov Chains

([Baier and Hermanns 97] define a similar notion for a process algebra.)
Applying Weak Probabilistic Bisimulation

This implies that our type system guarantees probabilistic noninterference.

\[ \forall T \sim \eta', \text{ then } (\eta', D) \sim (\eta', D) \]

\[ \text{and respects the } T\text{-observability of variables:} \]

\[ (\eta', D) \]

We can define a weak probabilistic bisimulation on well-typed configurations:

Applying Weak Probabilistic Bisimulation
Challenges

We'd like to infer the security class of "internal" variables.

Consider arrays, exceptions, object-oriented features, ...

Enrich the language.

We'd like to infer the security class of "internal" variables.

False positives.

Limited leaks.

What if you want to leak a little information?

Enrich the language.

Challenges
Sponsored by NSF grants CCR-9596113, CCR-9612176, CCR-9900951.

Dennis Volpano (Naval Postgraduate School, Cranite Systems)
Cynthia Irvine (Naval Postgraduate School, Cranite Systems, current PhD student)
Ryan Yocum (FIU master's thesis 02)
Zhenyue Deng (current PhD student)
Daniel Cazalis (current PhD student)

Collaborators