Putting Trojans on the Horns of a Dilemma: Redundancy for Information Theft Detection

Jedidiah R. Crandall[1], John Brevik[2], Shaozhi Ye[3], Gary Wassermann[3], Daniela A. S. de Oliveira[3], Zhendong Su[3], S. Felix Wu[3], and Frederic T. Chong[4]

[1] crandall@cs.unm.edu, Dept. of Computer Science, University of New Mexico
[2] jbrevik@csulb.edu, Dept. of Mathematics and Statistics, California State University, Long Beach
[3] {yeshao,wassermg,oliveira,su,wu}@cs.ucdavis.edu, Dept. of Computer Science, University of California at Davis
[4] chong@cs.ucsb.edu, Dept. of Computer Science, University of California at Santa Barbara

Abstract. Conventional approaches to either information flow security or intrusion detection are not suited to detecting Trojans that steal information such as credit card numbers using advanced cryptovirological and inference channel techniques. We propose a technique based on repeated deterministic replays in a virtual machine to detect the theft of private information. We prove upper bounds on the average amount of information an attacker can steal without being detected, even if they are allowed an arbitrary distribution of visible output states. Our intrusion detection approach is more practical than traditional approaches to information flow security.

We show that it is possible to, for example, bound the average amount of information an attacker can steal from a 53-bit credit card number to less than a bit by sampling only 11 of the 2^{53} possible outputs visible to the attacker, using a two-pronged approach of hypothesis testing and information theory.

Key words: Intrusion detection, information theft detection, malware analysis, information theory

1 Introduction

Suppose an attacker installs a Trojan on a system S (*e.g.*, an ATM[5]) to steal confidential information (*e.g.*, a 16-digit or 53-bit credit card number) and transmit it over some channel to the attacker through a low-security output such as the network. Let the confidential information (the high-security input) be A and the low-security output be B. The only way for the attacker who can view B to learn information about A is through the mutual information between A and B,

[5] Note that ATMs often run commodity operating systems and can be susceptible to malicious code attacks, such as the Slammer worm [1].

written as $I(A;B)$. In this setting the Trojan is the encoder that encodes input A and sends this over a channel where the attacker receives B and decodes it.

We wish to frame a policy that bounds $I(A;B)$, and detect any violations of this policy. There are many difficulties because the attacker can execute arbitrary code on system S and choose an arbitrary distribution of the possible output states for B. Furthermore, for a practical system S we must account for architectural nondeterminism, such as hard drive turbulence, electrical noise, high speed buses, memory scrubbing [2], or user input, and we must also consider the possibility of covert channels. We leave applying existing and new techniques to covert channel analysis as future work, and instead in this paper focus on a general two-pronged approach to bound $I(A;B)$ that could be applied to many different applications where an attacker (or their Trojan) can arbitrarily encode information.

The basic idea is to checkpoint the system and record all nondeterministic events while a transaction executes. Then, possibly in parallel, multiple deterministic replays of the transaction from the checkpoint with varying high inputs can be used to measure the entropy of low outputs. The recording and replaying can be done either by placing the system in a virtual machine or using special hardware. After removing the architectural nondeterminism, we can probabilistically measure the amount of information leaked using Shannon's information theory [3] and hypothesis testing. Figure 1 illustrates our approach.

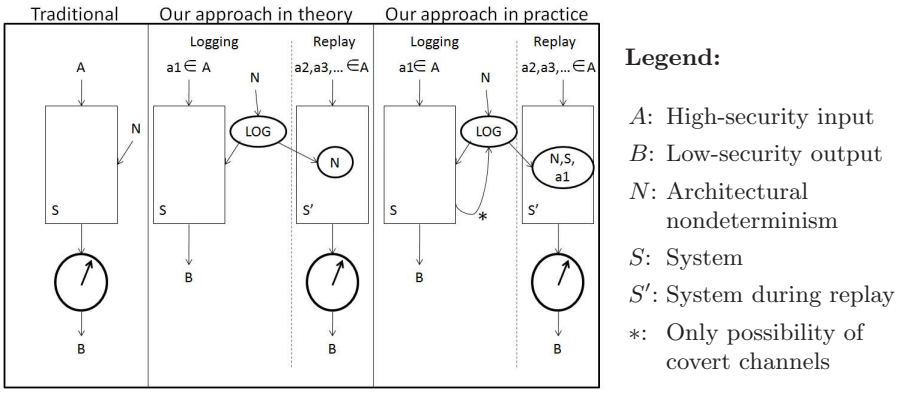

Fig. 1. An overview of our approach.

For the traditional system S in the first case, measurement of the entropy in the visible output B is confounded by the architectural nondeterminism N. In theory, as illustrated in the second case, we could log all nondeterminism N during the original transaction (where the high input is a_1) and then create a deterministic system S' using deterministic replay of the transaction, then repeatedly vary the input A $(a_2, a_3, ...)$ to obtain a measurement of the mutual information between A and B without the possibility of covert channels.

In practice, we cannot log architectural nondeterminism directly, but we can log nondeterministic events (interrupts and input events to the CPU) and perform deterministic replay. This is illustrated in the third case of Figure 1. There still is no possibility of exploiting covert channels from the original system, but it is possible for entropy to be hidden from us during replay because nondeterministic events can contain both architectural nondeterminism and information that should otherwise be deterministic, such as the original input $a_1 \in A$. For example, the Trojan might mask out the first bit of a_1 and make a request to the hard drive for data far away on the other side of the cylinder if the bit is a 1 or make a request for data closer to the head's current position if the bit is a 0. If we log the timing of this disk request and force it to be the same during repeated replays for $a_2, a_3, ...$ then mutual information between A and B is hidden from us during measurement. Fortunately, a very important property of our approach in practice is that covert channels are only possible as channels with timing characteristics involving at least one event in the log file. This could enable a systematic, exhaustive covert channel analysis by directly applying Wray's technique [4] to all events that we log. We leave the details for future work.

A hypothesis test is performed by counting the number of distinct output states for B, assuming an equal distribution. In this paper we will also show that with an attack model where the attacker must pass the hypothesis test with a minimum probability of ρ, the attacker has a very limited amount of additional entropy that can be utilized assuming an arbitrary distribution. Thus the attacker is placed "on the horns of a dilemma" [5] between using an unequal distribution, which lowers their channel capacity, and being detected.

1.1 Contributions

At a high level, the contribution of this paper is an approach that detects information theft by measuring explicitly everything that could have happened. Historically, information flow security mechanisms have suffered from specific problems, such as implicit channels and covert channels, which are caused at their root by the fact that information is a measure of everything that could have possibly happened, not what did happen.

The specific contributions of this paper are:

- A general method to detect information theft based on repeated deterministic replays. Compared to previous approaches, this method allows for more realistic policies where information leakage is bounded rather than strictly forbidden. For example, when passwords are entered one bit of information is leaked about the password file because of the success or failure of logging in, thus information flow cannot be completely contained but if it can be bounded then confidentiality of the password file can be enforced in a practical way.
- A theory of replay-based enforcement of information flow security based on logging nondeterministic events that incorporates the only possibility of covert channels: channels with timing characteristics through the logging

mechanism, where we show that all covert channels must act through mutual information between high-security input A and the log L.
- A probabilistic bound on the mutual information between a high input and low output based on hypothesis testing and assuming an equal distribution of states.
- A bound on the additional mutual information an attacker can gain by using an unequal distribution of states with an attack model that allows the attacker some probability ρ of passing the hypothesis test (and therefore evading detection).

The two bounds calculated with the latter two, ϵ and δ respectively, form a probabilistic bound $\epsilon + \delta$ on the average amount of information that can be leaked to an attacker.

1.2 Applications

We envision many applications of the approach proposed in this paper, including:

- **Malware analysis:** A Trojan that employs cryptovirology [6] and exploits covert channels in commodity operating systems could be placed in a virtual machine and the mutual information between various inputs (*e.g.*, credit card numbers, passwords, personal files) and potential outputs (*e.g.*, the network, the hard drive) could be measured as an analysis of the Trojan's capabilities and purpose.
- **Testing to guide system design:** The basic framework for information flow measurement via repeated deterministic replays presented in this paper could be used to test existing systems for inference and side channels [7–10].
- **Information flow security systems:** Researchers have begun to explore the possibility of building systems that enforce information flow security by comparing different outputs for different inputs [11]. We envision a virtualization system that could enforce probabilistic non-interference [12] policies on commodity systems by applying the basic concept of redundancy and using the unwinding theorem [13].
- **Intrusion detection for dedicated systems:** Compared to traditional time-sharing systems, systems such as voting machines and ATMs have well-defined transactions and relatively coarse information flow policies, making our intrusion detection approach to information security directly applicable to such systems.

1.3 Structure of the Rest of the Paper

This paper is structured as follows. After illustrating an example using our implementation of replay-based confidentiality enforcement in Section 2, we formally state our problem and give an overview of the entire approach in Section 3. Then Section 4 explains how to bound the mutual information measured in bits between a high input and low output by some parameter ϵ, using hypothesis

Fig. 2. Results from repeated replays for (a) no bits leaking, (b) 2 bits leaking (allowed by the policy $\epsilon = 2$), and (c) 3 bits leaking (violation of policy). Each replay has an input ⇒ output pair where ♣, ♡, ♠, and ♢ represent distinct outputs observed for that experiment.

testing and assuming an equal distribution of states is used by the attacker for the output. Section 5 shows that we can calculate some caveat δ, based on the equivocation that is forced on the attacker by a probabilistic attack model where they must have a probability ρ of passing this hypothesis test, where δ is the maximum amount of additional mutual information that can be gained by the attacker by violating the assumption that their distribution of states for the output is equal. We review the related work in Section 6 followed by the conclusion and discussion of future work in Section 7.

2 Prototype Implementation and Example

This section gives an example of replay-based detection of information theft Trojans. Our main goal in this example is to demonstrate that information flow detection can be applied in practice for a full system. We chose a simple policy to enforce: that no more than $\epsilon = 2$ bits of a secret input can be written to the hard drive during the transaction. We were able to enforce this policy in a full system implementation that exercised code in all the components, including all of the drivers, the kernel, library code, and applications of a Linux system. We used a deterministic full-system replay virtual machine implementation that is described in another paper [14].

In our example, the user inputs a high input, which is read as a generic block device. This generic block device is handled in the same manner as the hard drive by our VM implementation: interrupts and port input timings are logged and replayed but the data is read deterministically through port input. The input, a random string of 10 characters, after being processed by drivers and kernel services, is delivered to an application which encrypts it using RSA from the OpenSSL libraries [15] and a randomly generated key based on entropy from

Definitions	
$S : A \to B$	A nondeterministic system
D	The system specification
N	The nondeterminism for a particular trace
L	The log file of nondeterministic events
$S' : A \to B$	S' is a deterministic system, A is the high input, B is the low output
Parameters	
ϵ	Number of bits allowed to leak in the policy
c	Constant, shorthand for 2^ϵ
γ	Confidence of hypothesis test
ρ	Attack model parameter, probability the attacker needs of passing the hypothesis test
δ	Caveat, the most entropy the attacker can gain by using an unequal distribution of the states of B
r	The number of replays needed in the hypothesis test for a confidence of at least γ
q	The proportion of the states in B that the attacker must use as equivocation to pass the hypothesis test

Fig. 3. Legend.

"`/dev/urandom`". The output is the MD5 cryptographic checksum of the hard drive change log, which contains only hard drive blocks that have been modified since the last checkpoint.

Figure 2(a) shows the results of 10 replays of the original transaction when no information is leaked to the hard drive. Each entry shows an "input ⇒ output" pair. Because no information leaks in this case, all output states are the same.

Figure 2(b) shows the results of the same experiment when slightly less than 2 bits leak (per 10 characters). In this example two bits are simply masked out and written directly to the hard drive[6]. Figure 2(b) shows that after 38 replays of the transaction only four states have been observed, so that with 99.8% confidence (see Section 4.3) we can reject the null hypothesis and ensure that the policy $\epsilon = 2$ was enforced. Figure 2(c) shows the same experiment with slightly less than 3 bits leaking. After only 5 replays a fifth final state is observed, which signals that the policy $\epsilon = 2$ was probably violated and thus the information flow theft by the Trojan is detected.

3 Definitions and Overview

This section gives the problem definition and an overview of our approach to bounding the amount of information leaked from a high input A to a low output B. The important symbols are listed in Figure 3.

[6] The actual amount of information leaked is slightly less than two bits since the ASCII representation of possible characters between *"A"* and *"Z"* or *"a"* and *"z"* does not have an equiprobable distribution of the last two bits.

3.1 Problem Definition

Through logging and deterministic replay of a system S we have a deterministic system S' that takes a high input A and maps it to a low output B that is visible to the attacker, *i.e.*, $S' : A \rightarrow B$. We wish to bound with high confidence the mutual information $I(A;B)$ by the constant $\epsilon + \delta$, where ϵ is the maximum amount of entropy assuming the attacker uses a uniform distribution of the possible outputs in B and δ is the maximum amount of entropy that can be gained in addition to ϵ by using an arbitrary, possibly non-uniform distribution of B. Part of the attack model is that the attacker must encode the channel so that they have a probability ρ of passing the hypothesis test that assumes a uniform distribution of B. We assume that we know the distribution of confidential input A. ϵ is chosen beforehand and δ is calculated as a caveat added to the test for ϵ.

3.2 Overview of Approach

The first step is to perform r deterministic replays such that, assuming an equal distribution of B, the mutual information $I(A;B)$ is bounded by ϵ. This is done through hypothesis testing as described in Section 4.

The second step, described in Section 5, is to calculate some caveat, so that if the attacker did choose an encoding for the channel from input A to output B (recall that the attacker can view only B, but may install arbitrary malicious code on the system S to encode input A in output B) with an arbitrary distribution but under the constraint that they must have a probability ρ of passing the first step, then—because of the equivocation forced on them—the most additional entropy beyond ϵ bits that they can use to transmit information from A to B is bounded by δ. Thus, with confidence γ, $I(A;B) \leq \epsilon + \delta$. Here ϵ is a parameter and δ is calculated as a caveat based on the distribution of A, ϵ, and the number of replays r.

4 Hypothesis Testing to Bound ϵ

In this section we describe how to perform hypothesis testing of system S' through repeated deterministic replays to enforce a bound of ϵ on the mutual information between high input A and low output B.

4.1 Deterministic Replay In Theory

We want to bound the mutual information between A and B, $I(A;B)$. The entropy (or information) in B can only come from three sources (all defined from the attacker's perspective): D (determinism, or the specification of the system along with all malicious software and configuration), A (confidential data), and N (architectural nondeterminism such as hard drive turbulence or electrical

noise [2]). In theory, D, A, and N are independent[7]. Thus, $H(B) = I(B;D) + I(B;A) + I(B;N)$. The term $I(B;D)$ is zero by definition[8] so we can bound $I(B;A)$ to be as small as the arbitrary policy-specific constant ϵ, by bounding $H(B) - I(B;N)$, which is equal to the conditional entropy of B if N be known $H(B|N) = H(B) - I(B;N)$[9].

$$\begin{aligned} I(B;D) + I(B;A) + I(B;N) &= H(B) \\ 0 + I(B;A) &= H(B) - I(B;N) \\ I(B;A) &= H(B|N) \end{aligned}$$

Therefore, if we measure that $H(B|N) \leq \epsilon$, then:

$$I(B;A) \leq \epsilon$$

4.2 Theory vs. Practice

If we replay from a log file L then we are able to measure $H(B) - H(L)$, where $H(L) \leq I(L;D) + I(L;A) + I(L;N)$. This inequality says that the entropy in the log file can only come from three sources: D, A, and N. The deterministic behavior that we did not need to log and replay from the log but did ($I(L;D)$) is chosen nondeterministically, meaning that before the transaction we do not know what determinism will be logged, so this term can be non-zero[10]. Any entropy from the confidential data A that was logged ($I(L;A)$) and then replayed will mask the entropy and can hide a covert channel from us. The last term ($I(L;N)$) is all of the architectural nondeterminism that actually must be logged and replayed for deterministic replay to work correctly. The total entropy in the log $H(L)$ is given as an inequality since logging and replaying what would otherwise have been deterministic causes $I(L;D)$, $I(L;A)$, and $I(L;N)$ to no longer be independent. Now we have:

$$\begin{aligned} H(B) - H(L) \geq I(B;D) + I(B;A) + I(B;N) - I(L;D) \\ -I(L;A) - I(L;N) \end{aligned}$$

Under replay, assuming the replay is in fact deterministic with respect to the nondeterminism N, we have the equality $I(L;N) = I(B;N)$. It says that for the replay to work all of the truly nondeterministic events must be in the log.

---

[7] The input A that we are interested in is separated out from N by definition, since we assume that it is entered through a special device that distinguishes it as high-security information.

[8] Note that $H(D) = 0$ for a system that behaves as specified, so the information of D, and therefore its mutual information with all other variables, is 0. We take the system S as its own specification, so $H(D) \neq 0$ is only possible for S' during replay.

[9] Conditional entropy is also sometimes written as $H_N(B)$.

[10] Recall that D is the system specification and is therefore only deterministic when the system S behaves as specified, so that $I(L;D)$ can be non-zero for S' during replay.

With logging there is now some imprecision in our policy enforcement:

$$\begin{aligned} & I(B;D) + I(B;A) + I(B;N) \\ & \quad -I(L;D) - I(L;A) - I(L;N) \leq H(B) - H(L) \\ & 0 + I(B;A) + I(B;N) \\ & \quad -I(L;D) - I(L;A) - I(L;N) \leq H(B) - H(L) \\ & I(B;A) - I(L;D) - I(L;A) \qquad \leq H(B) - H(L) \end{aligned}$$

Assume that we have measured $H(B) - H(L) \leq \epsilon$, then:

$$I(B;A) \leq \epsilon + (I(L;D) + I(L;A))$$

In other words, for the policy to be enforced it suffices that $I(L;D) + I(L;A) = 0$, which implies that $I(L;D) = 0$ and $I(L;A) = 0$ since entropy cannot be negative. When $I(L;D) \neq 0$ that implies that there is some benign imprecision in our replay due to the fact that we are logging and replaying, explicitly or implicitly, part of the system specification that should be deterministic[11]. When confidential data is logged $I(L;A) \neq 0$, meaning that there is a covert channel from A to B through logged entropy that will be replayed from the log during replay and mutual information between A to B will be hidden from our measurement. Covert channels are a concern because we cannot tell the difference between these two sources of imprecision without a precise information flow analysis to distinguish among $I(L;D)$, $I(L;A)$, and $I(L;N)$, but a specific ordering property of events in the log enables a systematic, exhaustive covert channel analysis because the log file is the only source of covert channels. No high-security input can be transmitted through a covert channel without also being logged as $I(L;A)$, something we plan to explore in future work.

4.3 Measuring $H(B) - H(L)$

We wish to do r replays using the log file L, from the original transaction, and then be able to say that with some probability γ, that at least $c+1$ final system states in B (or c more final states) would have been observed if the policy were violated, meaning more than ϵ bits of entropy has leaked, where $c = 2^{\epsilon}$. The number of possible final system states may be greater than $c+1$ but we assume the worst case so that we can do hypothesis testing and gain certainty that the policy was enforced. Our alternate hypothesis is that $|B| - |N| \leq 2^{\epsilon} = c$, thus varying the confidential data will not introduce too many extra final states in B's view and we can say that the policy was enforced.

We assume an equal distribution of possible final states in B for the hypothesis test. If the attacker desires a probability $\rho = 1$ of passing the hypothesis test, *i.e.*, if they want a zero chance of the information flow security violation being detected, then they can not use more states than what ϵ allows. Therefore, the attacker must use an equal distribution of B to maximize $I(A;B)$ since an

[11] Note that for practical implementations this benign imprecision is unavoidable, because of caches and other hardware issues.

unequal distribution only decreases the entropy[12]. In the next section, Section 5, we will calculate a caveat δ that is the maximum additional entropy the attacker can gain from an unequal distribution of their coding of B when $\rho < 1$, but for the hypothesis test we assume an equal distribution.

Note that hypothesis testing only gives us confidence that the policy was enforced if we reject the null hypothesis, it does not prove that the policy was violated if the null hypothesis is not rejected. The designer of a system based on our technique would have to take great care in defining the set A and varying it randomly, since attacker assumptions need to be accounted for [16]. For our purposes in this paper we assume that the distribution of A that the attacker assumes is known to us, and our bound of $I(A;B) \leq \epsilon + \delta$ is probabilistic and is a bound on the *average* amount of information about A the attacker can learn.

For $\epsilon = 0$ there are two possible final states if a single bit leaks, so the probability is $\gamma = 1 - 2^{-(r-1)}$. With every replay that does not reach a different final state of B from the original trace we cut our uncertainty that the policy was enforced in half, so that our certainty that the policy was enforced asymptotically approaches 1. Only in the special case that $\epsilon = 0$ is the equation for $1-\gamma$ similar to a geometric distribution, in general the trials are not independent (a trial being whether or not we observe a new state for B), which becomes a factor when $\epsilon \neq 0$ meaning that there are more than two possible states for B.

Non-zero values of ϵ, even if they are small, may be important for some practical policies. For example, if the attacker can determine whether a user entered a valid input or not then that is a leak of one bit of information that the system designers may wish to allow. For practical values of ϵ we have performed a numerical matrix power calculation to determine the number of replays needed to gain 99.9% certainty ($\gamma = 0.999$), shown in Table 1.

ϵ	$c+1 = 2^{\epsilon}+1$	**Number of replays (r)**
0	2	11
1	3	15
2	5	39
3	9	78
4	17	161
5	33	338

Table 1. The number of replays necessary for 99.9% certainty that the policy was enforced.

The numerical method for generating this table is as follows. For the general case that $\epsilon > 0$, we can view the process as a Markov process with $c+2$ Markov process states (0 being the start state, $c+1$ being a saturating end state) where being in state j means that j different final system states have been observed

[12] For example, for a random variable with two possible output states the maximal entropy is $0.5 \log \frac{1}{0.5} + 0.5 \log \frac{1}{0.5} = 1$ bit, an unequal distribution decreases the entropy, such as $0.3 \log \frac{1}{0.3} + 0.7 \log \frac{1}{0.7} \approx 0.881$ bits.

(note that a final system state and a Markov state are not the same, just two uses of the same word). Then we assume that more than ϵ bits were leaked, *i.e.*, our null hypothesis is that at least $c+1$ final states will be observed.

Thus the probability of the Markov process going from state j to state $j+1$ for each replay is $m_{j,j+1} = \frac{c+1-j}{c+1}$, and the probability of staying in the same state j is $m_{j,j} = 1 - m_{j,j+1}$ when $j \neq c+1$ and 1 when $j = c+1$, all other elements being 0. For small values of ϵ a transition probabilities matrix M can be constructed with the elements $m_{i,j}$ forming an upper bidiagonal matrix, where M^k gives the k-step probabilities matrix that we will go from state i to state j in k steps. Then the probability γ that we make it to state $c+1$ in k steps can be solved numerically by performing the matrix power calculation M^k and γ will be equal to the upper right corner element $(0, c+1)$ of the upper triangular matrix Q^k. We can choose the minimum $r = k$ where $\left(M^k\right)_{0,c+1} > \gamma$.

5 Caveat δ

In this section we show how to calculate the caveat δ, which is the additional amount of entropy in $I(A;B)$ that the attacker can utilize given the constraint that they must pass the hypothesis test with probability ρ. The structure of this calculation is as follows: the attacker must use a proportion q of their state space of B in order to, with probability ρ, pass the hypothesis test that bounds ϵ. This equivocation (mapping multiple inputs from A to the same output states of B), which is forced on the attacker by the hypothesis test, will bound by δ the additional amount of entropy in $I(A;B)$ that can be gained by an arbitrary distribution. This section is structured as follows: first we show how to calculate the probability ρ of passing the hypothesis test for a given value for q, then we solve this equation for a bound for q as a function of ρ since in practice ρ is a policy parameter of the attack model and q is the value we want to calculate, and finally we show how to calculate δ as a function of q.

5.1 Start with ρ as a Function of q

The optimal strategy, in terms of ρ, for an attacker to pass the hypothesis test with probability ρ and $I(A;B) > \epsilon$ is to divide the states of B up into $c+1$ bins, with a proportion q of these states dedicated to the c bins of size $\frac{q}{c}$ that through equivocation will be allowed by the hypothesis test. The remaining bin of size $1-q$ is the bin which the attacker is allowed to use for additional entropy beyond ϵ. We assume that $1 - q < \frac{q}{c}$, which holds for appropriate values of ρ, r, and γ. The optimal strategy for the attacker to maximize their entropy is to use some bin smaller than $1-q$ and not equivocate all the states in the bin. By assuming a bin size of $1-q$ where the attacker can gain additional entropy, we are covering the worst case for a bin size, and then a worst case entropy for that bin size serves as our bound δ. Thus δ is not a tight bound, *i.e.*, the attacker cannot actually achieve δ bits of additional entropy, but δ serves as a good bound in practice.

Based on our assumption that $1 - q < \frac{q}{c}$, we can calculate the probability ρ that an attacker will pass the hypothesis test using a portion q of their states for B.

$$\rho = q^r + c\left(1 - \frac{q}{c}\right)^r$$

This equation is based on the insight that all r samples for the hypothesis test must come from at most c of the $c+1$ bins. Thus either all must from come the portion q, *or* if any sample comes from the bin of size $1 - q$ then there are c ways of not having sampled one of the $\frac{q}{c}$-sized bins.

5.2 Bound q as a Function of ρ

In practice, we wish to set ρ as a parameter and calculate q. Rather than solving for q, we can bound q as follows. The proof is provided in the appendix.

$$q > \left(\rho - \frac{c^{r+1}}{(c+1)^r}\right)^{1/r}$$

5.3 Calculate δ as a Function of q

We wish to bound the additional entropy the attacker can gain from an unequal distribution of states, $I(A;B) - \epsilon \leq \delta$. By equivocation, in order to pass the hypothesis test with probability ρ, the attacker now can have $n \leq (qH(A))$ states in a bin of size $p \leq 1 - q$ with which to transmit information beyond the ϵ bits allowed by the hypothesis test.

This additional entropy is maximized when $I(A;B) - \epsilon = p \log n - p \log p$ (this is a corollary to Shannon's result on additivity). Note that $q > 0.5$ because we have assumed that $1 - q < \frac{q}{c}$ with $c \geq 1$.

This means that $I(A;B) - \epsilon \leq (1-q)\log(qH(A)) - (1-q)\log(1-q)$, so we can bound $I(A;B) - \epsilon \leq \delta$ by setting $\delta = (1-q)\log(qH(A)) - (1-q)\log(1-q)$.

5.4 Example

Table 2 shows the various stages of the calculation of the caveat for different policy parameter values of ϵ, assuming $\rho = 0.9$ and that the high input A is a credit card number which can be viewed as a 53-bit number so that $H(A) = 53$. It is interesting to note that the caveat δ depends on ρ, $H(A)$, and r, and other than the dependence of r on γ, r is totally independent of ϵ. If a system designer's goal was to drive $I(A;B)$ as close to 0 as possible, for example, they could set $\epsilon = 0$, $r = 161$, and with a very high confidence γ bound $I(A;B)$ to be less than 0.0350100747.

A system designer that wanted to trade some security for better performance while allowing for small leaks that would lead to false positives in other systems might do the following. By setting $\gamma = \rho = 0.69$ and $\epsilon = 4$ we get $r = 64$ and $\delta = 0.185241912$. Thus a single transaction can be parallelized over, *e.g.*, 64

cores of a multicore processor, yet with an assurance of 0.69 a leak of more than 4.2 bits will be detected[13]. In the ATM example, a Trojan could steal more than 4.2 bits of at most 3.22580645 credit card numbers[14] before being detected (in the expected case), and if it steals all 53 bits then it is very likely to be detected on the first try.

Interestingly, if information theft detection is defined in terms of Moskowitz and Kang's short message criterion [17], then the system designer need only set $\epsilon = 0$ and integrate over multiple transactions the product of the maximum number of bits leaked (δ) and the attacker probability of not getting caught, with the former diminishing as ρ, ρ^2, ρ^3, ..., and so on down to zero. This geometric series converges at 5.078124405 bits for $\epsilon = 0$ and $\rho = 0.9$. To the best of our knowledge the short message criterion has never been achieved in practice, yet a replay-based approach makes this possible.

ϵ	$c+1 = 2^\epsilon + 1$	**Number of replays (r)**	q	δ	$I(A;B) \leq \epsilon + \delta$
0	2	11	0.9904186332	0.5078124405	0.5078124405
1	3	15	0.9926638266	0.3888171917	1.3888171917
2	5	39	0.9972832036	0.1439902071	2.1439902071
3	9	78	0.9986384823	0.0721604367	3.0721604367
4	17	161	0.9993394326	0.0350100747	4.0350100747
5	33	338	0.9996851317	0.0166880219	5.0166880219

Table 2. Example bounds where $H(A) = 53$ and $\rho = 0.9$.

5.5 Performance

Any application of information flow security is a tradeoff between security and performance [17]. We did not take any specific performance measurements because this paper proposes a general approach and the performance tradeoff is specific to the particular application, but we will note two important points about performance. First, the replays are trivial to parallelize. Each replay can be executed on a different core of a multicore processor or in its own virtual machine. Second the performance tradeoff can be much better than many traditional approaches such as noise injection [18], because our repeated-deterministic-replay approach operates on the other side of Shannon's equation for the equivocation of a noisy channel.

$$H(A) - H(A|B) = H(B) - H(B|A) \qquad (= I(A;B))$$

Noise injection works on the right side of the equation and increases the noise $H(B|A)$, which can inadvertently increase $H(B)$ so that the trick is to minimize $H(B) - H(B|A)$ to the point where an equivocation $H(A|B)$ is forced on an attacker to get their signal through the noisy channel. Our approach directly forces

[13] Note that $\epsilon + \delta < 4.2$.

[14] This can be calculated with a geometric series.

an equivocation of $H(A|B)$ on the attacker if they want to pass the hypothesis test. For $H(A) \ll H(B)$, which is typical of many common scenarios such as A is a credit card ($H(A) = 53$) and B is all network output (quite possibly $H(B)$ is in the millions or billions), working on the left side of the equivocation equation is orders of magnitude more efficient. Furthermore, noise injection requires that the randomly injected noise be unknown and unpredictable to the attacker, because if they can predict the noise then they can greatly increase their channel capacity [19]. Thus, our intrusion detection approach to information flow security is more practical than traditional approaches.

6 Related Work

We classify related work into four categories: secure information flow, information-theoretic approaches, language-based information flow security, and covert channel analysis.

6.1 Secure Information Flow

TightLip [11] detects application leaks of confidential information by spawning a "doppleganger" process with scrubbed inputs and comparing the output of the application process and the doppleganger process. This allows for low-overhead detection of confidentiality policy breaches for unmodified applications with minor changes to the operating system. Because TightLip focuses on applications with bugs and misconfigurations that leak confidential information it is implemented at the system call level. Our work, in contrast, can be applied to a full system in a virtual machine to bound information leakage for arbitrary malicious code such as a Trojan installed on the system, which is a more challenging problem. Also, when the basic concept of redundancy involves more than two versions of the process or system, the probability of getting the same output by chance is not geometrically distributed because the events of seeing each new output are not independent, which is why our approach is based on hypothesis testing and not simply comparing two versions of the output.

The RIFLE project [20] adds dynamic information flow mechanisms to the Itanium processor. The current implementation of RIFLE is conservative in marking information flow and relies on static analysis to reduce the marking of false information flows. Dynamic information flow systems always require static analysis in practice. Fenton's data mark machine [21, 22], for example, cannot support variable data marks and thus in practice would require the compiler to know the security class of every data object at each possible program point so that it can be placed in the appropriate register. Thus, both RIFLE and Fenton's data mark machine rely on static analysis for enforcing information flow security, which has its own set of limitations.

Both Denning [23] and Bishop [24] give a good summary of the challenges of enforcing information flow policies, either statically or dynamically. The common theme is that, for confidentiality policies, what could have happened carries as

much information as what did happen (demonstrated by the existence of implicit flows and covert channels). Our approach explicitly goes back to the beginning of a transaction and measures what could have happened.

6.2 Information Theory

Our own work was inspired by the idea of using entropy analysis to both understand and detect malicious software such as worms [25]. The idea of applying the information theory that Shannon [3] developed for the theory of communication to information flow security is not new. What is novel about our work is the observation that entropy can be controlled and measured using checkpointing and deterministic replay, even for commodity systems. A related work on deterministic replay is ReVirt [26], where the purpose of checkpointing and replaying is to analyze intrusions.

Wittbold and Johnson [27] provide a theoretical approach to information flow security and covert channels that is entirely from an information-theoretic perspective. Gray [28] also addresses covert channels with information theory, but observes that, "we need to provide a connection between source code and [the] system model." The architecture we present in this paper provides that connection by replaying the system, thus using the system as its own specification. Gianvecchio and Wang [29] take an entropy-based approach to detecting covert channels on a network. Köpf and Basin [30] develop quantitative, information-theoretic metrics for side-channel attacks.

The work of Browne [31–33] is the most relevant to our work in terms of the underlying ideas. The Turing test for information flow [32] represents entropy as uncertainty about the mathematical definition of a system, rather than true uncertainty. This observation is important for covert channel analysis. Another important observation [31] is that noise-effect entropy is not unique. This becomes a problem when a CPU event such as a hard drive request or keyboard interrupt depends on the high input A. Yumerefendi et al. [11] found that many useful applications of confidentiality policy enforcement do not have this problem, and for future work we plan to explore the possibility of branching the log file whenever nondeterministic events differ during replay to build an "entropy tree" and preserve deterministic replay under such a scenario.

6.3 Language-based Information Flow Security

Language-based information flow security is a well-developed area and we will refer the reader to a survey paper by Sabelfeld and Myers [34]. Here we will only point out some interesting results related to non-interference or practical implementations of information flow policies. JFlow [35] checks information flow of Java programs, mostly statically. A more recent development is the link between information theory and language-based information flow security, including the ability to handle loops [36]. This can allow a static calculation of the rate of information leakage rather than a binary answer about whether information leaks or not. McCamant and Ernst [37] develop a simulation-based proof technique

based on the same basic idea as TightLip [11], namely comparing the outputs of a program with access to secret information to the outputs of a version of the program without access to the secret information. More recently, the same authors demonstrate that quantitative information flow can be calculated as a network flow capacity [38].

6.4 Covert Channel Analysis

The "light pink book" [39] in the rainbow series is an excellent resource on the difficulties of designing a systematic covert channel [40, 41] analysis. See also McHugh's chapter in the Handbook for the Computer Security Classification of Trusted Systems [42] and Millen's position paper [43]. Kemmerer [44] developed the shared resource matrix methodology to help identify all covert channels as a transitive closure of subjects and their ability to read or modify objects. This methodology has been very successful due to its generality and because it can be applied to different specifications or implementations at every design stage. The limitation of this technique is that it requires all objects to be enumerated, and covert channels often use variables not normally viewed as data objects. A more attractive method for addressing covert channels in our repeated-deterministic-replay approach is due to Wray [4]. Wray's analysis can be applied to any covert channel with timing characteristics.

7 Conclusion

We described an approach to bound the mutual information between high input A and low output B for an arbitrary system S. We showed that, through a two-pronged approach of hypothesis testing and information theory, it is possible to bound $I(A;B)$ even for an attacker with an astronomical number of states and an arbitrary choice for the distribution of those states, assuming the attacker must have some probability ρ of not being detected. We expect that this general approach will lead to more secure systems and more fruitful analysis techniques for a variety of applications.

For future work, because covert channels are only possible through $I(L;A)$, it should be possible to address them exhaustively for real, interesting applications. For example, most nondeterministic events need not depend on A (*i.e.*, there is no reason why a different hard drive block should be requested when a different credit card number is entered), so that any attempt to exploit a covert channel can be detected as a difference in L during replay. We plan to explore this for specific applications in the future.

For some real applications, nondeterminism does depend on high input. If, for example, A is a filename of a file to be accessed then we must still be able to do replay and measure even when the log file will differ. We plan to explore the possibility of building "entropy trees" by branching the log file when nondeterministic events differ during replay, and the issues raised by this.

8 Acknowledgements

We would like to thank many who provided comments and suggestions on earlier ideas related to this work, including Norm Matloff and Richard Kemmerer. We are also very grateful to the editorial board and the anonymous reviewers of the special issue on Security in Computing. This work was supported by ETRI, Intel, NSF CyberTrust Grant No. 0627749, and US Air Force grant FA9550-07-1-0532.

References

1. Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., Weaver, N.: Inside the Slammer worm. IEEE Security and Privacy **1**(4) (2003) 33–39
2. Sarangi, S.R., Greskamp, B., Torrellas, J.: CADRE: Cycle-Accurate Deterministic Replay for Hardware Debugging. In: DSN '06: Proceedings of the International Conference on Dependable Systems and Networks (DSN'06), Washington, DC, USA, IEEE Computer Society (2006) 301–312
3. Shannon, C.E., Weaver, W.: The Mathematical Theory of Communication. University of Illinois Press, Urbana, Illinois (1949)
4. Wray, J.C.: An analysis of covert timing channels. In: IEEE Symposium on Security and Privacy. (1991) 2–7
5. General William T. Sherman, as quoted in B. H. Liddell Hart, *Strategy*, second revised edition
6. Young, A., Yung, M.: Malicious Cryptography: Exposing Cryptovirology. Wiley Publishing, Inc. (2004)
7. Song, D.X., Wagner, D., Tian, X.: Timing analysis of keystrokes and timing attacks on SSH. In: USENIX Security Symposium 2001. (2001)
8. Kuhn, M.G.: Optical time-domain eavesdropping risks of CRT displays. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy. (2002) 3–18
9. Kohno, T., Broido, A., Claffy, K.C.: Remote Physical Device Fingerprinting. IEEE Symposium on Security and Privacy (May 2005)
10. Wang, Z., Lee, R.B.: New cache designs for thwarting software cache-based side channel attacks. SIGARCH Comput. Archit. News **35**(2) (2007) 494–505
11. Yumerefendi, A., Mickle, B., Cox, L.P.: Tightlip: Keeping applications from spilling the beans. In: Networked Systems Design and Implementation (NSDI). (2007)
12. Goguen, J.A., Meseguer, J.: Security policies and security models. In: IEEE Symposium on Security and Privacy. (1982) 11–20
13. Goguen, J.A., Meseguer, J.: Unwinding and inference control. In: IEEE Symposium on Security and Privacy. (1984) 75–86
14. de Oliveira, D.A.S., Crandall, J.R., Wassermann, G., Su, Z., Wu, S.F., Chong, F.T.: ExecRecorder: VM-based full-system replay for attack analysis and system recovery. In: Workshop on Architectural and System Support for Improving Software Dependability, San Jose, CA (October 2006)
15. The OpenSSL Project: `http://www.openssl.org/`.
16. Clarkson, M.R., Myers, A.C., Schneider, F.B.: Belief in information flow. In: CSFW '05: Proceedings of the 18th IEEE Computer Security Foundations Workshop (CSFW'05), Washington, DC, USA, IEEE Computer Society (2005) 31–45
17. Moskowitz, I.S., Kang, M.H.: Covert channels - here to stay? In: Compass'94: 9th Annual Conference on Computer Assurance, Gaithersburg, MD, National Institute of Standards and Technology (1994) 235–244

18. Kang, M.H., Moskowitz, I.S.: A pump for rapid, reliable, secure communication. In: CCS '93: Proceedings of the 1st ACM conference on Computer and Communications Security, New York, NY, USA, ACM Press (1993) 119–129
19. Costa, M.: Writing on dirty paper (corresp.). IEEE Transactions on Information Theory **29**(3) (1983) 439–441
20. Vachharajani, N., Bridges, M.J., Chang, J., Rangan, R., Ottoni, G., Blome, J.A., Reis, G.A., Vachharajani, M., August, D.I.: RIFLE: An architectural framework for user-centric information-flow security. In: Proceedings of the 37th International Symposium on Microarchitecture (MICRO). (December 2004)
21. Fenton, J.S.: Information protection systems. In: Ph.D. Thesis, University of Cambridge. (1973)
22. Fenton, J.S.: Memoryless subsystems. The Computer Journal **17**(2) (1974) 143–147
23. Denning, D.E.R.: Cryptography and Data Security. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1982)
24. Bishop, M.: Computer Security: Art and Science. Addison-Wesley (2003)
25. Kumar, A., Paxson, V., Weaver, N.: Exploiting underlying structure for detailed reconstruction of an internet-scale event. In: IMC '05: Proceedings of the 5th ACM SIGCOMM on Internet measurement, New York, NY, USA, ACM Press (2006)
26. Dunlap, G.W., King, S.T., Cinar, S., Basrai, M.A., Chen, P.M.: ReVirt: Enabling intrusion analysis through virtual-machine logging and replay. SIGOPS Oper. Syst. Rev. **36**(SI) (2002) 211–224
27. Wittbold, J.T., Johnson, D.M.: Information flow in nondeterministic systems. In: IEEE Symposium on Security and Privacy. (1990) 144–161
28. Gray III, J.W.: Toward a mathematical foundation for information flow security. In: IEEE Symposium on Security and Privacy. (1991) 21–35
29. Gianvecchio, S., Wang, H.: Detecting covert timing channels: an entropy-based approach. In: CCS '07: Proceedings of the 14th ACM conference on Computer and Communications Security, New York, NY, USA, ACM (2007) 307–316
30. Köpf, B., Basin, D.: An information-theoretic model for adaptive side-channel attacks. In: CCS '07: Proceedings of the 14th ACM conference on Computer and Communications Security, New York, NY, USA, ACM (2007) 286–296
31. Browne, R.: An entropy conservation law for testing the completeness of covert channel analysis. In: CCS '94: Proceedings of the 2nd ACM Conference on Computer and Communications Security, New York, NY, USA, ACM Press (1994) 270–281
32. Browne, R.: The turing test and non-information flow. In: IEEE Symposium on Security and Privacy. (1991) 373–388
33. Browne, R.: Mode security: An infrastructure for covert channel suppression. In: IEEE Symposium on Security and Privacy. (1999) 39–55
34. Sabelfeld, A., Myers, A.: Language-based information-flow security. IEEE Journal on Selected Areas in Communications, 21(1) (2003)
35. Myers, A.C.: JFlow: Practical mostly-static information flow control. In: POPL '99: Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, New York, NY, USA, ACM Press (1999)
36. Malacaria, P.: Assessing security threats of looping constructs. In: POPL '07: Proceedings of the 34th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, New York, NY, USA, ACM Press (2007)
37. McCamant, S., Ernst, M.D.: A simulation-based proof technique for dynamic information flow. In: PLAS 2007: ACM SIGPLAN Workshop on Programming Languages and Analysis for Security, San Diego, California, USA (June 14, 2007)

38. McCamant, S., Ernst, M.D.: Quantitative information flow as network flow capacity. In: Proceedings of the ACM SIGPLAN 2008 Conference on Programming Language Design and Implementation, Tucson, AZ, USA (June 9–11, 2008)
39. Light Pink Book: A guide to understanding covert channel analysis of trusted systems, version 1. NCSC-TG-030, Library No. S-240,572 (November 1993) TCSEC Rainbow Series Library.
40. Lampson, B.W.: A note on the confinement problem. Communications of the ACM **16**(10) (1973) 613–615
41. Lipner, S.B.: A comment on the confinement problem. In: SOSP '75: Proceedings of the fifth ACM Symposium on Operating Systems Principles, New York, NY, USA, ACM Press (1975) 192–196
42. McHugh, J.: Covert channel analysis (1995)
43. Millen, J.K.: 20 years of covert channel modeling and analysis. In: IEEE Symposium on Security and Privacy. (1999) 113–114
44. Kemmerer, R.A.: Shared resource matrix methodology: an approach to identifying storage and timing channels. ACM Trans. Comput. Syst. **1**(3) (1983) 256–277

Appendix: Bound q as a Function of ρ

Proposition Let c and r be positive constants, ρ a real number such that $1 > \rho > \frac{c^r}{(c+1)^{r-1}}$. Then there exists q such that

$$\left(\rho - \frac{c^{r+1}}{(c+1)^r}\right)^{1/r} < q < 1 \text{ and } c\left(1 - \frac{q}{c}\right)^r + q^r = \rho.$$

Proof Set $f(x) = c\left(1 - \frac{x}{c}\right)^r + x^r$. Then $f'(x) = -r\left(1 - \frac{x}{c}\right)^{r-1} + rx^{r-1}$, so f has exactly one positive critical point x_0 where

$$x_0 = \left(1 - \frac{x_0}{c}\right), \text{ so that } x_0\left(1 + \frac{1}{c}\right) = 1 \text{ and } x_0 = \frac{c}{c+1}.$$

x_0 is a minimum (as one can see by taking the second derivative or simply noting that $f \to \infty$ as $x \to \infty$), and

$$f(x_0) = f(\frac{c}{c+1}) = c\left(1 - \frac{1}{c+1}\right)^r + \left(\frac{c}{c+1}\right)^r =$$

$$(c+1)\left(\frac{c}{c+1}\right)^r = \frac{c^r}{(c+1)^{r-1}}.$$

As $f(x_0) < \rho < 1 < f(1)$ and f is continuous, $f(q) = \rho$ for some q between x_0 and 1. Next, note that $f'(x) < rx^{r-1}$ for all positive x; therefore f grows less slowly than x^r. Thus for $x > x_0$, $f(x) < x^r + f(x_0) - x_0^r$, by the fact that the two

sides in the last inequality are equal at $x = x_0$ and the growth-rate observation above. This gives, for $x > x_0$,

$$f(x) < x^r + \frac{c^r}{(c+1)^{r-1}} - \left(\frac{c}{c+1}\right)^r = x^r + \frac{c^r(c+1) - c^r}{(c+1)^r}.$$

In particular, then, for the value q such that $f(q) = \rho$,

$$\rho < q^r + \frac{c^r(c+1) - c^r}{(c+1)^r} \Rightarrow \left(\rho - \frac{c^{r+1}}{(c+1)^r}\right)^{1/r} < q,$$

as desired.