

DECLARATION OF Z. MORLEY MAO, Ph.D.

I, Z. Morley Mao, do hereby state that:

1. I am currently a Professor in the Department of Electrical Engineering and Computer Science at the University of Michigan. My office address is 2260 Hayward Street, 4629 Computer Science Division, Department of EECS, University of Michigan, Ann Arbor, MI 48109-2121. I conduct research focused on network protocol and systems security, ML systems security, securing connected automated vehicles, Smart Home security, and mobile and computer networking issues. I teach classes on various subjects, including computer security, distributed systems, and computer networks.

2. In connection with this declaration, I am being compensated for my time at my standard rate of \$525 per hour. I do not have a financial interest in this action or its outcome, and my compensation in this matter is not based in any way on the substance of my declaration rendered here or the outcome of this case.

3. I received my Ph.D. in Computer Science from the University of California at Berkeley in December 2003, where I focused on network systems and mobile performance. My dissertation was titled "Solving the Interdomain Routing Puzzle." I also received a Master's Degree in Computer Science from the University of California at Berkeley in December 2000, and a B.S. in Electrical Engineering and Computer Science from the University of California at Berkeley in May 1998.

4. After obtaining my Ph.D., I became an Assistant Professor at the University of Michigan, EECS Department. In September 2010, I was promoted to Associate Professor. And in September 2016, I was promoted to Professor. Since working at the University of Michigan, I have led the Robustnet networking and network security group.

5. Over the years, I have authored and co-authored well over 200 published works relating to various topics, including network security, Internet routing, and mobile computing. These published works include:

- GPS Spoofing Attack Detection on Intersection Movement Assist using One-Class Classification, J Ying, Y Feng, QA Chen, ZM Mao, *ISOC Symposium on Vehicle Security and Privacy (VehicleSec) (2023)*.
- Accurate real-time identification of IP prefix hijacking, X Hu, ZM Mao, *2007 IEEE Symposium on Security and Privacy (SP'07), 3-17 (2007)*.
- Off-path TCP sequence number inference attack-how firewall middleboxes reduce security, Z Qian, ZM Mao, *2012 IEEE Symposium on Security and Privacy, 347-361 (2012)*.
- Kratos: Discovering Inconsistent Security Policy Enforcement in the Android Framework, Y Shao, QA Chen, ZM Mao, J Ott, Z Qian, *NDSS (2016)*.

6. In general, when working on my research, I work with any collaborators (e.g., co-authors) to identify research ideas, analyze work that has been done in the field, conduct research, and write a publication describing the nature of the research and the results. The purpose of these publications is to allow others in the field to review our work and build on our ideas.

7. It is common that my publications will contain citations to other works in the field. Indeed, it is considered best practice to cite other relevant works to demonstrate the differences from and relations to prior work. For example, I advise my graduate students to search for, review, and cite, if appropriate, other works in the field when working on publications.

8. When looking for other works in the field, I frequently rely on standard databases used in the field to identify those works. These include public and subscription-based databases. Currently, some examples of resources I use include Google Scholar. In the 2004-2005 time period when I was at the University of Michigan, I was using resources such as ACM Digital Library and IEEE Xplore Digital Library.

9. In addition to the above listed papers, I am one of the co-authors of the reference King S., et al., *CIDS: Causality-based Intrusion Detection System* (the “King Reference”). A copy of the King Reference is attached to this declaration as Exhibit A.

10. The King Reference discusses an intrusion detection system:

This paper presents a new style of intrusion detection system, CIDS, that links network and host based intrusion detection systems together using causal dependencies. This combination provides a number of benefits. First, combining alerts reduces false positives for both network and host based approaches. Second, host based intrusion detection systems have significantly less data to process since they only act on processes and files that are causally linked to a suspicious network packet. Third, alarms are associated with a specific network service, allowing system administrators to act accordingly after detecting a compromise. To show the utility of this new system, we describe how several existing intrusion detection systems benefit from using our platform, and we introduce a new host based intrusion detection system that leverages the full power of causal dependency tracking.

Ex. A, Abstract.

11. In writing the King Reference, we followed the general process described above, including looking at other works in the field. For example, the King Reference specifically discusses other types of systems that combine network alerts with host alarms. Ex. A, pg. 2. The King Reference also discusses UCLog and its linking of alarms using temporal data. *Id.*

12. As is standard practice, when discussing a specific system, we provided a citation to that system. When discussing UCLog, we cited to the following paper discussing UCLog:

Z. Li, J. Taylor, E. Partridge, Y. Zhou, W. Yurcik, C. Abad, J. J. Barlow, and J. Rosendale. UCLog: A Unified, Correlated Logging Architecture for Intrusion Detection. In *Proceedings of the 12th International Conference on Telecommunication Systems - Modeling and Analysis (ICTSM)*, 2004. (the “Li Reference”).

Ex. A, pg. 7 (citation 14).

13. The King Reference which was originally published in the University of Michigan, Computer Science and Engineering Technical Reports.

14. A technical report (or “Tech Report”) associated with an institution such as the University of Michigan is to create an official public record of the work without requiring a formal review process that is associated with publishing at a conference or a journal. One use of Tech Report is to create an official timestamp for the work before it’s officially published at a peer-reviewed venue (i.e., conference or journal) to establish an ordering compared to other closely related work that may occur at similar times. The Tech Report format also allows others to cite this work in their research papers, as it is an official record of this work.

15. The purpose of the Tech Reports webpage on the Computer Science and Engineering Department’s website is to catalogue and make publicly known ongoing research being performed by the students and faculty of the Department.

16. Others in academia and industry would frequently review the University of Michigan’s Tech Reports webpage to understand the research being performed at the University of Michigan.

17. We submitted the King Reference as a Tech Report. At the time, it was to create an official record of this work with a timestamp before we find a suitable peer-reviewed venue, i.e., a conference, for publishing this work.

18. The King Reference was then published on the University of Michigan’s website for Tech Reports. See Exhibit B.¹

¹ Exhibit B was obtained at the following URL: (https://www.eecs.umich.edu/eecs/etc/research/techreports/cse_tr/database/reports.cgi?04), and identifies the date of publication for the King Reference as August, 2004.

19. The Internet Archive also has an archived a version of the University of Michigan's website from September 2004 that indicates the King Reference was published online at the University of Michigan's website in August 2004. See Exhibit C;² *see also* Exhibit D.³

20. Because we cited to the Li Reference in our paper that was submitted and published online at the University of Michigan's website in August 2004, this means at least one of my co-authors and/or myself was able to find the Li Reference when looking for other works in the field related to intrusion detection and prevention. We would not have cited to a reference where at least one, it not more, of the King Reference authors reviewed the citation. Thus, based on my standard practice of writing articles, at least one of my co-authors and/or myself found and reviewed the Li Reference before we submitted the King Reference as a Tech Report in August 2004.

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct.

Executed on: October 11, 2023

By:



Z. Morley Mao, Ph.D.

² Exhibit C was obtained from the Wayback Machine at the following URL:

(https://web.archive.org/web/20040907143957/http://www.eecs.umich.edu/eecs/research/techreports/cse_tr/database/reports.cgi?04) (indicating that the snapshot was taken on September 7, 2004).

³ Exhibit D was obtained from the Wayback Machine at the following URL:

(<https://web.archive.org/web/20040908025643/http://www.eecs.umich.edu/techreports/cse/2004/CSE-TR-496-04.pdf>) (indicating that the snapshot was taken on September 8, 2004).

EXHIBIT A

CIDS: Causality-based Intrusion Detection System

Samuel T. King, Z. Morley Mao, and Peter M. Chen
Department of Electrical Engineering and Computer Science
University of Michigan
 {kingst, zmao, pmchen}@umich.edu

Abstract—This paper presents a new style of intrusion detection system, CIDS, that links network and host based intrusion detection systems together using causal dependencies. This combination provides a number of benefits. First, combining alerts reduces false positives for both network and host based approaches. Second, host based intrusion detection systems have significantly less data to process since they only act on processes and files that are causally linked to a suspicious network packet. Third, alarms are associated with a specific network service, allowing system administrators to act accordingly after detecting a compromise.

To show the utility of this new system, we describe how several existing intrusion detection systems benefit from using our platform, and we introduce a new host based intrusion detection system that leverages the full power of causal dependency tracking.

I. INTRODUCTION

Significant time and resources are spent making software more secure, yet the Internet continues to be plagued by malicious activity. As a result, even the most diligent system administrators must cope with the reality of computer break-ins. One way system administrators combat attacks is by using intrusion detection systems (IDS). There are a number of intrusion detection systems available at both the network and host level, and each type balances different advantages and disadvantages, but none are perfect. Network IDS has the advantage of a network-wide view for detecting anomalous traffic patterns. Host IDS has detailed knowledge of application behavior.

We propose combining network based intrusion detection systems (NIDS) with host based intrusion detection systems (HIDS) to increase the effectiveness of both. Specifically, when a NIDS detects a suspect packet, we track the effects of that packet on the host by following the causal dependencies that result from an application reading or writing the packet. Tracking causal dependencies results in a set of processes and files on the host that are affected by that network packet. Then, host based intrusion detection techniques are applied to the set of files and processes found to determine if the packet did in fact lead a successful break-in. We call this new method of intrusion detection CIDS.

CIDS provides a number of benefits when compared to existing methods. First, CIDS significantly reduces the number of false positives generated by both signature and anomaly based intrusion detection systems. Second, CIDS cuts down the amount of data HIDS have to examine by only highlighting the subset of processes and files that are affected by a specific network packet. Finally, CIDS aids in determining the exploited network service after detecting an attack.

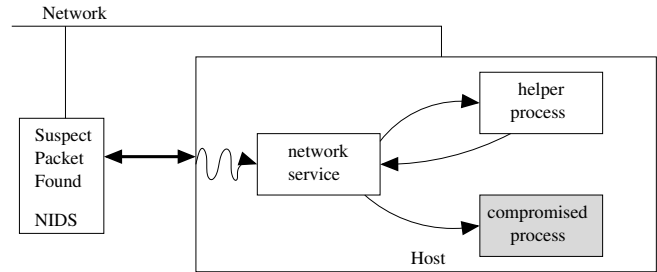


Fig. 1. Overall CIDS architecture. The NIDS queries the host upon detecting a suspicious network packet. The host then follows the causal dependencies resulting from the packet and applies HIDS rules to each of the files and processes in the set. The process deemed bad by the HIDS is highlighted.

This paper describes the benefits of combining host and network intrusion detection systems. Then, we discuss how causal tracking on the host works, followed by a description of several possible combinations of host and network intrusion detection systems. Finally, we describe a novel host anomaly detector, our prototype implementation, and CIDS performance for several attacks encountered on a honeypot system.

II. CIDS ARCHITECTURE

CIDS combines network and host based intrusion detection systems using the causal tracking available via BackTracker [11] and ForwardTracker. The NIDS identifies suspect packets, then queries the host that read the packet to determine if it led to a break-in (Figure 1). For now, packets are tracked using the TCP sequence numbers and IP addresses, but [20] outlines methods suitable for tracking general network packets in our system in the presence of IP spoofing.

CIDS tracks both incoming and outgoing network packets. Suspect incoming network packets are indicative of an attack on an existing network service. The effects of that packet are followed through the host and malicious activity is detected. In addition, suspicious outgoing network packets might show that a node has already been broken into. For example, if the NIDS detects a node launching attacks on other computers, CIDS is used to determine where the malicious outgoing packets came from.

This architecture is suitable for both real-time and forensic detection of intrusions, but there are trade-offs between the two methods. Real-time detection can stop an attacker before they leave lasting effects on the system. However, an attack that is detected at the network level, but has a delayed effect on the host, might be difficult to detect. Forensic, or post-intrusion analysis does not have trouble with delayed attacks.

However, sufficient amounts of network and host data must be stored to analyze the full attack.

III. BENEFITS OF CIDS

There are a number of benefits to combining NIDS and HIDS. In this section we evaluate the benefits of CIDS compared to stand-alone network based intrusion detection systems, host based intrusion detection systems, and the current efforts to combine the two.

A. NIDS

The CIDS architecture provides network based intrusion detection systems with an unparalleled level of semantic information. By querying the host, NIDS detect exactly which processes and files are directly affected by a specific network packet. Several different NIDS attempt to infer this information based solely on network data, where as CIDS actually monitors it directly at the host. This alleviates the need for inferring which network services handles a request and eliminates an entire class of NIDS evasion techniques. As a result, CIDS helps network based intrusion detection systems in a number of ways which we elaborate in Section V; one of the most promising is reducing false positives.

False positives are a problem with both signature and anomaly based NIDS. Signature based NIDS generate alarms regardless if the attack actually succeeded and anomaly based NIDS trigger false alarms by design. For example, we ran snort, an open source signature-based NIDS, on a /24 network in our department. After only three hours, we received alarms for every active IP address. None of the XXX machines that we were able to check had actually been broken into. Furthermore, we ran snort for 48 hours on the departmental web / MySQL server. Again there were 11956 alerts generated, none of which succeeded. In addition to tests run on our local network using snort, [16] discusses problems with commercial NIDS and false positives in detail.

These false alarms create an overwhelming amount of data for a system administrator to process and can eventually lead to a false sense of security. Although it appears that NIDS are difficult to use, they do provide some benefits. All systems had alarms associated with them, but only XX% of the packets were deemed bad. This significantly reduces the amount of data that host based intrusion detection systems must evaluate when using CIDS.

CIDS provide important feedback regarding the accuracy of the signature regardless of whether it is behavior based or signature rule based knowing the success of the attack. We propose to continuously refine the network signatures in NIDS using the host response information provided from HIDS.

B. HIDS

CIDS reduces the number of false positives generated by host based intrusion detection systems. Any processes or files examined by the HIDS directly result from a specific network packet that has been deemed bad. Combining the two alerts gives system administrators higher confidence that a break-in actually occurred.

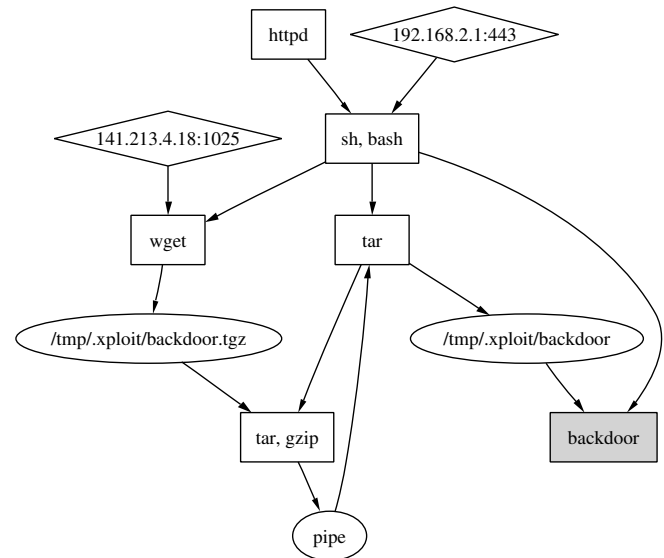


Fig. 2. Dependency graph for httpd attack. Processes are shown as boxes (labeled by program names called by execve during that process's lifetime); files are shown as ovals; sockets are shown as diamonds. BackTracker can also show process IDs and file inode numbers. The detection point is shaded. Note: httpd's parent processes (initlog, rc, init, swapper) are not shown in the interest of space. Please refer to [11] for examples of full BackTracker graphs.

In addition to reducing false positives, CIDS presents the HIDS with significantly less data to process. CIDS allows slower HIDS to be effective since it only highlights the subset of files and processes affected by the network packet. This subset is significantly smaller than the total activity on the system.

Finally, CIDS connects host alarms with specific network services. Some HIDS, like tripwire which periodically scans binaries looking for Trojans, detect intrusions but cannot connect the alarm with the actual break-in. CIDS provides this connection since actions detected are associated with a network packet, and thus a network service. Determining the exploited network service is important for system administrator so they can take action and either apply a patch to the service, or shut it down all together to avoid future compromises.

C. Combine IDS

Some commercial IDS vendors have combined network alerts with host alarms. However, they seem to simply combine alarms into a single administrative console. Because there is nothing other than IP address linking the alarms together, the false positive problem could become even worse since the administrator would view alarms from both the host and the network.

UCLog[14] links alarms using temporal data. However, the benefits from this are limited; it is possible to have two unrelated alarms close in time and delayed attacks would not be linked. Also, nothing links the alarm to a specific network service, leaving the possibility for future attacks.

IV. BACKTRACKER AND FORWARDTRACKER

BackTracker and ForwardTracker used within CIDS start from incoming and outgoing network packet respectively, and provide a graph of all relevant objects. In this section we describe how BackTracker and ForwardTracker use causal dependencies to link various operating system objects together. First, we describe BackTracker and causal dependencies. Second, we talk about ForwardTracker and how it varies from BackTracker. Please refer to [11] for a more detailed discussion of BackTracker.

A. BackTracker

BackTracker automatically identifies potential sequences of steps that occur in an intrusion. Starting with a single detection point (e.g., a suspicious outgoing network packet), BackTracker identifies files and processes that could have affected that detection point and displays chains of events in a dependency graph.

1) *Tracking Dependencies*: Dependency graphs are created by tracking various operating system events (system calls) and objects (files, processes). Events create causal dependencies between different operating system objects. For example, a `write` system call results in a causal dependency from the process that does the writing to the file that is written to. Also, `fork` system calls result in a causal dependency from the parent process to the child. Using causal events, BackTracker reduces the number of objects in a graph by focusing on events that are likely to be part of the attack. For example if a NIDS identifies a packet that is likely to be part of an attack, BackTracker starts with that packet and traverses backward in time beginning with the process that wrote the packet. BackTracker adds that process to the list of tainted objects and continues examining past events until the dependency graph is complete.

Figure 2 shows a BackTracker graph for an attack on a web server. The detection point for this particular attack is the `backdoor` application, which triggered an alarm from the HIDS being used. After detecting the intrusion, BackTracker uses this process as the starting point for the graph. Starting at the `backdoor` process and moving backward, BackTracker shows that the application runs using the executable file `/tmp/.xpl0it/backdoor`. This file is written by `tar` which unpacks an archive downloaded using `wget`. All of these applications launch from a `bash` shell which spawned from the `httpd` process as the result of a successful attack. By using causal dependencies, BackTracker filters out significant amounts of non-related events and objects present on the system during the attack, resulting in orders of magnitude less data to examine [11].

B. ForwardTracker

In addition to following causal dependencies backward in time, we modified BackTracker to track forward. In essence, rather than answering the question, “how did this get here”, ForwardTracker answers the question “what effect did this have”. Conceptually, ForwardTracker is similar to

BackTracker and they both use the same causal events. However, ForwardTracker starts from an *incoming* network packet, rather than an outgoing network packet, and tracks the effects of the process that read the packet.

V. NETWORK SENSORS

In this section, we describe in more details the state of the art of NIDS and the gained improvement using CIDS. Network intrusion detection inherently suffers from problems of false positives due to the impossibility of complete emulation of host behavior and the lack of full knowledge of which packets will actually be seen by the host. This is due to the fact that without external knowledge, NIDS does not typically know how hosts and applications are configured, how exactly applications will interpret a sequence of bytes, and how internal networks between the NIDS and the hosts are structured to determine reachability. These fundamental limitations have long been pointed out by Ptacek and Newsham [19]. In this section, we first describe several related efforts to remedy this fundamental flaw of NIDS, problems associated with these approaches, and how our approach of CIDS correct some of these problems. We use the term *monitor* to refer to the network intrusion detection system that observes the traffic to and from hosts and may alter the traffic as needed to reduce false alarms. We show that CIDS is complementary to existing approaches but enhances traditional NIDS with the ability to accurately disambiguate application protocol behavior. Given the ability to disambiguate relatively cheaply by the host to reduce false positives, CIDS is a promising new approach to improve NIDS by constructing a broader class of signatures to also lower false negatives.

A. Existing Approaches for Disambiguation

One existing proposal to deal with ambiguities at transport and network layer is traffic normalization [8]. It eliminates potential ambiguities before traffic is seen by the monitor to reduce evasion opportunities. For instance, it inserts a TCP “keep-alive” message when a RST packet is seen to ensure proper synchronization of the two TCP peers in the case of an open connection and to eliminate the ambiguity whether the TCP session is indeed closed. There are several semantic and performance problems associated with the approach of normalization. It undermines end-to-end protocol semantics; in effect it is still guessing how traffic *should* be seen by applications. This approach is very expensive, as it requires examining each packet possibly resulting in end-to-end performance slow down. The normalizer sits on the forwarding path and thus needs to be extremely reliable. It is susceptible to attacks such as stateholding and CPU overload attacks. Finally, traffic normalization is limited to ambiguities at only the network and transport layers; it can’t remove all ambiguities, e.g., those of applications semantics, which CIDS can address without any ambiguities. In effect, CIDS also encompasses the functionality of the traffic normalization without any of the above disadvantages, since the host sensors have direct access to protocol state above the network and transport layers.

Bifurcation analysis [17] is a related technique that splits the analysis context into multiple threads, one for each possible

interpretation. Analysis proceeds separately for each context in parallel. This can easily lead to the explosion of the number of analysis contexts, and is thus unscalable as a general disambiguation method.

A recent related work, Shield [25], uses host-based network filters targeting specific application vulnerabilities before software patches are properly installed. Positioned between the transport layer and applications, it filters traffic based on known vulnerabilities independent of exploits, thus greatly reducing false positives. The HIDS part of our CIDS framework is similar to Shield, but CIDS additionally makes use of alarms generated by NIDS to first identify suspicious traffic. Moreover, the ForwardTracker and BackTracker further disambiguate the behavior of the applications to reduce both false positive and false negatives of Shield which is complementary to our framework.

Traffic normalization relies on the specified transport and network protocol behavior as published by standards such as RFCs for the purpose of disambiguation. It may disrupt end-to-end protocol semantics, as not all implementations follow the standards and some ambiguities are inherently difficult to resolve due to differences in the semantics of application using TCP. For example, there exist application level parameters affecting the TCP/IP stack. The use of TCP “urgent” pointer can result in data delivered either via a signal or inline to the user process. Another related approach to deal with ambiguities or false positives is Active Mapping [21] which refers the technique of keeping a database cataloging the detailed behavior of individual end system protocol implementation and network topology. Based on the information in the database, the network monitor can resolve some but not all ambiguities at the transport and network layer. However, this approach is expensive and would require automatic update to the database, so that the information does not ever become out of date. Even if automation is possible, it is difficult to make it timely triggered by any changes relevant to the information in the database. Furthermore, for a large enterprise network, mapping individual hosts do not scale well. Information collected can be quickly out of date due to new software, OS upgrades, and network topology changes. Similar to traffic normalization, Active Mapping also only focuses on TCP/IP based ambiguities. It requires active probing. Arguable the information can be gathered via passive monitoring, but it would take much longer and may be less accurate. Other shortcomings of active mapping include user-controlled parameter in tcp stack, new semantics, non-deterministic packet drops, NAT, DHCP, TCP Wrappers, and potential attacks on the active mapper for the purpose of evasion.

Another proposal to reduce false positives of traditional NIDS is contextual signatures [22] focused on improving traditional string-based signature matching by augmenting the matching process with low-level context using regular expressions and high-level context. Examples include matching requests with replies, step-wise attacks, exploit scans. These types of signatures can make use of dependencies between existing signatures in so-called interdependent signatures. The disadvantages of these techniques include tedious manual instrumentation of signatures to consider additional context.

Such a tedious process can miss some important contexts and thus do not completely eliminate false positives. A signature consisting of a request and a reply for example can be easily evaded if the attacker manages to compromise the machine the single request and make the reply to evade the signature matching.

B. CIDS Advantages for NIDS

Given the shortcomings of the existing approaches to deal with the false positive issue of traditional NIDS, we now classify a set of attacks that would benefit from combining host and network IDS. One type are those OS or application version specific attacks that are difficult to generalize without trigger large number of false positives. These attacks depend on the OS or application configurations. To encompass all such attacks, the signatures or behavior-based alarms in the NIDS are required to be general, fundamentally resulting in many false positives. Active mapping can help deal with some of these problems; however, it is difficult for the database to be exhaustive. CIDS can be considered as a way to automatically update the Active Mapping database, whose entries should be timed out regularly to prevent staleness.

CIDS can handle those attacks much better than NIDS if attacks are more easily described by specifying which process is invoked in response to the sequence of network packets and host’s subsequent behavior. For example, polymorphic shellcode detection [2] is difficult due to the changing exploit signatures. The state of the art of recommended detection techniques [23] suggest to set NOP_NUMBER to detect no-effect instructions to be between 80 and 90 to avoid too many false positives. Such a setting is truly ad-hoc and can also result in false negatives. Attackers knowing these threshold settings can easily evade the detection. Using CIDS, a general signature for detecting NOPS section is used and the host sensor can subsequently eliminate false positives by examining the host behavior. Similarly, attacks such as cross site scripting attacks [4] where malicious code is contained in HTML Tags in client Web requests are more easily disambiguated using CIDS given the inherent difficult of interpreting host’s response to a sequence of code.

Another type of attacks that CIDS deal better with are those targeted at services using non-fixed ports. For example, for services using port knocking to obscure the port numbers by providing a stealthy method of authentication and information transfer to a networked machine that has no open port, it is difficult to identify which application protocol will receive the observed network packets. With the assistance from the host sensors, the process and application is easily identified and NIDS can perform retrospective network-level detection to analyze the traffic when needed.

NIDS themselves are also inherently vulnerable to attacks. Adversaries can structure network traffic that require NIDS keep significant state, e.g., record information on multiple data packets in succession. This imposes much overhead on the Network IDS, making it vulnerable to resource exhaustion and crash attacks. NIDS can offload some of such state-intensive analysis to host sensors in order to improve robustness and accuracy. Even when NIDS are overloaded and overlooks

some of the suspicious traffic, it could transition in the default case of marking all traffic as potentially suspicious, leaving host sensors for further analysis. This will prevent attackers from successfully exploiting overload attacks.

One major advantage of CIDS is that we can now identify which attacks are actually successful. Those successful and failed attacks are invaluable for upgrading network signatures to refine the signature database of NIDS over time. It also provides a way to automatically generating context for contextual signatures.

VI. HOST SENSORS

We now examine from the perspective of host intrusion detection systems the benefit of using NIDS alarms. There are numerous types of HIDS used to detect compromises. Tripwire [10] detects modified system files, LIDS [9] prevents various system files from being modified, Sandboxing tools look for unusual system calls [18][5][1][7] or system call sequences [6], and [24][13][15] detect injected code attacks, just to name a few.

In this section, we introduce three example services that combine existing HIDS with causal tracking. These services show how existing HIDS can be enhanced by using CIDS. Then, we discuss a novel HIDS anomaly detector, CausalTrace, that leverages the full power of causal tracking rather than simply combining it with an existing HIDS.

A. Static Analysis

Static Analysis [24] statically determines all possible sequences of system calls an application can go through by forming a number of different models of the application. Static analysis is especially promising because it traverses all code paths at compile time and does not rely on profiling or system administrator configuration. However, the static models incur significant runtime overhead, making them infeasible for system-wide operation or on production level systems.

CIDS helps mitigate the runtime overhead of running static analysis. When the network component of CIDS highlights either an incoming or outgoing suspect network packet, ForwardTracker or BackTracker returns a set of causally linked processes and files. This set is several orders of magnitude smaller than the total number of objects on the system [11] and even separates out individual server instances. Static Analysis can then be applied to the subset of relevant objects, reducing the amount of data it must process. This reduction of data allows Static Analysis to perform system wide analysis or even work for enterprise servers despite the runtime overhead incurred for checking each individual process instance.

B. Sandboxing HIDS

There are currently a number of different sandboxing HIDS [18][5][1][7][6]. Sandboxing HIDS vary widely, but all share one common theme: detect unexpected system calls. Profiles of known good behavior are either specified by hand or through a training period, but each system monitors how applications and children processes interact with the underlying operating system.

This style of anomaly detector defines good behavior and triggers an alarm when bad behavior is encountered. Defining good behavior allows sandboxing HIDS to detect unknown attacks, but also allows for false positives. For example, if not all code paths are traversed while profiling, an alarm triggers when a new, yet correct, code path is taken for the first time.

CIDS reduces the number of false positives resulting from Sandboxing HIDS by linking the alarm with a network alert. If a suspect network packet causes a NIDS alarm, the effects of the packet are tracked on the host and the resulting set of files and processes are evaluated by the Sandboxing HIDS. Because BackTracker and ForwardTracker separate out only the processes and files directly affected by the specific network packet, system administrators have higher confidence that the suspect network packet actually lead to an intrusion and that the host alarm is a true positive.

C. LIDS

LIDS hardens the Linux kernel and grants root users fewer privileges than a standard root user in UNIX systems. LIDS restricts system wide activities and can be configured to disallow non-trusted kernel modules and prevent modifications of system binaries. This level of protection prevents attackers from installing Trojan binaries, even when they get root access to the machine. However, the detection occurs *downstream* from the intrusion. For example, if an attacker breaks into a web server, spawns a bash shell, and attempts to install a root kit using a cron script, LIDS will prevent the root kit from being installed but cannot determine which network service was exploited. Fortunately, CIDS is able track the effects of a network packet and can link the attempted root kit installation with an exploited network service.

D. CausalTrace

In addition to using existing methods of host intrusion detection, we developed a new host anomaly detector based on BackTracker and ForwardTracker graphs called CausalTrace. BackTracker and ForwardTracker graphs provide a system-wide view of the attack and capture information that is not available to other HIDS. For example, a typical HIDS would most likely allow network services to write files in /tmp. However, if an attacker corrupts data in /tmp that is used by other applications, typical HIDS do not follow the causal link. Fortunately, BackTracker and ForwardTracker graphs capture the tainting information and use it as a means for detecting suspicious activity. The main observation is that attacks on network services interact differently with the operating system than typical network services.

Typical network services tend to have specific interactions with the underlying operating system. For example `ftpd`, `smbd`, and other file servers read and write files; `httpd` reads files, launches CGI scripts, and communicates with databases; and instant messaging clients relay text messages to various servers. These interactions are captured by CausalTrace and used to constrain attackers. Figure 3 shows the ForwardTracker graph for a typical instance of a ftp server. The ftp server writes files and uses helper processes that pipe information back to

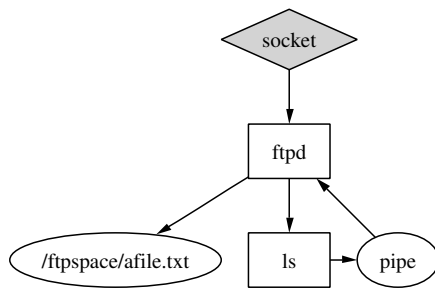


Fig. 3. ForwardTracker Graph for a typical instance of the wu.ftpd ftp server.

the main server process. As a result, a successful attack on the ftp server would be constrained to performing only the actions shown on this graph. Typical exploits are constrained to begin with (e.g., limited stack space for buffer overflows), so limiting attackers further helps prevent hackers from gaining full control of the system, even after a successful attack.

In addition to constraining hackers, CausalTrace detects attacks against CIDS itself. One possible way a hacker could attack CIDS is by forcing everything on the system to be tainted. However, this is not typical behavior for network services and is easily detected by CausalTrace.

1) *Prototype Implementation:* To implement CausalTrace, we use a set of policy rules for each process in a causal dependency graph. These rules define allowed causal events for each process and are used to enforce the constraints of a BackTracker or ForwardTracker graph. For example, the `ftpd` process shown in figure 3 would only be allowed to call `fork`, write to `/ftpspace/afile.txt` and read from a pipe. Also, the forked process could only call `exec` on `ls` and write to a pipe. As a result, constraints placed on the causal graph are almost identical to full graph matching, but algorithmically more efficient to compute.

To support CausalTrace, we implemented policies that are predicated based on the fork chain of each individual process. Applying different rules to children processes is similar to Sandboxing HIDS like MAPbox [1] and BlueBoX [5]. Child-boxes from MAPbox and inherited policies for BlueBoX apply different rules to children processes, however, our approach allows for even more fine grained control. For example, if a Web server uses `bash` for both a CGI script and a helper process for a different CGI script, the two `bash` instances have distinct privileges.

Furthermore, fork chains can have various root processes depending on the causal relationship. Most processes will have the specific network service for the root process in the fork chain. However, if a process that is not a direct descendant of the network service is causally related to the network service, it will have a different root process. Having a range of root processes allows CausalTrace to profile processes based on the full causal relationship.

As a result, existing processes within the causal graph are constrained using rules predicated on the location within the graph. If the behavior of these processes varies, an alarm is raised. Also, any unknown fork chains are flagged. This style of sandboxing is a good approximation of a full BackTracker

or ForwardTracker graph. Since files do not have rules applied to them, there is the possibility for ambiguities. However, there are still significant additional constraints placed on the system and it is effective at detecting attacks.

VII. REMAINING CHALLENGES

CIDS is a working prototype that is effective for many different cases. However, several unanswered research questions remain. First, we have to address network service interaction. For example, a single computer may double as a web server and a ftp server. Users upload web files using ftp, then the web server reads these files using the local file system. This scenario may lead to significant tainting of the system for certain network packets. One simple approach is to isolate various network services using virtual machines. Virtual machines add little or no overhead [12][3][26] and can be used to isolate the actions of network services from one another. This is just one possible solution, other methods might also be effective.

Many network services we examined (`ftp`, `smbd`, `named`, and `httpd`) have specific interactions with the operating system that can be summarized by a ForwardTracker or BackTracker graph. One exception is the `sshd` network service. `sshd` is a general purpose secure remote login application. Making an accurate ForwardTracker CausalTrace signature requires allowing behavior that is so general, it would be trivial for an attacker to hide within it. Fortunately, `sshd` is the only network service we encountered with this level of generality. There are heuristics that could allow `sshd` to benefit from the added constraints imposed by CIDS, but this is the topic of ongoing research.

VIII. CONCLUSIONS

Both host and network based intrusion detection systems are prone to having false positives. By using CIDS, network and host alarms are causally linked, giving system administrators more confidence that intrusions have actually occurred. NIDS are able to query the host and have access to a level of semantic information not previously available. HIDS have significantly less data to evaluate, opening the door for more sophisticated detection algorithms. By bridging the gap between network and host based intrusion detection systems, CIDS introduces a promising new technique in the defense of computer systems.

REFERENCES

- [1] A. Acharya and M. Raje. Mapbox: Using parameterized behavior classes to confine untrusted applications. In *Proceedings of the 9th USENIX Security Symposium*, August 2000.
- [2] K. H. at The SANS Institute. What is polymorphic shell code and what can it do? . http://www.sans.org/resources/idfaq/polymorphic_shell.php.
- [3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *Proceedings of the 2003 Symposium on Operating Systems Principles*, October 2003.
- [4] C. C. Center. CERT Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests. <http://www.cert.org/advisories/CA-2000-02.html>.
- [5] S. N. Chari and P.-C. Cheng. Bluebox: A policy-driven, host-based intrusion detection system. *ACM Trans. Inf. Syst. Secur.*, 6(2):173–200, 2003.

- [6] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for Unix processes. In *Proceedings of 1996 IEEE Symposium on Computer Security and Privacy*, pages 120–128, 1996.
- [7] I. Goldberg, D. Wagner, R. Thomas, and E. A. Brewer. A Secure Environment for Untrusted Helper Applications. In *Proceedings of the 1996 USENIX Security Symposium*, pages 1–13, July 1996.
- [8] M. Handley, C. Kreibich, and V. Paxson. Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics. In *Proc. USENIX Security Symposium 2001*.
- [9] X. Huang. Build a secure system with LIDS, 2000. http://www.lids.org/document/build_lids-0.2.html.
- [10] G. H. Kim and E. H. Spafford. The design and implementation of Tripwire: a file system integrity checker. In *Proceedings of 1994 ACM Conference on Computer and Communications Security (CCS)*, pages 18–29, November 1994.
- [11] S. T. King and P. M. Chen. Backtracking Intrusions. In *Proceedings of the 2003 Symposium on Operating Systems Principles (SOSP)*, pages 223–236, October 2003.
- [12] S. T. King, G. W. Dunlap, and P. M. Chen. Operating System Support for Virtual Machines. In *Proceedings of the 2003 USENIX Technical Conference*, pages 71–84, June 2003.
- [13] V. Kiriakos, D. Bruening, and S. Amarasinghe. Secure Execution Via Program Shepherd. In *Proceedings of the 2002 USENIX Security Symposium*, August 2002.
- [14] Z. Li, J. Taylor, E. Partridge, Y. Zhou, W. Yurcik, C. Abad, J. J. Barlow, and J. Rosendale. UCLog: A Unified, Correlated Logging Architecture for Intrusion Detection. In *Proceedings of the 12th International Conference on Telecommunication Systems - Modeling and Analysis (ICTSM)*, 2004.
- [15] G. C. Necula, S. McPeak, and W. Weimer. Ccured: type-safe retrofitting of legacy code. In *Proceedings of the 29th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 128–139. ACM Press, 2002.
- [16] D. Newman, J. Snyder, and R. Thayer. Crying wolf: False alarms hide attacks, 2002. <http://www.nwfusion.com/techinsider/2002/0624security1.html>.
- [17] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks*, 31(23-24):2435–2463, December 1999.
- [18] N. Provos. Improving host security with system call policies. In *Proceedings of the 12th USENIX Security Symposium*, August 2003.
- [19] T. H. Ptacek and T. N. Newsham. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. Technical report, Secure Networks, Inc., 1998.
- [20] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Network support for ip traceback. In *ACM/IEEE Transactions on Networking*, 9(3), June 2001.
- [21] U. Shankar and V. Paxson. Active Mapping: Resisting NIDS Evasion Without Altering Traffic. In *Proc. IEEE Symposium on Security and Privacy*.
- [22] R. Sommer and V. Paxson. Enhancing Byte-Level Network Intrusion Detection Signatures with Context. In *Proc. ACM CCS 2003*.
- [23] N. G. S. Technologies. Polymorphic Shellcodes vs. Application IDSs. http://www.cgisecurity.com/lib/polymorphic_shellcodes_vs_app_IDSs.PDF, 21 Jan 2002.
- [24] D. Wagner and D. Dean. Intrusion Detection via Static Analysis. In *Proceedings of 2001 IEEE Symposium on Computer Security and Privacy*, 2001.
- [25] H. J. Wang, C. Guo, D. R. Simon, and A. Zugenmaier. Shield: Vulnerability-Driven Network Filters for Preventing Known Vulnerability Exploits. In *Proc of ACM SIGCOMM 2004*.
- [26] A. Whitaker, M. Shaw, and S. D. Gribble. Scale and Performance in the Denali Isolation Kernel. In *Proceedings of the 2002 Symposium on Operating Systems Design and Implementation (OSDI)*, December 2002.

EXHIBIT B



CSE Research

Research Areas

Labs and Centers

Technical Reports

CSE Technical Reports Sorted by Technical Report Number

TR Number	Title	Authors	Date	Pages
CSE-TR-485-04	Infeasible Group Inversion and Broadcast Encryption	Irrer Lokam Opyrchal and Prakash	Jan, 04	13

Broadcast encryption allows a sender to broadcast messages in such a way that only the sender-specified subset of users can decrypt them. Over the years a number of schemes have appeared that solve different variants of the problem. The number of applications that could take advantage from an efficient broadcast encryption scheme has also grown and includes such examples as content-based publish subscribe systems. Such systems are characterized by messages going to arbitrary and rapidly changing subsets of users. Each message can potentially go any of the 2^{n-1} possible subsets. Collusion resistance and small resource requirements are necessary. In this paper we present a simple and efficient broadcast encryption scheme based on one-way group operations. A group operation $\$gpop\$$ is one-way if $\$x gpop y\$$ is easy to perform for any two group elements $\$x\$$ and $\$y\$$, but computing the inverse $\$inv\{x\}\$$ (w.r.t. $\$gpop\$$) is infeasible. Groups with infeasible inversion have recently received attention as a cryptographic primitive and their existence is still uncertain. Under the assumption that such groups exist, we prove the security of our broadcast encryption scheme and explore a few areas where such constructions might come from. Finally, we discuss some functions used in existing cryptosystems and compare them with one-way group operations.

CSE-TR-486-04	Coordinated Navigation of Multiple Independent Disk-Shaped Robots	Karagoz Bozma and Koditschek	Feb, 04	32
----------------------	---	------------------------------	---------	----

This paper addresses the coordinated navigation of multiple independently actuated disk-shaped robots - all placed within the

same disk-shaped workspace. We encode complete information about the goal, obstacles and workspace boundary using an artificial potential function over the cross product space of the robots' simultaneous configurations. The closed-loop dynamics governing the motion of each robot take the form of the appropriate projection of the gradient of this function. We show, with some reasonable restrictions on the allowable goal positions, that this function is an essential navigation function - a special type of artificial potential function that is ensured of connecting the kinematic planning with the dynamic execution in a correct manner. Hence, each robot is guaranteed of collision-free navigation to its destination from almost all initial free placements.

CSE-TR-487-04	Practical Slicing and Non-slicing Block-Packing without Simulated Annealing	Chan and Markov	Feb, 04	31
----------------------	---	-----------------	---------	----

We propose a new block-packer BloBB based on multi-level branch-and-bound. It is competitive with annealers in terms of runtime and solution quality. We empirically quantify the gap between optimal slicing and non-slicing floorplans by comparing optimal packings and best seen results. Most ongoing work deals with non-slicing packings, and implicitly assumes that best slicing packings are highly sub-optimal. Contrary to common belief, we show that the gap in optimal slicing and non-slicing packings is very small. Optimal slicing and non-slicing packings for apte, xerox and hp are reported. We extend BloBB to the block-packer CompaSS, that handles soft blocks. Optimal slicing packings for soft versions of apte, xerox and hp are reported. We discover that the soft versions of all MCNC benchmarks, except for apte, and all GSRC benchmarks can be packed with zero dead-space. Moreover, the aspect ratio bound $[0.5, 2]$ turns out to be not very restrictive, when area is concerned. Our heuristic slicing block-packer is able to pack with zero dead-space in most cases when we restrict the aspect ratio bound to $[0.57, 1.70]$. CompaSS improves the best published results for the ami49 X benchmarks suite, outperforming the leading multilevel annealer in runtime, solution quality and scalability. Additionally, realistic floorplans often have blocks with similar dimensions, if design blocks, such as memories, are reused. We show that this greatly reduces the complexity of block-packing.

CSE-TR-488-04	A Compressed Memory Hierarchy Using an	Hallnor and Reinhardt	Mar, 04	21
----------------------	--	-----------------------	---------	----

[Indirect Index Cache](#)

The large and growing impact of memory hierarchies on overall system performance compels designers to investigate innovative techniques to improve memory-system efficiency. We propose and analyze a memory hierarchy that increases both the effective capacity of memory structures and the effective bandwidth of interconnects by storing and transmitting data in a compressed form. Caches play a key role in hiding memory latencies. However, cache sizes are constrained by die area and cost. A caches effective size can be increased by storing compressed data, if the storage unused by a compressed block can be allocated to other blocks. We use a modified Indirect Index Cache to allocate variable amounts of storage to different blocks, depending on their compressibility. By coupling our compressed cache design with a similarly compressed main memory, we can easily transfer data between these structures in a compressed state, increasing the effective memory bus bandwidth. This optimization further improves performance when bus bandwidth is critical. Our simulation results, using the SPEC CPU2000 benchmarks, show that our design increases performance by up to 107% on some benchmarks while degrading performance by no more than 2% on others. Compressed bus transfers alone account for up to 59% of this improvement, with the remainder coming from increased effective cache capacity. As memory latencies increase, our design becomes even more beneficial.

CSE-TR-489-04	Network Overlay Construction under Limited End-to-End Addressability	Wenjie Wang, Cheng Jim ad Sugih Jamin	May, 2004	14
----------------------	--	---------------------------------------	-----------	----

Network overlay construction has found many applications in today's Internet, such as P2P networks [1][2], end-host multicast [3][4], and network security [5]. Many researchers have proposed solutions to building an overlay network among a given group of end-hosts. The usual overlay construction assumes two-way network communication — each host can initiate connections to and accept requests from other hosts. This is not always true on the Internet due to the use of Network Address Translation (NAT) and firewalls. Our experiments with one P2P file-sharing system revealed that over 34% of hosts were guarded hosts, i.e., hosts that cannot accept connections from other Internet hosts. The lack of two-way communication capability presents a challenge because only a subset of hosts can act as

overlay routers. Using a cluster-base approach, we design a new overlay construction mechanism called e^* , which organizes members based on reachability and intelligently select overlay routers to reduce end-to-end latencies on the overlay. Under realistic scenarios involving guarded hosts, e^* can reduce average end-to-end latency on the overlay by 28-61% compared to existing protocols. Furthermore, e^* significantly improves the worst case end-to-end latency on the overlay.

CSE-TR-489-04	Network Overlay Construction under Limited End-to-End Addressability	Wenjie Wang, Cheng Jim ad Sugih Jamin	May, 2004	14
----------------------	--	---------------------------------------	-----------	----

Network overlay construction has found many applications in today's Internet, such as P2P networks [1][2], end-host multicast [3][4], and network security [5]. Many researchers have proposed solutions to building an overlay network among a given group of end-hosts. The usual overlay construction assumes two-way network communication — each host can initiate connections to and accept requests from other hosts. This is not always true on the Internet due to the use of Network Address Translation (NAT) and firewalls. Our experiments with one P2P file-sharing system revealed that over 34% of hosts were guarded hosts, i.e., hosts that cannot accept connections from other Internet hosts. The lack of two-way communication capability presents a challenge because only a subset of hosts can act as overlay routers. Using a cluster-base approach, we design a new overlay construction mechanism called e^* , which organizes members based on reachability and intelligently select overlay routers to reduce end-to-end latencies on the overlay. Under realistic scenarios involving guarded hosts, e^* can reduce average end-to-end latency on the overlay by 28-61% compared to existing protocols. Furthermore, e^* significantly improves the worst case end-to-end latency on the overlay.

CSE-TR-490-04	Data Propagation in a Distributed File System	Kim, Noble, Chen, Tilbury	July, 2004	11
----------------------	---	---------------------------	------------	----

Distributed file systems benefit greatly from optimistic replication that allows clients to update any replica at any site. However, such

systems face a new dilemma: Once data is updated at a replica site, when should it be shipped to others? In conventional file system workloads, most data is read by its writer, so it needs to be shipped only for administrative reasons. Unfortunately, shipping on demand substantially penalizes those who do share, while shipping aggressively places undue load on the system, limiting scalability. This paper explores a mechanism to predict and support sharing in a wide area file system. This mechanism uses the observation that updates that invalidate cached objects elsewhere are likely to be shared, and they should be propagated. The mechanism is evaluated on its precision, recall, and F-measure over traces capturing live file system uses. It reduces data shipped by orders of magnitude compared to aggressive schemes, while reducing the performance penalty of on-demand shipment by nearly a factor of two.

CSE-TR-491-04	The Internet Motion Sensor: A distributed global scoped Internet threat monitoring system	Cooke, Bailey, Watson, Jahanian and Nazario	July, 2004	16
---------------	---	---	------------	----

Networks are increasingly subjected to a broad spectrum of threats that impact the reliability and availability of critical infrastructure. In response, researchers and network operators have increasingly relied on monitoring to characterize and track these threats. This paper introduces the Internet Motion Sensor (IMS), a globally scoped Internet threat monitoring system whose goal is to measure, characterize, and track threats. The dark address sensors in the IMS extend simple passive capture using a novel transport layer service emulation technique to elicit payloads across all services, thereby addressing the issue of service coverage. To achieve breadth of coverage, the IMS employs a distributed infrastructure and utilizes sensors that are aware of their address diversity and their position in the actively routed topology. Finally, the IMS uses an innovative signature encoding and data warehousing system combined with a hierarchical architecture to realize a system that is not only time and space efficient, but is also scalable to a global deployment. We explore the various architectural tradeoffs in the context of a 3 year deployment across multiple dark address blocks ranging in size from /24s to a /8. We show how the current architecture emulates services across a diverse set of routed and address topologies in a scalable manner. Results from three recent events are presented to illustrate the utility of such a system: the SCO Denial of Service attacks

(December, 2003), the Blaster worm (August, 2003), and the Bagle backdoor scanning efforts (March, 2004).

CSE-TR-492-04	<u>Pueblo: A Modern Pseudo-Boolean SAT Solver</u>	Sheini and Sakallah	July, 2004	11
----------------------	---	---------------------	------------	----

In this report we introduce a new SAT solver that integrates logic-based reasoning and integer programming methods to systems of CNF and PB constraints. Its novel features include an efficient PB literal watching strategy that takes advantage of the preponderance of unit-coefficient literals in most PB constraints. Additionally, the solver incorporates several PB learning methods that take advantage of the pruning power of PB constraints while minimizing their overhead. Empirical evidence suggests that such judicious injection of IP techniques can be quite effective in practice.

CSE-TR-493-04	<u>CIDS: Causality Based Intrusion Detection System</u>	King, Mao and Chen	August, 2004	6
----------------------	---	--------------------	--------------	---

This paper presents a new style of intrusion detection system, CIDS, that links network and host based intrusion detection systems together using causal dependencies. This combination provides a number of benefits. First, combining alerts reduces false positives for both network and host based approaches. Second, host based intrusion detection systems have significantly less data to process since they only act on processes and files that are causally linked to a suspicious network packet. Third, alarms are associated with a specific network service, allowing system administrators to act accordingly after detecting a compromise. To show the utility of this new system, we describe how several existing intrusion detection systems benefit from using our platform, and we introduce a new host based intrusion detection system that leverages the full power of causal dependency tracking.

CSE-TR-494-04	<u>What and How Much to Gain by Spectral Agility?</u>	Chou, Kim and Shin	August, 2004	26
----------------------	---	--------------------	--------------	----

Static spectrum allocation has resulted in low spectrum efficiency in licensed bands and poor performance of radio devices in crowded unlicensed bands. To remedy these problems, we exploit the concept of "spectral agility" such that radio devices can dynamically utilize

idle spectral bands. We establish a mathematical model for the performance gain made by spectral agility, and use the model to evaluate important performance metrics such as spectrum efficiency, throughput of a spectral-agile network, and packet blocking/waiting time of spectral-agile devices. We propose three basic mechanisms to realize spectral-agile networks: spectrum opportunity discovery, spectrum opportunity management, and spectrum use coordination. These mechanisms are implemented in the ns-2, and the control overhead incurred by using spectral agility is evaluated. Our simulation results have shown that the throughput of a spectral-agile network is improved by up to 90%, and the improvement is very close to the performance bound predicted by our analytical (mathematical) model. These results demonstrate and confirm the spectral agility's capability of improving spectral utilization in an efficient, distributed, and automatic manner.

CSE-TR-495-04	<u>Debugging operating systems with time-traveling virtual machines</u>	King, Dunlap, and Chen	August, 2004	14
----------------------	---	------------------------	--------------	----

Operating systems are among the most difficult of software systems to debug with traditional cyclic debugging. They are non-deterministic; they run for long periods of time; their state and code is large and complex; and their state is easily perturbed by the act of debugging. This paper describes a time-traveling virtual machine that overcomes many of the difficulties associated with debugging operating systems. By time travel, we mean the ability to navigate backward and forward arbitrarily through the execution history of a particular run and to replay arbitrary segments of the past execution. We integrate time travel into a general-purpose debugger to enable a programmer to debug an OS in reverse, implementing commands such as reverse breakpoint, reverse watchpoint, and reverse single step. The space and time overheads needed to support time travel are reasonable for debugging, and movements in time are fast enough to support interactive debugging. We demonstrate the value of our time-traveling virtual machine by using it to understand and fix several OS bugs that are difficult to find with standard debugging tools.

CSE-TR-496-04	<u>Enriching intrusion alerts through multi-host causality</u>	King, Mao, Lucchetti, and Chen	August, 2004	13
----------------------	--	--------------------------------	--------------	----

Current intrusion detection systems point out suspicious states or events but do not show how the suspicious state or events relate to other states or events in the system. We show how to enrich an IDS alert with information about how those alerts causally lead to or result from other events in the system. By enriching IDS alerts with this type of causal information, we can leverage existing IDS alerts to learn more about the suspected attack. Backward causal graphs can be used to find which host allowed a multi-hop attack (such as a worm) to enter a local network; forward causal graphs can be used to find the other hosts that were affected by the multi-hop attack. We demonstrate this use of causality on a local network by tracking the Slapper worm, a manual attack that spreads via several attack vectors, and an e-mail virus. Causality can also be used to correlate distinct network and host IDS alerts. We demonstrate this use of causality by correlating Snort and host IDS alerts to reduce false positives on a testbed system connected to the Internet.

CSE-TR-497-04	Design and Applications of a Virtual Context Architecture	Oehmke, Binkert, Reinhardt, Mudge	September, 2004	42
----------------------	---	--	-----------------	----

This paper proposes a new register-file architecture that virtualizes logical register contexts. This architecture makes the number of active register contexts—representing different threads or activation records— independent of the number of physical registers. The physical register file is treated as a cache of a potentially much larger memory-mapped logical register space. The implementation modifies the rename stage of the pipeline to trigger the movement of register values between the physical register file and the data cache. We exploit the fact that the logical register mapping can be easily updated—simply by changing the base memory pointer—to construct an efficient implementation of register windows. This reduces the execution time by 8% while generating 20% fewer data cache accesses. We also use the large logical register space to avoid the cost of a large physical register file normally required for a multithreaded processor, allowing us to create an SMT core with fewer physical registers than logical registers that still performs within 2% of the baseline. Finally, the two are combined to create a simultaneous multithreaded processor that supports register windows. This architecture achieves a 10% increase in performance over the baseline architecture even with fewer physical than logical registers while also reducing data cache bandwidth. Thus we are able to combine the advantages of register windows with multithreading.

CSE-TR-498-04	Plato: A Platform for Virtual Machine Services	King, Dunlap, and Chen	October, 2004	14
----------------------	--	------------------------	---------------	----

Virtual machines are being used to add new services to system level software. One challenge these virtual machine services face is the semantic gap between VM services and the machine-level interface exposed by the virtual machine monitor. Using the virtual machine monitor interface, VM services have access to hardware-level events like Ethernet packets or disk I/O. However, virtual machine services also benefit from guest software (software running inside the virtual machine) semantic information, like sockets and files. These abstractions are specific to the guest software context and are not exposed directly by the machine-level virtual machine monitor interface. Existing ways to bridge this semantic gap are either ad-hoc or use debuggers. Ad-hoc methods often lead to cutting-and-pasting large sections of the guest operating system to reconstruct its interpretation of the hardware level events. Debuggers add too much overhead for production environments. Both ad-hoc methods and debuggers could cause unwanted perturbations to the virtual system. To address these shortcomings, we developed a new platform for implementing virtual machine services: Plato. The goal of Plato is to make it easy and fast to develop and run new virtual machine service. Plato allows VM services to call guest kernel functions and access guest local variables, eliminating the need to cut-and-paste sections of the virtual machine source code. Plato provides a checkpoint/rollback facility that allows VM services to correct for undesired perturbations to the virtual state. Plato adds less than 5% overhead for a variety of macrobenchmarks.

CSE-TR-499-04	A Hybrid Honeypot Architecture for Scalable Network Monitoring	Bailey, Cooke, Watson, Jahanian and Provos	October, 2004	16
----------------------	--	--	---------------	----

To provide scalable, early warning and analysis of new Internet threats like worms or automated attacks, we propose a globally distributed, hybrid monitoring architecture that can capture and analyze new vulnerabilities and exploits as they occur. To achieve this, our architectures increases the exposure of high-interaction honeypots to these threats by employing low-interaction honeypots as frontend content filters. Host-based techniques capture relevant

details such as packet payload of attacks while network monitoring provides wide coverage for quick detection and assessment. To reduce the load of the backends, we filter prevalent content at the network frontends and use a novel handoff mechanism to enable interactions between network and host components. We use measurements from live networks over five months to demonstrate the effectiveness of content prevalence as a filtering mechanism. Combining these observations with laboratory measurements, we demonstrate that our hybrid architecture is effective in preserving the detail of a specific threat while still achieving performance and scalability. We illustrate the benefits of this framework by showing how it enables earlier, higher-confidence detection, more detailed forensics, and robust signatures for mitigation of threats.

CSE-TR- 500-04	Weakly Supervised Graph-Based Methods for Classification	Radev	October, 14 2004
---------------------------	--	-------	---------------------

We compare two weakly supervised graph-based classification algorithms: spectral partitioning and tripartite updating. We provide results from empirical tests on the problem of number classification. Our results indicate (a) that both methods require minimal labeled data, (b) that both methods scale well with the number of unlabeled examples, and (c) that tripartite updating outperforms spectral partitioning.

CSE-TR- 501-04	Media-Aware Rate Control	Wang, Banerjee and Jamin	November, 33 2004
---------------------------	--	--------------------------------	----------------------

Streaming media transfers over the Internet are expected to behave in a TCP friendly manner and react appropriately to congestion similar to the way TCP does. However, our work shows that existing "TCP-friendly" protocols are not necessarily "media friendly". In this paper, we present our findings on the impact of the TFRC protocol on streaming media quality. We propose the MARC (Media Aware Rate Control) protocol that, unlike TFRC, exhibits significant tolerance towards transient changes in background workload, while still maintaining TCP-friendliness.

CSE-TR- 502-04	Network Maps Beyond Connectivity	Jin, Wang	November, 18 2004
---------------------------	--	--------------	----------------------

and
Jamin

Knowing network topology is becoming increasingly important for a number of applications such as server placement and traceback of DDoS attacks. Recent works in modeling the Internet topology and constructing network maps have focused on the connectivity aspect. This paper describes the first study in incorporating connectivity, latency, and routing information all into a network map based on a large set of traceroute data. We highlight the common challenges in constructing such network maps and discuss our solutions. We evaluate our network map based on various Internet routing models proposed in the literature. The evaluation shows that, for those traceroute data that we are able to evaluate, at least 85% of computed hop-counts and latencies are within a factor of two of the actual values. Furthermore, we show that a flat routing model based on hop-count performs as well as more complicated policy routing models.

CSE-TR- 503-04	<u>Worm Hotspots: Explaining Non- Uniformity in Worm Targeting Behavior</u>	Cooke, Mao, and Jahanian	November, 17 2004
---------------------------	---	--------------------------------	----------------------

Long after the Blaster, Slammer/Sapphire, and CodeRedII worms caused significant worldwide disruptions, a huge number of infected hosts from these worms continue to probe the Internet today. This paper investigates hotspots (non-uniformities) in the targeting behavior of these important Internet worms. Recent data collected over the period of a month and a half using a distributed blackhole data collection infrastructure covering 18 networks including ISPs, enterprises, and academic networks show 75K Blaster infected hosts, 180K slammer infected hosts, and 55K CodeRedII hosts. We discover through detailed analysis how critical flaws and side effects in the targeting behavior lead to a significant bias for certain destination address blocks. In particular, we demonstrate three previously unexplored biases: a severely restricted initial random seed forcing infection attempts to certain blocks; flaws in the parameters of a random number generator making certain hosts cycle through limited target addresses; and the widespread use of private address space dramatically changing the targeting distribution of certain worms. A direct consequence of these biases is that certain blocks are subjected to far more infection attempts than others. We discuss the

implication of these hotspots on worm simulation and modeling, placement of blackhole sensors, worm detection and quarantine.

CSE-TR-504-04	Cooperative ReVirt: Adapting Message Logging for Intrusion Analysis	Basrai and Chen	Nov, 2004	11
----------------------	---	-----------------	-----------	----

Virtual-machine logging and replay enables system administrators to analyze intrusions more completely and with greater integrity than traditional system loggers. One challenge in these types of systems is the need to log a potentially large volume of network traffic. Cooperative ReVirt adds message-logging techniques to ReVirt to reduce the amount of network traffic that needs to be logged. Cooperative ReVirt adapts message-logging techniques to address the challenges of intrusion analysis, such as the need to work in the presence of network attacks and unreliable networks, the need to support asymmetric trust relationships among computers, and the need to support dynamic trust and traffic patterns. CooperativeReVirt is able to reduce the log volume needed to replay a computer by an average of 70% for a variety of distributed computing benchmarks, while adding less than 7% overhead. Measurements of a live network indicate that Cooperative ReVirt would be able to avoid logging 85% of the received network data.

CSE-TR-505-04	Analyzing NIC Overheads in Network-Intensive Workloads	Binkert, Hsu, Saidi, Dreslinski, Schultz and Reinhardt	December, 2004	23
----------------------	--	--	----------------	----

Modern high-bandwidth networks place a significant strain on host I/O subsystems. However, despite the practical ubiquity of TCP/IP over Ethernet for high-speed networking, the vast majority of end host networking research continues in the current paradigm of the network interface as a generic peripheral device. As a result, proposed optimizations focus on purely software changes, or on moving some of the computation from the primary CPU to the off-chip network interface controller (NIC). We look at an alternative approach: leave the kernel TCP/IP stack unchanged, but eliminate bottlenecks by closer attachment of the NIC to the CPU and memory system. To evaluate this approach, we have developed a simulation environment specifically targeted for networked systems. It simulates server and client systems along with a network in a single process. Full-system

simulation captures the execution of both application and OS code. Our model includes a detailed out-of-order CPU, event-driven memory hierarchy, and Ethernet interface device. Using this simulator, we find that tighter integration of the network interface can provide benefits in TCP/IP throughput and latency. We also see that the interaction of the NIC with the on-chip memory hierarchy has a greater impact on performance than the raw improvements in bandwidth and latency that come from integration.

[Technical Reports Page](#)

EXHIBIT C

CSE Technical Reports

Sorted by Technical Report Number

Technical Reports

TR Number	Title	Authors	Date	Pages
CSE-TR-485-04	<u>Infeasible Group Inversion and Broadcast Encryption</u>	Irrer Lokam Opyrchal and Prakash	Jan, 04	13
Broadcast encryption allows a sender to broadcast messages in such a way that only the sender-specified subset of users can decrypt them. Over the years a number of schemes have appeared that solve different variants of the problem. The number of applications that could take advantage from an efficient broadcast encryption scheme has also grown and includes such examples as content-based publish subscribe systems. Such systems are characterized by messages going to arbitrary and rapidly changing subsets of users. Each message can potentially go any of the 2^n-1 possible subsets. Collusion resistance and small resource requirements are necessary. In this paper we present a simple and efficient broadcast encryption scheme based on one-way group operations. A group operation $gpop$ is one-way if $x gpop y$ is easy to perform for any two group elements x and y, but computing the inverse $inv\{x\}$ (w.r.t. $gpop$) is infeasible. Groups with infeasible inversion have recently received attention as a cryptographic primitive and their existence is still uncertain. Under the assumption that such groups exist, we prove the security of our broadcast encryption scheme and explore a few areas where such constructions might come from. Finally, we discuss some functions used in existing cryptosystems and compare them with one-way group operations.				
CSE-TR-486-04	<u>Coordinated Navigation of Multiple Independent Disk-Shaped Robots</u>	Karagoz Bozma and Koditschek	Feb, 04	32
This paper addresses the coordinated navigation of multiple independently actuated disk-shaped robots - all placed within the same disk-shaped workspace. We encode complete information about the goal, obstacles and workspace boundary using an artificial potential function over the cross product space of the robots' simultaneous configurations. The closed-loop dynamics governing the motion of each robot take the form of the appropriate projection of the gradient of this function. We show, with some reasonable restrictions on the allowable goal positions, that this function is an essential navigation function - a special type of artificial potential function that is ensured of connecting the kinematic planning with the dynamic execution in a correct manner. Hence, each robot is guaranteed of collision-free navigation to its destination from almost all initial free placements.				
CSE-TR-487-04	<u>Practical Slicing and Non-slicing Block-Packing without Simulated Annealing</u>	Chan and Markov	Feb, 04	31

We propose a new block-packer BloBB based on multi-level branch-and-bound. It is competitive with annealers in terms of runtime and solution quality. We empirically quantify the gap between optimal slicing and non-slicing floorplans by comparing optimal packings and best seen results. Most ongoing work deals with non-slicing packings, and implicitly assumes that best slicing packings are highly sub-optimal. Contrary to common belief, we show that the gap in optimal slicing and non-slicing packings is very small. Optimal slicing and non-slicing packings for apte, xerox and hp are reported. We extend BloBB to the block-packer CompaSS, that handles soft blocks. Optimal slicing packings for soft versions of apte, xerox and hp are reported. We discover that the soft versions of all MCNC benchmarks, except for apte, and all GSRC benchmarks can be packed with zero dead-space. Moreover, the aspect ratio bound $[0.5, 2]$ turns out to be not very restrictive, when area is concerned. Our heuristic slicing block-packer is able to pack with zero dead-space in most cases when we restrict the aspect ratio bound to $[0.57, 1.70]$. CompaSS improves the best published results for the ami49 X benchmarks suite, outperforming the leading multilevel annealer in runtime, solution quality and scalability. Additionally, realistic floorplans often have blocks with similar dimensions, if design blocks, such as memories, are reused. We show that this greatly reduces the complexity of block-packing.

CSE-TR-488-04 [A Compressed Memory Hierarchy Using an Indirect Index Cache](#)

Hallnor and
Reinhardt

Mar, 04 21

The large and growing impact of memory hierarchies on overall system performance compels designers to investigate innovative techniques to improve memory-system efficiency. We propose and analyze a memory hierarchy that increases both the effective capacity of memory structures and the effective bandwidth of interconnects by storing and transmitting data in a compressed form. Caches play a key role in hiding memory latencies. However, cache sizes are constrained by die area and cost. A caches effective size can be increased by storing compressed data, if the storage unused by a compressed block can be allocated to other blocks. We use a modified Indirect Index Cache to allocate variable amounts of storage to different blocks, depending on their compressibility. By coupling our compressed cache design with a similarly compressed main memory, we can easily transfer data between these structures in a compressed state, increasing the effective memory bus bandwidth. This optimization further improves performance when bus bandwidth is critical. Our simulation results, using the SPEC CPU2000 benchmarks, show that our design increases performance by up to 107% on some benchmarks while degrading performance by no more than 2% on others. Compressed bus transfers alone account for up to 59% of this improvement, with the remainder coming from increased effective cache capacity. As memory latencies increase, our design becomes even more beneficial.

CSE-TR-490-04 [Data Propagation in a Distributed File System](#)

Minkyong
Kim, Brian
Noble, Xichu
Chen, Dawn
Tilbury

July, 04 11

Distributed file systems benefit greatly from optimistic replication that allows clients to update any replica at any site. However, such systems face a new dilemma: Once data is updated at a replica site, when should it be shipped to others? In conventional file system workloads, most data is read by its writer, so it needs to be shipped only for administrative reasons. Unfortunately, shipping on demand substantially penalizes those who do share, while shipping aggressively places undue load on the system, limiting scalability. This paper explores a mechanism to predict and support sharing in a wide area file system. This mechanism uses the observation that updates that invalidate cached objects elsewhere are likely to be shared, and they should be propagated. The mechanism is evaluated on its precision, recall, and F-measure over traces capturing live file system uses. It reduces data shipped by orders of magnitude

compared to aggressive schemes, while reducing the performance penalty of on-demand shipment by nearly a factor of two.

CSE-TR-491-04	<u>The Internet Motion Sensor: A distributed global scoped Internet threat monitoring system</u>	Cooke, Bailey, Watson, Jahanian and Nazario	July, 2004	16
----------------------	---	---	------------	----

Networks are increasingly subjected to a broad spectrum of threats that impact the reliability and availability of critical infrastructure. In response, researchers and network operators have increasingly relied on monitoring to characterize and track these threats. This paper introduces the Internet Motion Sensor (IMS), a globally scoped Internet threat monitoring system whose goal is to measure, characterize, and track threats. The dark address sensors in the IMS extend simple passive capture using a novel transport layer service emulation technique to elicit payloads across all services, thereby addressing the issue of depth of service coverage. To achieve breadth of coverage, the IMS employs a distributed infrastructure and utilizes sensors that are aware of their address diversity and their position in the actively routed topology. Finally, the IMS uses an innovative signature encoding and data warehousing system combined with a hierarchical architecture to realize a system that is not only time and space efficient, but is also scalable to a global deployment. We explore the various architectural tradeoffs in the context of a 3 year deployment across multiple dark address blocks ranging in size from $1/24$ s to $1/8$. We show how the current architecture emulates services across a diverse set of routed and address topologies in a scalable manner. Results from three recent events are presented to illustrate the utility of such a system: the SCO Denial of Service attacks (December, 2003), the Blaster worm (August, 2003), and the Bagle backdoor scanning efforts (March, 2004).

CSE-TR-492-04	<u>Pueblo: A Modern Pseudo-Boolean SAT Solver</u>	Hossein M. Sheini and Karem A. Sakallah	July, 2004	11
----------------------	--	---	------------	----

In this report we introduce a new SAT solver that integrates logic-based reasoning and integer programming methods to systems of CNF and PB constraints. Its novel features include an efficient PB literal watching strategy that takes advantage of the preponderance of unit-coefficient literals in most PB constraints. Additionally, the solver incorporates several PB learning methods that take advantage of the pruning power of PB constraints while minimizing their overhead. Empirical evidence suggests that such judicious injection of IP techniques can be quite effective in practice.

CSE-TR-493-04	<u>CIDS: Causality Based Intrusion Detection System</u>	Samuel T. King, Z. Morley Mao and Peter M. Chen	August, 2004	6
----------------------	--	---	--------------	---

This paper presents a new style of intrusion detection system, CIDS, that links network and host based intrusion detection systems together using causal dependencies. This combination provides a number of benefits. First, combining alerts reduces false positives for both network and host based approaches. Second, host based intrusion detection systems have significantly less data to process since they only act on processes and files that are causally linked to a suspicious network packet. Third, alarms are associated with a specific network service, allowing system administrators to act accordingly after detecting a compromise. To show the

utility of this new system, we describe how several existing intrusion detection systems benefit from using our platform, and we introduce a new host based intrusion detection system that leverages the full power of causal dependency tracking.

CSE-TR-494-04	<u>What and How Much to Gain by Spectral Agility?</u>	Chun-Ting Chou, Hyoil Kim and Kang G. Shin	August, 2004	26
----------------------	---	--	--------------	----

CSE-TR-495-04	<u>Debugging operating systems with time-traveling virtual machines</u>	Samuel T. King, George W. Dunlap, and Peter M. Chen	August, 2004	14
----------------------	---	---	--------------	----

Operating systems are among the most difficult of software systems to debug with traditional cyclic debugging. They are non-deterministic; they run for long periods of time; their state and code is large and complex; and their state is easily perturbed by the act of debugging. This paper describes a time-traveling virtual machine that overcomes many of the difficulties associated with debugging operating systems. By time travel, we mean the ability to navigate backward and forward arbitrarily through the execution history of a particular run and to replay arbitrary segments of the past execution. We integrate time travel into a general-purpose debugger to enable a programmer to debug an OS in reverse, implementing commands such as reverse breakpoint, reverse watchpoint, and reverse single step. The space and time overheads needed to support time travel are reasonable for debugging, and movements in time are fast enough to support interactive debugging. We demonstrate the value of our time-traveling virtual machine by using it to understand and fix several OS bugs that are difficult to find with standard debugging tools.

CSE-TR-496-04	<u>Enriching intrusion alerts through multi-host causality</u>	Samuel T. King, Morley Mao, Dominic G. Lucchetti, and Peter M. Chen	August, 2004	13
----------------------	--	---	--------------	----

Current intrusion detection systems point out suspicious states or events but do not show how the suspicious state or events relate to other states or events in the system. We show how to enrich an IDS alert with information about how those alerts causally lead to or result from other events in the system. By enriching IDS alerts with this type of causal information, we can leverage existing IDS alerts to learn more about the suspected attack. Backward causal graphs can be used to find which host allowed a multi-hop attack (such as a worm) to enter a local network; forward causal graphs can be used to find the other hosts that were affected by the multi-hop attack. We demonstrate this use of causality on a local network by tracking the Slapper worm, a manual attack that spreads via several attack vectors, and an e-mail virus. Causality can also be used to correlate distinct network and host IDS alerts. We demonstrate this use of causality by correlating Snort and host IDS alerts to reduce false positives on a testbed system connected to the Internet.

[[Home](#) | [Technical Reports Page](#) | [Search](#)]

EXHIBIT D

Enriching intrusion alerts through multi-host causality

Samuel T. King, Z. Morley Mao, Dominic G. Lucchetti, Peter M. Chen

University of Michigan

Abstract

Current intrusion detection systems point out suspicious states or events but do not show how the suspicious state or events relate to other states or events in the system. We show how to enrich an IDS alert with information about how those alerts causally lead to or result from other events in the system. By enriching IDS alerts with this type of causal information, we can leverage existing IDS alerts to learn more about the suspected attack. Backward causal graphs can be used to find which host allowed a multi-hop attack (such as a worm) to enter a local network; forward causal graphs can be used to find the other hosts that were affected by the multi-hop attack. We demonstrate this use of causality on a local network by tracking the Slapper worm, a manual attack that spreads via several attack vectors, and an e-mail virus. Causality can also be used to correlate distinct network and host IDS alerts. We demonstrate this use of causality by correlating Snort and host IDS alerts to reduce false positives on a testbed system connected to the Internet.

1 Introduction

Intrusion detection systems (IDSes) are used to alert system administrators to possible attacks. IDSes monitor events at various levels in a system and attempt to alert administrators to events that match known attacks (signature-based IDS) or do not match normal behavior (anomaly-based IDS). Host-based IDSes monitor events on a single host, such as system calls or file accesses. Network-based IDSes monitor packets that are sent or received.

One factor that limits the accuracy and usefulness of IDS alerts is the lack of accompanying context. IDS alerts point out suspicious events without relating them to other events and state in the system. For example, Snort [1] highlights suspicious messages but does not describe how the receiving system reacted to these messages or what caused the sending system to send the suspicious message. Tripwire [6] alerts administrators to modified system files but does not show what caused the modifications or how the modified system files were used afterward. Additional context may help an administrator see the connections between different alerts or find other hosts that have been compromised.

This paper describes how to enrich an IDS alert with causal information about the state or events detected by the IDS. Informally, causal information shows which events led to the state/event detected by the IDS and which events were affected by the state/event detected by the IDS. Formally, this causal information is defined by Lamport's *happens-before* relationship [8], which results from messages between hosts and interprocess communication within a host. For example, a causal dependency is created from process A to process B if process A forks process B, if process A sends a message to process B, or if process A writes a file that is later read by process B. By iteratively analyzing these causal dependencies, we compute the transitive closure of