Tracing Based Active Intrusion Response[*]

Xinyuan Wang[†], Douglas S. Reeves[†‡], S. Felix Wu[††]

[†]Departments of Computer Science
[‡]Department of Electrical and Computer Engineering
North Carolina State University
xwang5@eos.ncsu.edu, reeves@eos.ncsu.edu

[††]Department of Computer Science
University of California at Davis
wu@cs.ucdavis.edu

Abstract

Network-based intrusion has become a serious threat to today's highly networked information systems, existing intrusion defense approaches such as intrusion prevention, detection, tolerance and response are "passive" in response to network-based intrusions in that their countermeasures are limited to being local to the intrusion target and there is no automated, network-wide counteraction against detected intrusions. While they all play an important role in counteracting network-based intrusion, they do not, however, effectively address the root cause of the problem – intruders.

What missing from existing intrusion prevention, detection, tolerance and response is an effective way to identify network-based intruders and hold them accountable for their intrusions. Network-based intrusion can not be effectively repelled or eliminated until its source is known.

In this paper, we propose Tracing Based Active Intrusion Response (TBAIR) as a new way to address the problem of network-based intrusion. Based on Sleepy Watermark Tracing (SWT), TBAIR is able to effectively trace the detected intrusion that utilizes stepping stone to disguise its origin at real-time, and dynamically push the intrusion countermeasures such as remote monitoring, blocking, containment and isolation close to the source of the intrusion. Through SWT's unique active watermark technique, TBAIR is able to trace even when the intrusion connection is idle. Therefore it helps to apprehend the intruders on the spot and hold them accountable for their intrusions.

Keywords: Network-based Intrusion, Active Intrusion Tracing, Active Intrusion Response.

1. Introduction

Existing approaches in defending network-based intrusions can be categorized as *intrusion prevention*, *intrusion detection*, *intrusion tolerance* and *intrusion response*. In particular, intrusion prevention utilizes authentication, encryption and firewall etc. to protect system from being attacked and compromised. However, experiences show that not all intrusions can be successfully prevented. For example, firewall assumes a static perimeter defense and partitions the network into inside and outside networks. It assumes that insiders of the firewall are trustworthy and outsiders of the firewall are not trustworthy (unless having been authenticated). While it is very useful to prevent networked systems from being attacked from outside network, it has virtually no defense against insider attacks. There are evidences showing that most successful system break-ins involve insiders [10,19].

Intrusion Detection Systems (IDS) attempts to detect intrusion through analysing observed system or network activities. Based on the type of observed activities, IDS can be classified as network-based or host-based. IDS will raise alarms when it has detected misuse or anomaly. It may also report intrusions by

[*] This work has been supported by the Defense Advanced Research Projects Agency, administered by AFOSR under contract F30602-99-1-0540. The views expressed are not necessarily those of AFOSR, DARPA, or the US Government.

emailing or paging system administrator and even disconnect intrusion connection locally. However, IDS is passive and it gives little information about where the intrusion really comes from.

Intrusion tolerance is an emerging approach to build survivable systems in recognizing that no system will be absolutely exempt from intrusions. Instead of focusing on intrusion prevention, it aims to design systems with the capacity to fulfil its primary missions in the presence of intrusion or partial compromising. Given the complexity of today's distributed, heterogeneous system design, there are always potential vulnerabilities that intrusions may explore to cause intolerable damages.

While intrusion prevention, detection and tolerance all play an important role in solving the problem of today's network-based intrusion, they are all passive and not adequate to solve the intrusion problem. One fundamental problem with existing intrusion prevention, detection and tolerance is that they do not effectively eliminate or deter network-based intrusions. The best they can do is to avoid being victims of network-based intrusions temporarily. Because they do not address the root cause of intrusions – intruders, those intruders can always explore new system vulnerabilities, find new accomplices – potentially insiders, and launch new attacks from different places. Because intrusion prevention, detection and tolerance do not effectively address the problem of compromised system recovery, intruders can use those compromised system as new base for further intrusion. What we need is an effective way to hold network-based intruders accountable for their intrusions, and an automated way to recover compromised hosts from doing further harm.

So far most computer security research regarding network-based intrusions has focused on prevention and detection. Intrusion response has been an afterthought and is generally limited to logging, notification and disconnection at local host. Any further response usually involves manual interactions such as off-line analysis, reporting incidents to CERT and installing fixes. Given today's high-speed network, many network-based intrusions can be very fast and short across wide areas of networks. Current ad hoc and manual intrusion response process neither provides the needed real-time intrusion response nor scales with network. Furthermore, because current automated intrusion responses are passive and lack network-wide response, they do not, however, eliminate or even effectively deter network-based intrusions.

In this paper, we propose *Tracing-Based Active Intrusion Response* (TBAIR) to address the problem of network-based intrusions. It differs from existing intrusion defense approaches in that it addresses the intrusion problem by targeting the root cause of the problem: intruders. It collaborates with IDS and will trigger automatic and network-wide tracing when intrusions are detected. Based on active tracing, TBAIR can effectively push the intrusion defense near the source of network-based intrusions. By blocking, containing and isolating network-based intrusions near their sources, TBAIR not only is more effective in protecting network infrastructure but also helps to recover more previously compromised hosts. By actively tracing network-based intrusions at real-time, TBAIR also helps to apprehend the intruders on the spot and hold them accountable for the intrusions. Therefore it is likely a more effective deterrent against further intrusion attempts.

2. Active Intrusion Response and Tracing

Ideally, an effective intrusion response (IR) to network-based intrusion should be able to: 1) disable detected intrusions at real-time; 2) counter detected intruder's ability to attack other targets; 3) help to apprehend network-based intruders at real-time; 4) help to recover as much as possible compromised network nodes; 5) repel future intrusions that are similar to the detected ones. In order to be effective in today's high-speed global networks, network-based intrusion response has to be network-wide and automatic.

One major problem in building an effective response to network-based attacks is the lack of source identification. Without effective source tracing, the attacked victim is blind at defending network-based attacks, and no effective intrusion countermeasures such as blocking and containing can be implemented. Network-based attacks can not be effectively repelled or eliminated until its source is known.

On the other hand, effective intrusion source tracing enables network-based intrusion response effectively pushes the intrusion countermeasures near the sources of network-based attacks. With real-time intrusion source tracing, automatic and network-wide responses such as blocking, containment and isolation can be built near the source of network-based intrusions at real-time. The closer these countermeasures are to the intrusion source, the more effective they will be. However, active intrusion response should not be over-reactive in order to minimize the possibility of being tricked into new forms of smurf attacks. Nevertheless, real-time intrusion source tracing is an appropriate intrusion response and it forms the foundation for further intrusion responses.

A complete solution to the problem of tracing network-based attacks is complicated by different anonymity gaining techniques used by different network-based attacks. For example, distributed denial-of-service (DDoS) attacks are usually generated from multiple previously compromised slave machines, under control of a remote master machine. The unidirectional flooding traffic from slave machines usually comes with a "spoofed" source IP address, which makes it difficult to trace even the slave machines. For bi-directional, interactive intrusions, one of the most widely used techniques to conceal their true origin is to connect through "stepping stones" [38]: intruders connect through a series of intermediate hosts before attacking the final target. All these techniques are easy to implement and use, making source tracing of network-based attacks among the hardest network security problems.

In this paper, we focus on the real-time tracing and response of interactive, bi-directional network-based intrusions that utilize connection chains to disguise their sources. A real-time solution to this problem not only enables us to build more effective intrusion responses to better repel or eliminate network-based intrusions, but also helps to deter DDoS by better protecting hosts from being compromised into slave machines.

3. Tracing Problem and Approaches

Given a series of computer hosts H_1, H_2, … H_n (n>2), when a person (or a program) sequentially connects from H_i into H_{i+1} (i=1,2,..n-1), we refer to the sequence of connections on <H_1, H_2, … H_n> as a *connection chain*, or *chained connection*. The *tracing problem* of a connection chain is, given H_n of a connection chain, to identify H_{n-1}, … H_1.

Tracing the source of intrusion through a connection chain over the Internet is a difficult problem. It requires network-wide collaboration among hosts in the network, and yet some of the hosts may be compromised and not trustworthy, As a network security mechanism, intrusion source tracing should be based on trust of appropriate network resources, and be robust against compromised hosts in the network. To trace back the chained connections through multiple hosts, effective correlation is needed at intermediate nodes. Because network-based intrusion in today's high-speed network can be very short, correlation at intermediate nodes needs to be fast and accurate. Additionally, to scale the tracing system to the Internet, the tracing system should have minimum overhead while providing a fast response to detected network-based intrusion.

In general, tracing approaches for a connection chain can be divided into two categories: host-based and network-based, each of which can further be classified into either active or passive. Table 1 provides a classification of existing tracing approaches, as well as our proposed tracing mechanism.

	Passive	**Active**
Host-based	DIDS CIS	Caller ID
Network-based	Thumbprinting Timing-Based Deviation-Based	IDIP SWT

Table 1: Classification of Existing Tracing Approaches and SWT

Distributed Intrusion Detection System (DIDS) [29] developed at UC Davis is a host-based tracing mechanism that attempts to keep track of all the users in the network and account for all activities to network-wide intrusion detection systems. Each monitored host in theDIDS domain collects audit trails and sends audit abstracts to a centralized DIDS director for analysis. While DIDS is capable of keeping track of all users moving around the network through normal login within the DIDS domain, it seems not feasible in large-scale network such as the Internet, because of its centralized monitoring of network activities.

The Caller Identification System (CIS) [17] is another host-based tracing mechanism. It eliminates centralized control by utilizing a truly distributed model. Each host along the login chain keeps a record about its view of the login chain so far. When the user from the *n-1*th host attempts to login into the *n*th host, the *n*th host asks the *n-1*th host about its view of the login chain of that user, which should be 1,2 ... n-1 ideally. The *n*th host then queries host n-1 to 1 about their views of the login chain and so on. Only when the login chain information from all queried hosts matches, will the login be granted at the *n*th host. While CIS attempts to maintain the integrity of login chain by reviewing information from hosts along the login chain, it introduces excessive overheard to the normal login process.

Caller ID, described by Stuart Staniford-Chen [7] , is yet another interesting host-based approach that is said to be used by the Air Force. Caller ID is controversial in that it actually utilizes the same break-in technique used by intruders to break into the hosts along the connection chain reversibly. If the intruder from H_0 connects through H_1, H2…H_{n-1} to the final target H_n, the network security personnel at H_n first breaks into $H_{n-1;}$ from there they can find out the intruder comes from H_{n-2}, then they break into H_{n-2} and so on. Eventually they can find the origin of the intruder. One compelling advantage of Caller ID is that it is scalable to the Internet. It is also efficient in the sense that it introduces less overhead compared to DIDS and CIS. But its manual approach makes it difficult to trace short intrusion in today's high-speed network. Besides it legal complications, Caller ID also has the drawback that one must perform manual tracing on the host where the intruder is active, which is easily-noticed by the intruder.

The fundamental problem with the host-based tracing approach is its trust model. Host-based tracing places its trust upon the monitored hosts themselves. In specific, it depends on the correlation of connections at every host in the connection chain. If one host is compromised and is providing misleading correlation information, the whole tracing system is fooled. Because host-based tracing requires participation and trust of every host involved in the network-based intrusion, it is very difficult to be applied in the context of the public Internet.

Network-based tracing is the other category of tracing approaches. Neither does it require the participation of monitored hosts, nor does it place its trust on the monitored hosts. It is based on the property of network connections: the application level content of chained connections is invariant across the connection chain. In particular, the thumbprint [7] is a pioneering correlation technique that utilizes a small quantity of information to summarize connections. Ideally it can uniquely distinguish a connection from unrelated connections and correlate those related connections in the same connection chain. While thumbprinting can be useful even when only part of the Internet implements it, it depends on clock synchronization to match thumbprints of corresponding intervals of connections. It also is vulnerable to retransmission variation. This severely limits its usefulness in real-time tracing.

The timing-based scheme [38] by Zhang and Paxson is a novel network-based correlation scheme for detecting stepping-stones across the connection chain. The correlation is based on the distinctive timing characteristics of interactive traffic, rather than connection contents. It pioneered new ways of correlating encrypted connections. It requires no clock synchronization and it is robust against retransmission variation. However, because its timing characteristics are defined over the entire duration of each connection to be correlated, it is difficult to be used in real-time correlation.

The deviation-based approach [37] by Yoda and Etoh is another network-based correlation scheme. It defines the minimum average delay gap between the packet streams of two TCP connections as *deviation*. The deviation considers both timing characteristics and the TCP sequence number, and it does not depend on the TCP payload. Similar to the timing-based approach, the deviation-based approach does not require clock synchronization and is robust against retransmission variations. However it is difficult to be used in real-time correlation as the deviation is defined over all the packets of a connection. Another drawback of deviation-based approach is that it correlates only TCP connections.

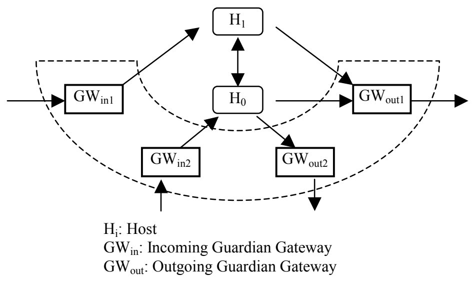
One fundamental problem with passive network-based approaches is its computational complexity. Because it passively monitors and compares network traffic, it needs to record all the concurrent incoming and outgoing connections even when there is no intrusion to trace. To correlate at any host in the connection chain, it needs to match every concurrent incoming connection with every concurrent outgoing connection at that host. That is, for a host with m concurrent incoming connections and n concurrent outgoing connections, the passive network-based correlation approach would take $O(m \times n)$ comparisons, in addition to the $O(m+n)$ scanning and recording of concurrent connections.

On the other hand, the active network-based approach dynamically controls how connections are correlated through customized packet processing. It does not need to record all the concurrent incoming and outgoing connections at any host in the connection chain. It does not need to match each concurrent incoming connection with each concurrent outgoing connection. For a host with m concurrent incoming connections and n concurrent outgoing connections, the active network-based approach is able to correlate within *time dependent only on the number of connections being actively traced,* in addition to the $O(m+n)$ scanning of concurrent connections.

IDIP (Intrusion Identification and Isolation Protocol) [27] is a proposal by Boeing's Dynamic Cooperating Boundary Controllers Program that uses an active approach to trace the incoming path and source of intrusion. In the proposal, boundary controllers collaboratively locate and block the intruder by exchanging intrusion detection information, namely, attack descriptions. While it does not require any boundary controller to record any connections for correlation, its intrusion tracing is closely coupled with intrusion detection. The effectiveness of IDIP depends on the effectiveness of intrusion identification through the attack description at each boundary controller. Therefore IDIP requires each boundary controller to have the same intrusion detection capability as the IDS at the intrusion target host. It is questionable whether the intermediate boundary controller is able to identify an intrusion based on a hard-coded attack description.

4. Sleepy Watermark Tracing

SWT [33] is an active network-based tracing framework. It is "sleepy" in that it does not introduce overhead when no intrusion is detected. Yet it is "active" in that when an intrusion is detected, the target will inject a watermark into the backward connection of the intrusion and "wakes up" intermediate routers along the intrusion path. SWT exploits the observations: 1) interactive intrusions with chained



H1
GWin1
H0
GWout1
GWin2
GWout2
Hi: Host
GWin: Incoming Guardian Gateway
GWout: Outgoing Guardian Gateway

Figure 1: Guardian Gateway Set

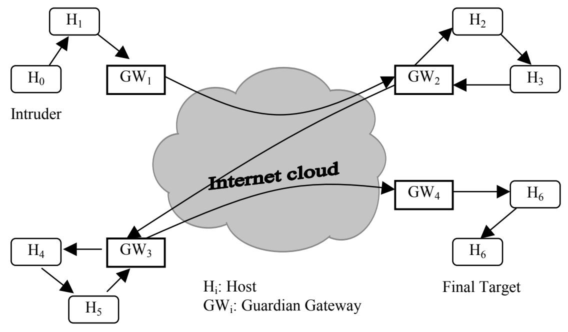


Figure 2: Intrusion Chain Tracing Model

connections are bi-directional and symmetric at the granularity of connections; 2) application level contents are invariant across connection chains.

By watermarking selected packets and processing them accordingly, SWT provides many potential advantages over existing intrusion tracing approaches. 1) SWT separate intrusion tracing from intrusion detection and it does not require any node other than the intrusion target to have the intrusion detection capability. 2) Unlike thumbprinting, timing-based and deviation-based approaches, SWT does not need to record all the concurrent incoming and outgoing connections at any node, and it does not require matching each of the incoming connections with each of the outgoing connections for correlation at any node. 3) SWT requires no clock synchronization and is robust against retransmission variation. 4) SWT traces only when needed. 5) So far the most compelling advantage of SWT is its correlation accuracy and efficiency. By using watermarks, SWT can trace the intrusion connection chain to its origin within a single keystroke of the intruder. With its unique active tracing, SWT can trace the intrusion connection chain back to its origin even when the intruder is inactive. 6) We have found that SWT can be implemented efficiently. It does not introduce any noticeable overhead to routers, and it only requires a few network server applications at the intrusion target host to be modified to inject watermarks.

In the remainder of this section, we describe the SWT model concepts and assumptions.

4.1 Basic SWT Concepts

In order to keep track of network-based intrusions to hosts, it is desirable to monitor hosts through the nearest router or gateways. This is termed a *Guardian Gateway*. We define the *Incoming Guardian Gateway* of host *H* as the nearest router that forwards incoming traffic to H and the *Outgoing Guardian Gateway* of host *H* as the nearest router that forwards outgoing traffic from *H*. It is possible that one host has more than one incoming or outgoing guardian gateway. We define the union of incoming and outgoing guardian gateways of a host as its *Guardian Gateway Set* (e.g., $\{GW_{in1}, GW_{in2}, GW_{out1}, GW_{out2}\}$ in Figure 1). For any guardian gateway set *G*, we define those hosts as *Guarded Host* of *G* whose guardian gateway closure is a subset of *G*. For a host *H*, while the traffic between *H* and its directly connected neighbour hosts does not pass through any gateways, the traffic between *H* and any non-directly-connected hosts must pass through its guardian gateway closure.

We further define a *leap* as one connection step between hosts within a connection chain (e.g., $<H_i$, $H_{i+1}>$ in Figure 2). One leap may consist of multiple hops (or links in the physical network) and the two guardian gateways of the two end hosts. A leap can be specified by a 5-tuple consisting of

<protocol number, source ip address, source port number, destination ip address, destination port number>

Now the tracing problem of chained intrusion is defined as discovering and sequencing the guardian gateways of those hosts in the intrusion path, or (equivalently) as finding the leaps along the intrusion path.

4.2 Basic SWT Assumptions

We have identified the following assumptions that motivate and constrain our design:

- Intrusions are interactive and bidirectional,
- Routers are trust worthy and hosts are not trust worthy,
- Each host has a single SWT guardian gateway and
- There is no link-to-link encryption.

The first two assumptions represent our assessments of the nature of the intrusions. Here we refer to intrusions as those attacks aiming to gain unauthorized access, rather than denial of service attacks. A study of CERT security incidents [14] indicates that almost all security incidents, especially unauthorized access incidents, happened at computer hosts rather than routers or gateways. Therefore we believe our assumption to trust routers will cover most intrusion cases. In case there are indeed compromised routers involved in intrusion, the compromised router will be effectively indistinguishable from an attacker. The compromised router needs to be addressed first, before the tracing of the intrusion can go any further. In this case SWT can still trace to the farthest trustworthy guardian gateway.

The assumption of each host having a single SWT guardian gateway is only for simplifying the presentation of the SWT architecture. In case some host has multiple SWT guardian gateways, the guardian gateway set will be used in SWT tracing.

The final assumption represents the inherent limitation of any tracing based on network content. We believe that correlation of encrypted connections in real-time is still an open problem.

5. Active Intrusion Responses Based on SWT

Without effective intrusion source tracing, intrusion response is limited to the nearby of intrusion target and is passive in front of network-based intrusions. On the other hand, effective intrusion source tracing enables us to build a more active and dynamic intrusion response by pushing the intrusion countermeasures near the source of network-based intrusions. In particular, active network [4, 30,34] is an emerging framework that seeks to increase the programmability of computer networks and network components. It enables user and application to dynamically control how packets are handled. This customized packet processing opens new ways of network-based intrusion response that were not available in traditional passive networks.

With the guidance of real-time intrusion source tracing of SWT, a number of novel, automatic and network-wide intrusion responses can be built on SWT guardian gateway by applying active network principles:

- **Remote monitoring and surveillance**. After successful intrusion source tracing, the SWT guardian gateway close to the intruder can be devised to monitor the intruder and report his/her activities back to the intrusion target. By monitoring and surveillance near the intruder, it not only gives us closer view on the intruder itself, but also helps us to find intruder's accomplices. It also helps to reveal previously compromised hosts.

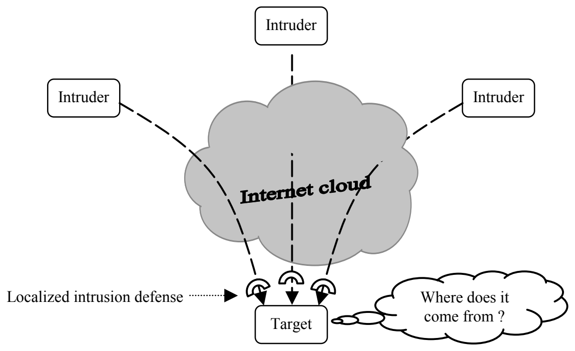


Figure 3: Passive, Localized Intrusion Response

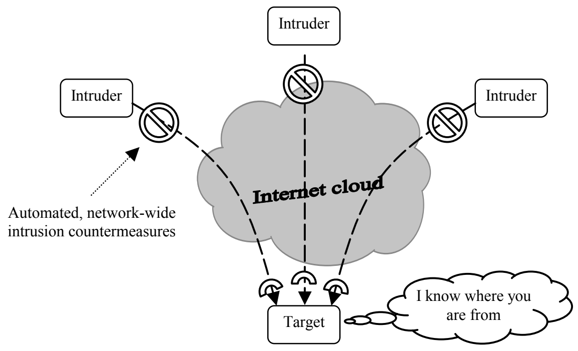

Intruder
Intruder
Intruder
Internet cloud
Automated, network-wide intrusion countermeasures
Target
I know where you are from

Figure 4: Active, Network-wide Intrusion Response

- **Remote decoy and trap**. Sometimes it is desirable to give the intruder the impression that he/she is still inside the intrusion target while he/she has actually been diverted to a dynamically created decoy. Such application specific decoy can be dynamically created at the SWT guardian gateway and be moved along the backward of the intrusion path toward the intruder. In the same time, the actual intrusion connection to the intrusion target can be disconnected. To facilitate monitoring and surveillance, the decoy can deliberately introduce response delays to keep the intruder on hook.
- **Remote blocking and containment**. Blocking can be easily implemented at various SWT gateways along the intrusion path upon successful intrusion tracing. Remote blocking is different from existing local blocking in that the blocking happens at a series of points toward the intruder. This blocking can also be combined with remote decoy to trick the intruder. After enough network access points from the intruder have been blocked or decoyed, the intruder will be contained.
- **Remote isolation and quarantine**. During the process of intrusion source tracing and response, it is desirable to isolate those previously compromised hosts that are being used as stepping stones by the intruder. SWT can be extended to automatically "bypass" those identified stepping-stones of the intrusion connections being traced. This automatic bypassing of stepping-stones effectively cut the intruder's control over those previously compromised hosts and thus it helps to recover them.
- **Dynamic perimeter defense**. By combining above active intrusion responses, we can build dynamic perimeter defense as opposed to current static perimeter defense. To improve efficiency, we can also introduce "sleepiness" into the active defense. That is the active intrusion response will be active only for a configurable period of time unless being refreshed.

While these examples of active intrusion response are based on SWT, they are not necessarily limited to SWT. Same active intrusion responses can be built upon other real-time intrusion source tracing.

6. Open Problems and Future Research

While active intrusion response paradigm brings many potential advantages over existing passive intrusion response, it also introduces a number of open issues. Many of these issues are applicable to the general active intrusion response approaches.

One fundamental problem regarding active intrusion response is that we do not completely understand what kinds of active intrusion responses are appropriate. While different responses are needed for different intrusions, the selection of appropriate active response for a particular intrusion involves more than just policy. One concern over active intrusion response is the possibility of active intrusion response being tricked into new form of network-based attacks. How to balance the overall effectiveness and robustness of active intrusion response is still an open problem.

A second problem involves the trust issue of various network nodes. As with any other security mechanism, network-wide active intrusion response has to be built upon trust on appropriate components in the network. SWT assumes that all routers are trust worthy and remote hosts are not. This trust model is appropriate for tracing but may not fit for active intrusion response as a compromised router could do much more harm during active intrusion response than just tracing. An appropriate trust model for building effective active intrusion response over large-scale networks such as the Internet is still unknown.

A third problem is how to make sure active intrusion response is not over-sensitive or over-reactive while maximizing its real-time response capability. If the active response is over-sensitive or over-reactive, it could be turned into new form of denial of service attacks against supposedly beneficial systems.

There are also some particular aspects of TBAIR may be deemed controversial. For example, active tracing and remote monitoring and surveillance introduce privacy concerns. Like wiretapping, these active intrusion response capabilities are very useful when used by right people for right reason, but they could infringe people's privacy when they are at wrong hands.

Unlike most other tracing mechanisms, SWT is intrusive in the sense that the watermark-enabled application actively injects a watermark into the backward traffic of the intrusion connection. For interactive applications such as telnet, rlogin, watermark can be made invisible to normal end users by careful selection. For applications such as ftp, rcp, injecting watermark will break the integrity of data that the intruder receives. Because watermark injection only happens when there is an intrusion detected, only the intruder's network application will receive watermarked response. Therefore only the intruder's intrusive network application could potentially be broken by watermarks. We believe this is a reasonable price to pay for the highly accurate, real-time, single packet tracing capability. By controlling the number of watermarks injected by watermark-enabled application, we can further keep the intrusiveness to a minimum and make SWT harder to detect by intruders and their confederates. TBAIR could be intrusive in that it may block and even divert intrusion connections. Care must be taken in order to make sure that these intrusion countermeasures will not be over-aggressive and be turned into new form of intrusions and cause chaos.

7. Conclusion

In this paper, we have argued that existing intrusion defense approaches such as intrusion prevention, detection and tolerance are passive and not adequate to eliminate network-based intrusions. We have proposed a more active response paradigm to help to better repel or eliminate network-based intrusions and have identified the need of effective network-wide intrusion source tracing in order to build automated, network-wide response system. We have presented SWT as an example of such network-wide intrusion tracing capability and discussed possible active intrusion countermeasures that can be built upon SWT to dynamically push the intrusion defense perimeter close to the source of network-based intrusion. While there are still several open issues, we believe our work represents a valuable first step toward an automated, network-wide intrusion response framework.

References

[1] S. M. Bellovin. (2000) ICMP Traceback Messages. Internet Draft: draft-bellovin-itrace-00.txt.

[2] W. Bender, D. Gruhl, N. Morimoto and A. Lu. (1996) Technique for Data Hiding. *IBM Systems Journal*, Vol. 35, Nos. 3&4

[3] S. Bhattacharjee, K. L. Calvert and E. W. Zegura. (1997) An Architecture for Active Networking. High Performance Networking (HPN'97), White Plans, NY.

[4] K. L. Calvert, S. Bhattacharjee and E. Zegura. (1998) Directions in Active Networks. *IEEE Communication Magazine*.

[5] R. H. Campbell, Z. Liu, M. D. Mickunas, P. Naldurg and S. Yi. (2000) Seraphim: Dynamic Interoperable Security Architecture for Active Networks. In *Proceedings of IEEE OPENARCH'2000*.

[6] H. Y. Chang, R. Narayan, S.F. Wu, B.M. Vetter, X. Y. Wang et al. (1999) DecIdUouS: Decentralized Source Identification for Network-Based Intrusions, In *Proceedings of 6th IFIP/IEEE International Symposium on Integrated Network Management*.

[7] S. Staniford-Chen, L. T. Heberlein. (1995) Holding Intruders Accountable on the Internet. In Proceedings of *IEEE Symposium on Security and Privacy*.

[8] Computer Emergency Response Team. (2000) CERT Advisory CA-2000-01 Denial-of Service Development. http:// www.cert.org/advisories/CA-2000-01.html.

[9] Computer Emergency Response Team. (1999) Results of the Distributed-Systems Intruder Tools Workshop. http://www.cert.org/reports/dsit_workshop.pdf.

[10] Computer Security Institute. Annual CSI/FBI Computer Crime and Security Survey. (2001) http://www.gocsi.com/prelea_000321.htm.

[11] N.G. Duffield and M. Grossglauser. (2000) Trajectory Sampling for Direct Traffic Observation. *Proceedings of the ACM SIGCOMM '2000*.

[12] M. B. Greenwald, S. K. Singhal, J. R. Stone and D. R. Cheriton. (1996) Design an Academic Firewall: Policy, Practice and Experience with SURF. *Internet Society Symposium on Network and Distributed System Security (NDSS '96)*.

[13] L. T. Heberlein, K. Levitt and B. Mukherjee. (1992) Internetwork Security Monitor: An Intrusion-Detection System for Large-Scale Networks. In *Proceedings of 15th National Computer Security Conference*.

[14] J. D. Howard. (1997) An Analysis of Security Incidents on The Internet 1989 - 1995, PhD Thesis, http://www.cert.org/research/JHThesis/Start.html.

[15] J. Ioannidis and M Blaze. (1993) The Architecture and Implementation of Network-Layer Security under Unix. In *Proceedings of 4th USENIX Security Symposium*.

[16] W. Jansen, P. Mell, T. Karygiannis, D. Marks. (1999) Applying Mobile Agents to Intrusion Detection and Response. NIST Interim Report (IR) – 6416.

[17] H. Jung, et al. Caller Identification System in the Internet Environment. (1993) In *Proceedings of 4th USENIX Security Symposium*.

[18] S. Kent, R. Atkinson. (1998) *Security Architecture for the Internet Protocol*. IETF RFC 2401.

[19] B. R. Koerner. (1999) Special Report: Can Hackers Be Stopped ? *US News & World Report V.126 no.2*.

[20] L. H. Lehman, S. J. Garland, and D. Tennenhouse. (1998) Active Reliable Multicast. In *Proceedings of IEEE INFOCOM '98*.

[21] Mark Linehan. Comparison of Network-Level Security Protocols. (1994) IBM T.J. Watson Research Center.

[22] S. Mittra, T. Y. Woo. (1997) A Flow-Based Approach to Datagram Security. In *Proceedings of the ACM SIGCOMM '97.*

[23] B. Mukherjee, L.T. Heberlein, and K.N. Levitt. (1994) Network Intrusion Detection, *IEEE Network*, Vol. 8, No. 3.

[24] S. Murphy et al. (1998) Security Architecture for Active Nets. AN Security Working Group.

[25] Rolf Oppliger. (1997) Internet Security: Firewalls and Beyond. *Communications of the ACM,* Vol. 40, No. 5.

[26] S. Savage, D. Wetherall, A. Karlin and T. Anderson. (2000) Practical Network Support for IP Traceback. *Proceedings of the ACM SIGCOMM '2000*.

[27] D. Schnackenberg. (1998) Dynamic Cooperating Boundary Controllers. http://www.darpa.mil/ito/Summaries97/E295_0.html, Boeing Defense and Space Group.

[28] B. Schwartz, A. W. Jackson, W. T. Strayer, W. Zhou, R. R. Rockwell and C. Partridge. (1999) Smart Packets for Active Networks. In *Proceedings of IEEE OPENARCH '1999.*

[29] S. Snapp, et all. (1991) DIDS (Distributed Intrusion Detection System) – Motivation, Architecture and Early Prototype. In *Proceedings of 14th National Computer Security Conference*.

[30] D Tennenhouse and D Wetherall, (1996) Towards an Active Network Architecture. In *SPIE Proceedings of Conference on Multimedia Computing and Networking1996*, January.

[31] V. C. Van. (1997) A Defense Against Address Spoofing Using Active Networks. Master's thesis, Department of Electrical Engineering and Computer Science, MIT.

[32] X. Y. Wang. (2000) Survivability through Active Intrusion Response. In *Proceedings of 3rd IEEE Information Survivability Workshop (ISW-2000)*.

[33] X. Y. Wang, D. S. Reeves, S. F. Wu and J. Yuill. (2001) Sleepy Watermark Tracing: An Active Intrusion Response Framework. In the *Proceedings of 16th International Conference of Information Security (IFIP/SEC'01)*, Paris, France.

[34] D. Wetherall, J. Guttag and D. Tennenhouse. (1998) ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols. In *Proceedings of IEEE OPENARCH '1998.*

[35] G. White and V. Pooch. (1996) Cooperating Security Managers: Intrusion Detection Systems. *Computer & Security*, Vol. 16, No. 5.

[36] S. F. Wu. (1996) Sleepy Network-Layer Authentication Service for IPSEC. In *G. Martella E. Bertino, H. Kurth and E. Montolivo, editors, 4th European Symposium on Research in Computer Security – ESORICS 96 LNCS-1146,* Rome, Italy.

[37] K. Yoda and H. Etoh. (2000) Finding a Connection Chain for Tracing Intruders. In *F. Guppens, Y. Deswarte, D. Gollmann and M. Waidner, editors, 6th European Symposium on Research in Computer Security – ESORICS 2000 LNCS-1895,* Toulouse, France.

[38] Y. Zhang and V. Paxson. (2000) Detecting Stepping Stones. In *Proceedings of 9th USENIX Security Symposium*.