Trapping Malicious Insiders in the SPDR Web

J. Thomas Haigh*, Steven A. Harp*, Richard C. O'Brien*, Charles N. Payne*, Johnathan Gohde+, John Maraist~
* *Adventium Labs, Minneapolis, MN (email: firstname.lastname@adventiumlabs.org)*
+ *General Dynamics Advanced Information Systems, Bloomington, MN (email: Johnathan.Gohde@gd-ais.com)*
~ *Smart Information Flow Technologies, Minneapolis, MN (jmaraist@sift.info)*

Abstract

The insider threat has assumed increasing importance as our dependence on critical cyber information infrastructure has increased. In this paper we describe an approach for thwarting and attributing insider attacks. The Sense, Prepare, Detect, and React (SPDR) approach utilizes both a highly intelligent software reasoning system to anticipate, recognize, respond to, and attribute attacks as well as a widely distributed set of hardware-based sensor-effectors to provide alerts used by the reasoning system and to implement responses as directed by it. Using hardware sensor-effectors greatly reduces the risk that a savvy malicious insider can bypass or cripple the system's monitoring and control capabilities. In this paper we describe the prototype SPDR system and the results of its successful evaluation by an independent, DARPA-sponsored Red Team. We conclude with thoughts on possible SPDR enhancements and further research.

1. Introduction

Unlike an outsider, the malicious insider begins with a launch point within the organization's boundary, thus eliminating the effort and risk associated with defeating boundary protection mechanisms. Also, in many cases the insider can mount a successful attack using only his authorized access to the system. According to the *Webster Report*, most of Robert Hanssen's exploits involved the use of authorized access for unauthorized purposes [1]. In the one attack where Mr. Hanssen used unauthorized access, hacking into his supervisor's workstation, his job was made simpler because of his knowledge of the organization, its personnel and its network. He knew which workstation to target. He had a good idea of how it was configured, and he had personal knowledge of the authorized user. More recent reports [2] confirm that ready access to knowledge about the system and acquaintance with other authorized users is the third advantage an inside adversary has.

A compensating difference between an outsider and a malicious insider is that the insider's connection to the organization is generally known, through use of identification and authentication mechanisms. This makes it easier to associate insiders with their actions and hold them accountable. As a result, attribution is an important aspect of defense against malicious insiders. Even when the defenders cannot thwart an initial attack, *post facto* attribution provides the opportunity to prevent the malicious insider from conducting further attacks. The risk of attribution, even after the fact, can serve as a strong deterrent to insiders, who have more to lose, and who can be held more accountable, than typical outside adversaries.

To counter the malicious insider, defenders need tools that detect patterns of suspicious behavior and then implement responses that will prevent or delay the suspicious behavior sufficiently to ensure successful completion of critical network operations. Moreover, these responses must not themselves interfere with successful completion of critical operations.

Defenders can also use tools that support the attribution of an attack to a specific individual. If it is not possible to unequivocally attribute the attack to a particular individual, then it can be quite useful to reduce the space of possible perpetrators so that investigators know whom to target.

The goal of the DARPA-sponsored Sense, Prepare, Detect, and React (SPDR) project was to develop a system that enables individuals to accomplish their authorized tasks, but severely limits a malicious insider's potential to do harm, both by thwarting attacks as they progress and by supporting the attribution of attacks to particular individuals within the organization.

The SPDR system uses an approach for detecting, attributing, and thwarting attacks by malicious insiders which combines intelligent software for anticipating, recognizing, responding to, and attributing attacks with a powerful, hardware-based sensor-effector to monitor and control the traffic between hosts on the protected network. In May of 2008, it performed well during a rigorous three day evaluation during which SPDR defended a mock aircraft carrier-based air mission planning system against an independent Red Team. The SPDR prototype successfully thwarted 75% of the attacks and correctly attributed over 80% of the significant attacks. SPDR also generated relatively

978-0-7695-3450-3/09 $25.00 © 2009 IEEE

few false responses (10% of test caes) or attributions (5%).

SPDR appears to be different from previous published research on the insider threat. There have been prior studies of organizational factors or personal psychological factors related to the insider threat [3], [4]. Other researchers have used anomaly analysis and detection techniques to support detection of insider attacks. These have been either user/role based [5] or application based [6], coupled with refined access control policies. Some authors have combined several of these techniques [7]. SPDR's use of anomaly detection, enhanced by predictive reasoning to identify attacks, coupled with dynamic access controls implemented on a highly trustworthy hardware base, integrates and extends several of these approaches to the problem..

Before presenting an overview of the SPDR system and describing the results of the assessment, we describe the testbed used for the assessment. We conclude with a discussion of approaches for addressing deficiencies of the current SPDR prototype.

2. The SPDR Testbed

The goal of the SPDR prototype was to defend a simple version of an aircraft carrier Air Operations Center (AOC) that plans air sorties, implemented using a Services Oriented Architecture (SOA). The AOC is part of a larger network wholly within the carrier's internal network, and we assumed the insider was somewhere on the carrier. As is typically the case, there are different classes of insiders, depending on their degree of system access.

Figure 1 illustrates three types of insider. Some insiders have authorized access to the AOC, composed of the four workstations at the top of the figure. Others only have authorized access to non-mission workstations outside the AOC but are authorized to chat with users in the AOC. This provides a broader attack surface for the Red Team to exploit. They can launch malware from the non-AOC workstation or use it as a bounce point for exfiltration. Finally, some insiders only have physical access to the carrier network. They can connect their own rogue workstations to the network, but they cannot authenticate to any DRED-protected host. Local system administrators, who have access to all the hosts, were a fourth class of insider. They can install malware on any host and then log out before the malware activated.

The AOC hosts the Combat Direction Center application and the associated SOA services:

- Intelligence Operations Service
- Strike Operations Service
- Air Operations Service

We discuss the DREDs and the ISM in the next section. The firewall router represents the boundary between the carrier and the of the world, and the Scorebot is a test artifact used to start, stop, and assist in scoring the Red Team tests.

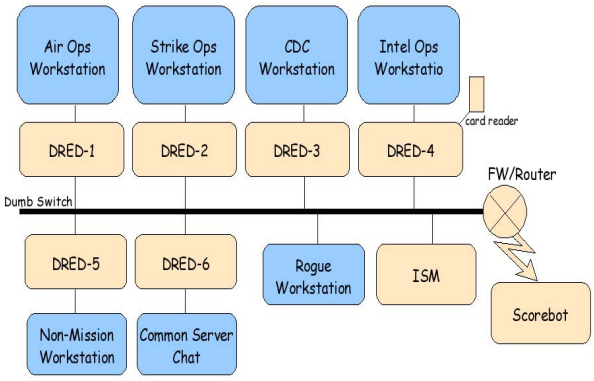

Figure 1. The SPDR testbed is a prototype SOA-based air operations planning center on an aircraft carrier

A typical mission planning loop might be the following. U.S. Intelligence identifies a surface threat in the area of a carrier group's engagement and enters it into a national Global Information Grid (GIG) threat database to which the carrier's Intelligence Operations Service is subscribed. The threat might be hostile destroyers with medium-range surface-to-air missiles. Information obtained from the GIG threat database indicates detection and engagement ranges for the threats, images of the class of vessel, and possibly other information.

The Intelligence Operations Service fuses this data with any information the carrier has obtained from monitoring surface tracks in its vicinity. If Intelligence Operations confirms it is a potential threat to the carrier group's presence, it sends the fused information to the Combat Direction Center.

Acting on the threat message, the Combat Direction Center plans the mission and requests a mission flight plan from the Strike Operations service. This request includes data such as the threat and its location, the goal of the mission, a mission timeline, and details about the threats to the mission.

The Strike Operations service queries Air Operations about available assets to support the

mission and in return receives a list of assets that have the proper capabilities and availability. Strike Operations then crafts a flight plan that satisfies the goals of the mission. For instance, the flight plan might involve sending F/A-18s to strike the destroyers. The flight plan may include other assets, such as Airborne Early Warning.

Strike Operations then reserves the assets and sends the flight plan back to the Combat Direction Center. If the flight plan is acceptable, the Combat Direction Center then forwards it to Intelligence Operations in order to brief the crew that will fly the mission. Once everything is ready, the Combat Direction Center gives the okay to start the mission.

3. SPDR Overview

As shown in Figure 2, the SPDR approach incorporates both off-line and on-line reasoning elements. Off-line, the SPDR Plan Generation Module uses automated planning techniques developed on the BAMS and SCOAP projects [8] to reason about the potential threats faced by the defended network based on models of potential attacker goals and capabilities. The result of this reasoning is an attack plan library, consisting of a complete (with respect to the network and adversary models and a pre-determined depth) set of possible attacker courses of action.

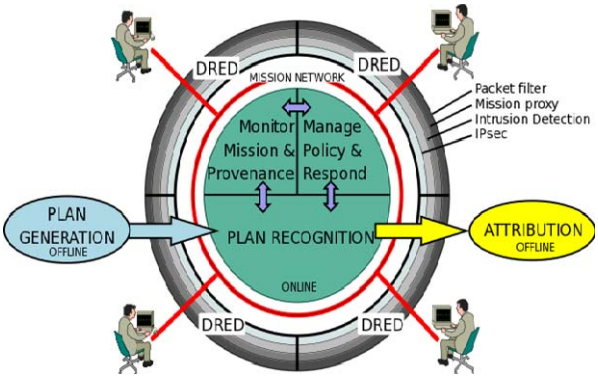

Figure 2. The SPDR system utilizes both off-line and online reasoning as well as a Detection and Response Embedded Device (DRED) on every protected host.

The online SPDR components include a set of centrally located intelligent components, known collectively as the Intelligent Security Manager (ISM), that detect and respond to ongoing attacks and that attribute those attacks to specific hosts and users, based on alerts from sensors distributed throughout the network.

SPDR also strengthens critical elements of the system to prevent a knowledgeable adversary with insider access from disabling or bypassing defenses. In particular, critical detection and response capabilities of SPDR are placed off limits to the insider within a cost-effective tamper-resistant Detection and Response Embedded Device (DRED). This paper describes the DRED and the ISM. For further information on the off-line SPDR components the reader is directed to [8].

3.1 The DRED

Any viable solution to the insider threat problem must address the concern that a user might circumvent the system by modifying, disabling, or bypassing the sensing and response mechanisms. SPDR addresses this concern by placing critical sensing and response mechanisms in the DRED, a hardware sensor-effector platform that protects a single host, yet is not under the control of the host processor or the user currently logged into the host.

The DRED only accepts configuration information over encrypted connections with a central policy manager and the SPDR Response Module on the ISM. Thus, the DRED is highly resistant to attacks. The associated host has no interface it can use to modify or disable the DRED. Any other network device would have to acquire the appropriate keying material to masquerade as the ISM.

Moreover, the DRED encrypts communications to the network, and only it possess the cryptographic keys required for network access to and from its host. Thus the DRED cannot be bypassed; all network access is consistent with the user-based network authorization policy enforced by the DRED, and the DRED can examine all traffic for malicious content and intent. However, SPDR would not prevent a malicious insider from connecting to some other network if it were physically possible to do so. For example, the DRED would not prevent connection to a nearby wireless network.

As its name implies, the intended form factor for the DRED is as an embedded device on the host computer. As powerful network processing hardware trends downwards in price, this is an increasingly attractive approach. However, for the SPDR prototype each DRED was implemented as an inline, bump-in-the-wire single board computer running a specially

configured SELinux operating system. There are other possible form factors for operational use, possibly as a separate virtual machine on the host, or in a switch protecting many hosts. Each alternative has its strengths and drawbacks.

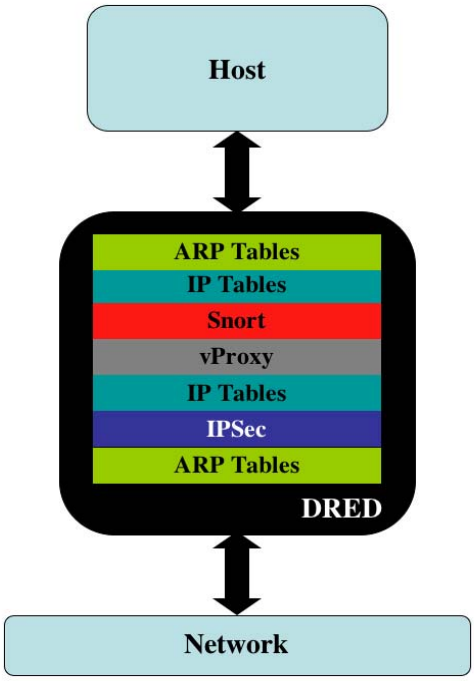

Figure 3. The DRED defenses operate at multiple layers of the network stack.

Because each DRED has ample storage and a powerful CPU running a full-featured operating system, it can support more sophisticated prevention, detection and response capabilities than previous generations of COTS hardware-based, distributed firewalls [9]. Figure 3 illustrates the defenses on the prototype SPDR DRED. The DRED performs MAC address filtering and stateful, deep-packet inspection on both its internal interface to the host and its external interface to the network. It also runs the Snort NIDS on its internal interface to detect attacks launched from a malicious insider on the host. On the external interface the DRED provides IPsec encryption, thus ensuring that any rogue connection to the network is blinded. This applies to rogue connections from authorized hosts as well as all connections from unauthorized hosts.

Finally, the DRED proxies all SOA traffic associated with the testbed applications. The *vProxy* detects and blocks malformed traffic and alerts the ISM of each message that is sent and the message's content. vProxy is a highly modified version of the *TinyProxy* http proxy [10] that uses Xerces-C for real-time XML validation. Each vProxy loads XML Schema for all of the legitimate SOAP mission traffic that the given DRED should see to and from its host. Messages that fall outside these fairly strict bounds will never leave the DRED. Thus, vProxy policy and configuration is specific to the applications and services provided by the associated host.

Coordinating policies for multiple defensives across many hosts is a non-trivial task [11]. To address this, DRED policies are generated automatically using Adventium's *Conversations* policy management tool, which separates the concerns of authorization and enforcement to guarantee policy consistency, both across defenses and between hosts, and to reduce the management burden by an order of magnitude. This makes it simple for SPDR to incorporate otherwise tedious protections such as static Address Resolution Protocol (ARP) tables into the deployed network.

The operational model for the DRED is conceptually simple. Until a user logs in directly to the DRED using a standard smart card, there is no network connectivity. This could be modified to provide limited connectivity to support necessary house-keeping functions, such as back-up and patch management, as long as proper remote administrator authentication and auditing are enforced. Following user login the DRED activates a user specific policy configuration that includes cryptographic keys, network monitoring and authorization rules and sensors and the appropriate proxy configuration.

The prototype DRED policies embody a strict concept of *Least Privilege*, authorizing only the network accesses required by the user's role. The sensors report all detected observables to the ISM, where they are distributed to the appropriate ISM components, as shown in Figure 4. The DRED also implements responses as instructed by the Response Module on the ISM. When the user logs out, the DRED policy returns to its default state.

The Least Privilege network authorization policies play a key role on SPDR. The ISM reasons about suspicious observed events and the extent to which those events may indicate malicious intent by a user. What is considered malicious varies from user to user according to the role of the user. Certainly unauthorized actions should be identified as suspicious, but it is also possible for a user to perform authorized actions with malicious intent. By constraining users to engage only in *authorized* behaviors, the DRED effectively reduces the potential search space that the

ISM must explore for malicious intent. Without this restriction the ISM would have to consider plans containing both authorized and unauthorized behaviors, which would be a much larger search space.

SPDR relies on DRED devices to protect all hosts in the system, including servers that do not have traditional users. In these cases, the DRED device's default policy specifies the authorizations that are appropriate for the server. It is also necessary to decrypt DRED-encrypted IPsec traffic before it leaves the SPDR-protected enclave.

3.2 The ISM

The ISM contains the online intelligent components used to identify, respond to, and attribute attacks based on sensor information provided by the DREDs and any network or host-based sensors that might be incorporated in the future. For the SPDR prototype the ISM only uses sensors on the DREDs. As shown in Figure 4, the ISM includes:

- A pair of application layer sensors, the Mission Monitor and the Provenance Monitor,
- The Plan Recognition Module,
- The Response Module, and
- The Attribution Module.

The Mission Monitor tracks the progress of the mission planning workflow. Events are supposed to follow in a particular (partial) order and stay within temporal bounds. The Mission Monitor detects when any task fails to register completion by its deadline (or is started out of order) and emits an alert indicating the sort of failure detected. The SPDR Response Module may choose to take corrective action when the mission is failing to make progress, e.g. restart the mission from a point at which it was making progress. This could involve replacing the individual, host, or service responsible for the lack of progress. For the mission planning application on our testbed, the workflow was linear with a limited ability for a user to modify the flow by aborting a planning loop. There is nothing to preclude the Mission Monitor from enforcing more complex workflows with branches and loops. However, this would require tighter coordination with the Provenance Monitor to understand how a specific instance of the workflow should branch or loop.

The Provenance Monitor is responsible for checking message integrity and for preserving message metadata used to help identify the cause of a mission failure and the user, host, or service responsible for the failure. A mission can fail because information in the mission plan was compromised or corrupted or because of delay in planning the mission. The vProxy on the DRED inspects each mission planning message and sends the Provenance Monitor metadata about the message. The Provenance Monitor performs an analysis of the message content to determine if any semantic invariants have been violated, and alerts the Plan Recognition Module of these integrity violations. The Provenance Monitor also stores the meta-data for use by the Attribution Module.

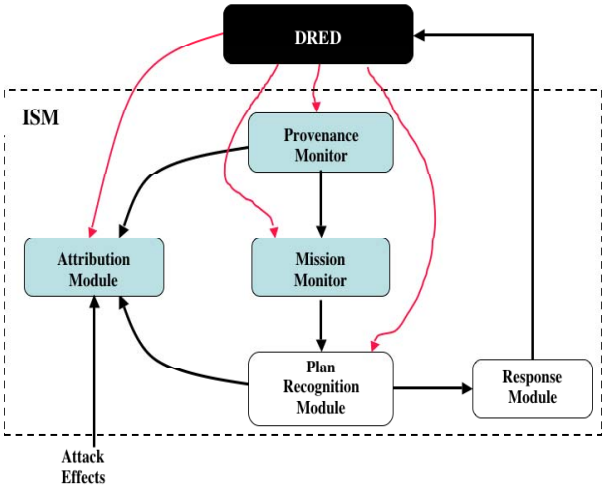

Figure 4. The ISM provides the intelligence to identify, respond to, and attribute attacks. Light (red) lines indicate DRED sensor alerts.

The Plan Recognition Module uses an efficient precompiled plan recognition engine that matches sequences of observations with plans in the library of possible attack plans generated by the offline planning engine [12]. The observations are selected alerts from the sensors deployed on each DRED and the Mission and Provenance Monitors. As new observations arrive the Plan Recognition Model updates its likelihood estimates for the possible attack plans underway.

The SPDR Response Module generates responses in a manner that balances the uncertainty of the true situation with the potential effects of the response. The prototype Response Module implements a fairly simple set of responses. It can direct the DREDs to ban a user from a specific host or from all hosts, or block network access from a specific host or from a specific service running on a host. In cases where the actual situation is suspicious but uncertain, it generates alerts that allow a system administrator or a duty officer to investigate the cause of suspicion, thus bringing a human into the response loop when appropriate. It is easy to conceive

of richer response policies, such as isolating a host and restarting its services somewhere else whenever a host or service has been corrupted, so that the mission planning loop can continue.

The Attribution Module, performs both online and *post mortem* attribution of attacks. It identifies the user or users most likely to be guilty parties, as well as to indicate potentially compromised hosts. This is done in a probabilistic fashion via a Bayesian belief network derived from the mission planning workflow [13]. The Attribution Module determines the steps in the planning workflow where the malicious insider might have acted. DRED authentication records inform the Attribution Module which users were apparently present on which hosts before or during the mission planning loop. Then observable attack effects, as evidenced by metadata from the Provenance Monitor and alerts from other SPDR components, are used to reason backwards to potential hostile causes and agents. For example, an attempt to transmit a maliciously crafted XML SOAP message from one service to another is detected by the vProxy and is attributed to the service and host that caused the inconsistency. Attempts to probe the network are detected by *Snort* on the DRED where the scans originate and are attributed to the related host. Inconsistencies introduced in the mission data is detected by provenance monitor, and attributed to the service and host that caused the inconsistency.

The Attribution Module also includes consequences of an attack in its *ex post facto* reasoning. If a planning loop fails to complete or completes with a corrupted result, or if post-mission analysis indicates the mission plan may have been compromised, the Attribution Module considers the nature of failure. It does so by using non-monotonic reasoning to attribute failure to one or more parties that may have had the *opportunity* to compromise, delay or tamper with the mission in a way that is consistent with the failure mode.

The output of the Attribution Module is a posterior probability distribution over the users (including "nobody") and another distribution over the hosts (including an unknown "rogue" host). The Attribution Module identifies the user logged on at the time of detection as well as any user previously logged in as suspicious—these users may have left trojan horse or remote control software meant to deflect blame. The granularity of attribution would be improved by including alerts from host-based sensors.

4. Results of the Red Team Assessment

The final test for SPDR was a rigorous three day assessment by the independent DARPA-sponsored Red Team. The assessment followed a seven month period during which the SPDR Blue Team worked with the Red Team to familiarize them with the SPDR testbed, design and implementation. The Blue Team provided the Red Team with the SPDR software and helped them with their analysis of the code. Until three weeks prior to the assessment, the teams worked collaboratively, discussing different lines of attack and the anticipated SPDR responses to those attacks. These interchanges took place via teleconferences as well as face-to-face meetings. They proved quite useful for improving the SPDR defenses.

Most notably, one week before the SPDR code freeze date the Red Team sketched a set of MAC layer attacks that they intended to use. During the next week the Blue Team managed to install a set of MAC layer defenses that thwarted all but one of the Red Team's attacks, an ARP flood denial of service from a rogue workstation. The natural way to thwart this attack would be to disconnect the rogue from the network at the switch port, but the testbed switch had not been defined as part of the SPDR system, so this response was off-limits during the assessment.

The Red Team developed forty-nine test cases, counting variations. Most of the test cases involved attacks against confidentiality, integrity, or availability. Some cases tested the extent to which SPDR would interfere with benign system activities. Such interference could take the form of responses to perceived attacks that prevented completion of a correct mission planning loop. It could also take the form of a false attribution of an attack when no attack was underway.

To mount a successful confidentiality attack the Red Team had to transmit all or part of a mission plan through the router representing the boundary between the carrier network and the rest of the world. For an integrity attack to succeed, the Red Team had to corrupt some part of the mission plan that was briefed. For both confidentiality and integrity attacks if SPDR detected the compromise or corruption prior to the command to briefing the flight plan, SPDR would cancel the mission, and the Red Team had failed to achieve its objective of compromising or corrupting a mission that was actually launched. To succeed with an availability attack, the Red Team had to delay the mission planning loop long enough so that the mission

could not be briefed within a pre-determined time window set to ten minutes for the assessment.

The teams spent the first day of the assessment *white-boarding* the test cases. In some cases the teams agreed on the expected results of the test and chose to score it based on that agreement rather than actually run the test. In other cases the teams determined that there would be value in running the tests and assessing the results. During this white-boarding session the teams dropped eighteen of the test cases as duplicates, outside the negotiated Rules of Engagement, or irrelevant to assessing SPDR's capabilities. Twenty-eight of the remaining cases were attacks; three were tests of whether or not SPDR would interfere with mission planning or accuse an insider when there was no attack.

In the following two days the teams ran all but a very small number of the remaining cases. They used a follow-up phone call to white-board the cases they did not have time to run. The tests included DoS and integrity attacks using floods at several layers of the network stack, ARP poisoning attacks, and spoofed or replayed messages, responses, and heartbeats,. Other attacks included attempts to delay, corrupt, or compromise a mission plan using malware or via direct human action. The Red Team used general tools and specially crafted tools to inject malware remotely and to inject spoofed messages or packets or to capture and replay them. The Red Team did not develop tools for exfiltrating information using covert or steganographic channels. By direction of the customer's Independent Evaluation Team, prevention of this sort of exfiltration was outside the scope of the project.

SPDR failed to thwart seven attacks. However two of these attacks were thwarted after very simple bug fixes, one to the Mission Monitor and one to the ordering of IP filter rules. The Red and Blue teams agreed that two more would have been thwarted if SPDR had included fairly obvious host-based sensors to detect the attacks. Another attack would have been thwarted if the SPDR response set had been extended to include actions at the network switch. Thus, these attacks identified limitations in the prototype implementation, not limitations to the SPDR approach.

Only two attacks exposed inherent limitations in SPDR's ability to thwart attacks. These were cases where the insider used legitimate accesses to either steal information or to corrupt information in semantically correct ways. In both these cases SPDR correctly attributed the attacks after the fact. In fact, SPDR correctly attributed all but one of the successful attacks, and after a simple two line bug fix, SPDR correctly attributed them all.

SPDR failed to attribute nine attacks. Five of these attacks were network and MAC layer DoS attacks that were thwarted trivially by the initial DRED defenses. No detection or response was required. SPDR could have correctly attributed four of these attacks by including known ARP sensors and snort detection rules in the SPDR sensor set. Attribution of the fifth would require sensing at the IPsec layer. We are investigating the feasibility of such sensing. After the same two line bug fix to the Mission Monitor identified above, SPDR attributed two more of these attacks, and another could be detected by including appropriate host-based sensors.

In summary, it appears that after two bug fixes, the addition of further low level network sensors and a small number of host-based integrity sensors, SPDR could attribute almost 100% of attacks and could thwart all but a small, albeit significant, class of attacks involving the insider's abuse of authorized access. It is not surprising that SPDR cannot thwart these attacks. After all, if the system supports a strong notion of Least Privilege access control, then the insider is using precisely the accesses required to perform critical functions on the network. However, it is very significant that SPDR can attribute this class of attacks after the fact because this means that eventually the malicious insider will be identified and apprehended and the organization can apply appropriate legal and policy-based penalties for the behavior.

It is worth noting that SPDR did very well with regard to avoiding incorrect attribution of attacks and interfering with legitimate network activities. There was only one case of incorrect attribution, and there were only three cases where SPDR interfered with legitimate activities. We suspect these low false positive rates are an artifact of the testbed and/or the test cases identified by the Red Team. It would be interesting to retest SPDR with a more complex network and a larger set of services.

5. Future SPDR Directions

A successful research project should identify new avenues of investigation, and the SPDR project has done this. The assessment validated the power and flexibility of the DRED as both a policy enforcement device and a tamper resistant sensor-effector for traffic between a host and a network. The Red Team did not identify any attacks that penetrated the DRED or bypassed its defenses, and the experience adding ARP defenses illustrates how easily new defensive mechanisms can be added to the DRED.

A logical next step would be to move toward the vision of an embedded device. Since the beginning of the SPDR project, several attractive commercial options have appeared [14], [15]. It would also be possible to virtualize the DRED by running it in its own partition on a virtual machine monitor and forcing all network access through the DRED partition [16], [17]. A virtual DRED would have more visibility into the host it is controlling, thus providing a relatively tamper-resistant location for host-based sensors as well as some more intrusive effectors than could be located on a separate device. On the other hand, there would be a greater risk that a savvy insider would be able to disable or bypass the virtual device.

As noted in the previous section on the Red Team assessment, the extension of SPDR to include a broader set of sensors, particularly host-based sensors, would increase its effectiveness considerably, pushing SPDR's attribution rate close to 100% and also increasing its ability to thwart attacks sooner. We conjecture that the inclusion of a relatively small number of ARP alert and snort sensor rules and information readily available in host and applications logs would suffice. The best way to test this conjecture would be to deploy SPDR on a larger network with a richer set of applications. The Attribution Module and the Plan Recognition Module would have to be extended to include reasoning about the resulting larger set of sensor alerts. This is not an immediate scalability concern since both of these modules performed very quickly during the assessment.

Once the validity of the conjecture is established, the next step would be to explore methods for maximizing the trustworthiness of host-based sensor reports. Certainly sensors implemented directly on operating systems under the control of the end-user would be highly suspect. However, there are better approaches. As noted earlier, virtualization would strengthen the defenses around the sensors. It is also likely that for more realistic architectures where the SOA services are hosted on servers in protected server rooms and users interact with the servers via relatively thin clients, many of the host-based sensors could reside in the server room, thus offering greater protection from most malicious insiders. This approach could work quite well for integrity attacks, but it will be inadequate for dealing with confidentiality attacks. Those are likely to require new classes of sensors that can detect and attribute sophisticated attempts at exfiltration. These sensors can be incorporated into SPDR when they are developed.

Besides being somewhat sensor poor, the current SPDR system has a relatively limited set of responses. Just as the way to increase the breadth of sensing is to use sensors that are already available, a good way to increase the breadth of responses is to take advantage of existing response mechanisms. As noted above, smart switches can provide reliable, effective responses to many classes of network DoS attacks. Redundancy and high availability solutions are also useful. Quite often an attack requires the corruption of integrity for the host or service under attack. Once SPDR detects such corruption, the most effective response is to restart the service, either on the same host or on a different, uncorrupted host, and to rollback to the last known good messages. The SPDR Provenance Monitor provides information for such a rollback. It would be interesting to see what else might be necessary to accomplish these *relocate and rollback* responses.

An underlying concern with scalability for the SPDR approach is the strong reliance of the ISM on a set of detailed network and adversary models for attack plan generation, and application models for use by the Attribution Module and the Mission and Provenance Monitors, and possibly for generation of specific instances of the vProxy. For the prototype these models were built by hand, but for the SPDR approach to generalize, this modeling process must include a fairly high degree of automation. On the SCOAP project we developed methods for automating the creation of network models, and we are investigating approaches for incorporating information from vulnerability and exploit databases into the adversary models.

To address the application modeling problem, it may be fruitful to investigate approaches for using an existing workflow modeling language [18] and approaches to provenance monitoring [19] to create a model that could be used to derive an initial version of the four SPDR components and then display the derivations for human review and editing. Then it would be useful to examine the feasibility of deriving the workflow model automatically from generally available development artifacts and to understand the limitations of scope associated with such derivations. We conjecture that it would be possible to extract such a workflow model from java source for a wide range of SOA application systems.

6. Conclusion

On the SPDR project we developed what we believe is a unique two-pronged approach to thwarting

and attributing attacks of malicious insiders. The first prong is to place critical sensing and response mechanisms in the DRED, a hardware sensor-effector platform that protects a single host, yet is not under the control of the host processor or the user currently logged into the host. The second prong is to apply powerful reasoning tools and application aware intelligent sensors to identify attacks based on inputs from the distributed sensors, to plan and implement responses, and to attribute the attacks to specific insiders.

In an assessment by a skilled independent Red Team, SPDR proved its worth. After fixing two small bugs, SPDR thwarted over 80% of the Red Team's attacks, and it correctly attributed all the attacks it failed to thwart. Ignoring several low level DoS attacks that SPDR easily preventted, SPDR attributed almost 90% of the attacks. Both percentages would have been higher if SPDR had used inputs from host-based sensors to complement sensors on the DREDs. The related false positive rates for incorrectly thwarting a perceived attack and falsely attributing a perceived or actual attack were quite low.

The only Red Team attacks SPDR could not have thwarted were attacks where the malicious insider used authorized access to either compromise sensitive information or to corrupt critical information in a semantically consistent manner. It is not surprising that SPDR fails to thwart these attacks, since thwarting them would require SPDR to either read the attacker's mind or to interfere with benign operations, thus preventing network users from doing their jobs. SPDR did correctly attribute these attacks after the fact, thus making such attacks highly unattractive to most would-be malicious insiders.

During the assessment the DRED proved to be a very strong policy enforcement and sensor-effector device. Many attacks simply bounced off the DRED with no impact whatsoever on the network. This included low-level network DoS attacks as well as user-level attempts to circumvent the DRED policies. These all failed, and the Red Team was unable to corrupt or bypass the DRED. It would be reasonable to begin efforts to deploy DREDs in limited operational environments. This could be done in one of several form factors, ranging from bump-in-the wire single board computers as was done for the prototype, to embedded physical or virtual devices.

SPDR is not a complete insider threat solution in itself. It will work most effectively when it is part of the larger network defenses, relying on alerts from existing sensors to complement the DRED alerts, and on other security and survivability mechanisms to complement DRED-based responses to attacks. Moreover, research on the prototype indicated the need for tools to automate the processes used to build the models used by SPDR's intelligent sensors and its reasoning components.

Further research should be performed on the investigation of approaches for automating these model-building processes and for integrating SPDR with the larger set of defenses available on an operational network. This should include the use of non-DRED sensors and effectors, and the exploitation of architectural features to provide a richer, more durable set of responses to attacks. The best way to do this would be to apply SPDR to a larger, more complex network.

Acknowledgments

This work was supported by DARPA and AFRL under contract number FA8750-07-C-0017. The authors would like to thank their reviewers for several very useful comments.

8. References

[1] William H. Webster, *et al.*, "A Review of FBI Programs", U. S. Department of Justice, Washington, DC, March, 2002. http://www.usdoj.gov/05publications/websterreport.pdf

[2] M. Keeney, *et al.* "Insider Threat Study: Computer System Sabotage in critical Infrastructure Sectors." Technical Report. U.S. Secret Service and CMU Software Engineering Institute. May, 2005.

[3] Ignacio J. Martinez-Moyano Michael E. Samsa James F. Burke Bahadir K. Akcam1. "Toward a Generic Exploring Insider Threats to Information Infrastructure," in Proceedings of the 41st Hawaii International Conference on System Sciences. 2008.

[4] E.D. Shaw, K.G. Ruby, and J. M. Post, "The insider threat to information systems." Security Awareness Bulletin, volume 2, 1998.

[5] J.S. Park and J. Giordano. "Role-based profile analysis for scalable and accurate insider-anomaly detection, in Proceeding of the IEEE International Performance Computing and Communications Conference. 2006.

[6] Suranjan Pramanik, Vidyaraman Sankaranarayanan, and Shambhu Upadhyaya. "Security Policies to Mitigate Insider Threat in the Document Control Domain," in Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC). 2004.

[7] E. Eugene Schultz. "A framework for understanding and predicting insider attacks." Compsec 2002. October 2002.

[8] Mark Boddy, Johnathan Gohde, J. Thomas Haigh, Steven Harp, "Course of Action Generation for Cyber Security Using Classical Planning", ICAPS-05, June, 2005.

[9] Charles N. Payne, Jr., T. R. Markham, "Architecture and Applications for a Distributed Embedded Firewall", 17th Annual Computer Security Applications Conference, New Orleans, LA., pp. 329-338, 2001.

[10] Computation Group, "Tiny Proxy Home Page," Banu, 2008. http://tinyproxy.banu.com/.

[11] Paul Rubel, Michael Ihde, Steven Harp, Charles Payne. "Generating Policies for Defense in Depth" in Proceedings of the 21st Annual Computer Security Applications Conference, 2005.

[12] Christopher W. Geib, Robert P. Goldman and John Maraist, "A New Probabilistic Plan Recognition Algorithm Based on String Rewriting," in: Proc. ICAPS 2008, September 2008.

[13] [Jensen01] F.V. Jensen, *Bayesian Networks and Decision Graphs*, Springer. 2001.

[14] Pranav Mehta, "Intel® Embedded Processor for 2008 (Tolopai) SoC Architecture Overview," Intel Corp., 2008. http://download.intel.com/technology/quickassist/tolapaisoc2008.pdf

[15] Harlan Tytus Beverly, "An Introduction to LLR2," Bigfooot Networks, June, 2007. http://www.killernic.com/technology/llr2.pdf.

[16] Reiner Sailer, Trent Jaeger, Enriquillo Valdez, Ramon Caceres, Ronald Perez, "Building a MAC-based Security Architecture for the Xen Opensource Hypervisor,"in Proceedings of ACSAC, 2005. http://www.acsac.org/2005/papers/171.pdf.

[17] M. Rosenblum, and T. Garfinkel, "A virtual machine introspection based architecture for intrusion detection," In Proceedings of the 2003 Network and Distributed System Security Symposium. 2003.

[18] Michael Adams, A.H.M. ter Hofstede, David Edmond, W.M.P. van der Aalst. •"Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows" in Proceedings of the 14th International Conference on Cooperative Information Systems (CoopIS 06), Montpellier, France, November, 2006.

[19] Luc Moreau, *et alia*, "The Provenance of Electronic Data". Communications of the ACM, vol. 51, No. 3, April, 2008.