Using Security Policies to Automate Placement of Network Intrusion Prevention

Nirupama Talele[1], Jason Teutsch[1], Trent Jaeger[1] and Robert F. Erbacher[2]

[1] {nrt123,teutsch,tjaeger}@cse.psu.edu,
Systems and Internet Infrastructure Security Lab
Pennsylvania State University
[2] Robert.F.Erbacher@us.army.mil,
Network Security Branch
Army Research Laboratory

Abstract. System administrators frequently use Intrusion Detection and Prevention Systems (IDPS) and host security mechanisms, such as firewalls and mandatory access control, to protect their hosts from remote adversaries. The usual techniques for placing network monitoring and intrusion prevention apparatuses in the network do not account for host flows and fail to defend against vulnerabilities resulting from minor modifications to host configurations. Therefore, despite widespread use of these methods, the task of security remains largely reactive. In this paper, we propose an approach to automate a minimal mediation placement for network and host flows. We use Intrusion Prevention System (IPS) as a replacement for certain host mediations. Due to the large number of flows at the host level, we summarize information flows at the composite network level, using a conservative estimate of the host mediation. Our summary technique reduces the number of relevant network nodes in our example network by 80% and improves mediation placement speed by 87.5%. In this way, we effectively and efficiently compute network-wide defense placement for comprehensive security enforcement.

1 Introduction

Many security administrators rely on network monitoring and Intrusion Detection and Prevention Systems (IDPS) to protect the hosts in their networks from remote adversaries. An Intrusion Detection System (IDS) inspect information flows to detect any malicious activity while Intrusion Prevention Systems (IPS) block remote access when such malicious activity is detected. When an IDS detects a malicious packet, either it may log the packet, allowing an administrator to take further action, or it may drop the packet to protect the receiving host process. An intrusion prevention system is often used to detect malware in packets and block denial of service attacks.

Despite widespread use of IDPSs and the deployment of host security mechanisms such as host firewalls and mandatory access control, the task of security practitioners is still reactive, responding to vulnerabilities as adversaries identify them. The IDPS can be classified as network based or host based, where the former monitors the network for suspicious activity and the latter monitors a single host for malicious activity. We highlight two key reasons for the lack of security methods. First, network based monitoring is inherently incomplete, as only certain threats can be identified and/or blocked at the network without creating false positives. For example, only known malware is blocked by the network based IPS, so that no valid functionality is accidentally blocked. Second, systems do not coordinate network monitoring with host defenses, resulting in security loopholes. For example, the host may overlook remote threats that a network based IPS cannot block, or a compromised process may propagate a remote threat to another host.

In order to proactively block remote adversaries, one must defend against all adversary accesses. We wish to monitor only a small number of mediation points so as to minimize resource costs. Researchers have previously explored methods to compute minimal cost placements for network based IPS configurations. These methods only focus on network flows [2, 35, 4] or utilize heuristic models of potential host vulnerabilities, as in methods for computing *attack graphs* [24, 14, **?**], to guide placement choices. Security under these frameworks rely on heuristic models of possible attacks (e.g., host scans [22]), which may miss previously unseen attacks and remain vulnerable to minor host configuration changes. More specifically, we identify two major limitations in the current models of attacks: (1) they do not account for the hierarchical structure of network-connected resources into subnets, hosts, and individual processes to represent possible attack paths and (2) they fail to account for network defenses, such as labeled network connections [13, 27, 17].

Rather than just computing a minimal placement for network based IPS, we compute minimal mediation placements for the network and host flows. Such mediation must account for both the network based IPS as described above as well as the host mediation necessary to enforce a set of security properties. In this paper, we develop a method that utilizes the available security policies on commodity operating systems and networks to compute *mediation placements* which automatically block adversary access to security-critical data. Our new method, which places mediators based on authorized data flows and security policies at the network and host levels, yields robust defense. We find that those defenses that cannot be enforced by network based IPS must be implemented by host mediation. The proposed method produces this necessary host mediation, given the IPS capabilities and constraints.

We implement a two-stage algorithm for computing network-wide mediation placement. The idea behind this method is that, where possible, our method replaces host mediation with network based IPS and vice versa. In the first stage, we compute conservative host mediation for a worst-case set of remote threats for the hosts. Our mediation suffices to protect the hosts in any deployment without network defenses. In the second stage, we summarize data flows within each host by utilizing the conservative host mediation. We observed that host data flow summaries built from conservative host mediations reduce the number of nodes in the example network graph by over 80% and the number of edges by more than

60%. Using these summaries, the automated network-wide mediation placement time is reduced by 87.5% for the example network when the summaries are produced in advance, which is feasible in many cases. This result demonstrates the feasibility of automated, comprehensive, network-wide defense placement.

Contributions: The first contribution of this paper is combining the host data flow graphs as computed in [19] with the network data flow and generating a hierarchical encapsulated graph model representing information flows in both network and host in order to compute near minimal defense placement for the entire network. The second contribution is the optimization in the conservative host mediation placement, where the method replaces the host mediation with network based IPS wherever possible. And the third contribution of this paper is summarization of the data flow graph for each host utilizing the conservative host mediation to address the scalability in huge networks.

The remainder of the paper is organized as follows. Section 2 provides background on network defenses and host security mechanisms, and defines the mediation placement problem. Section 3 defines an information flow problem whose solution is also a solution for the mediation placement problem. Section 4 outlines the design of our method. Section 5 describes the evaluation platform and experimental results. Section 6 concludes the paper and identifies future work.

2 Background

In this section, we identify the need for mediation placement in networks of hosts and justify the need to consider the available security policies in producing such placements.

2.1 Network Scenario

Figure 1 shows a typical modern networked application. Such applications consist of servers and their clients, where clients may be deployed in either wired or wireless networks. In many cases, clients and servers perform security-critical processing, assuming that the application data is protected from unauthorized modification or leakage and application data is available when necessary.

However, networked applications face a variety of threats. First, remote adversaries may launch attacks on processes that are accessible to the network at large. These processes often include custom programs such as PHP web applications on servers and unprivileged applications on clients with network access. Many system compromises now start by attacks on unprivileged processes. Second, processes on hosts within the network may launch attacks against other hosts. Such attacks may focus on system services by exploiting trust among hosts inside the network (and unauthenticated network protocols) or leverage the openness of wireless networks. Third, remote adversaries who are able to compromise an unprivileged process or trick users into installing untrusted data may launch local exploits against more privileged processes to install root kits

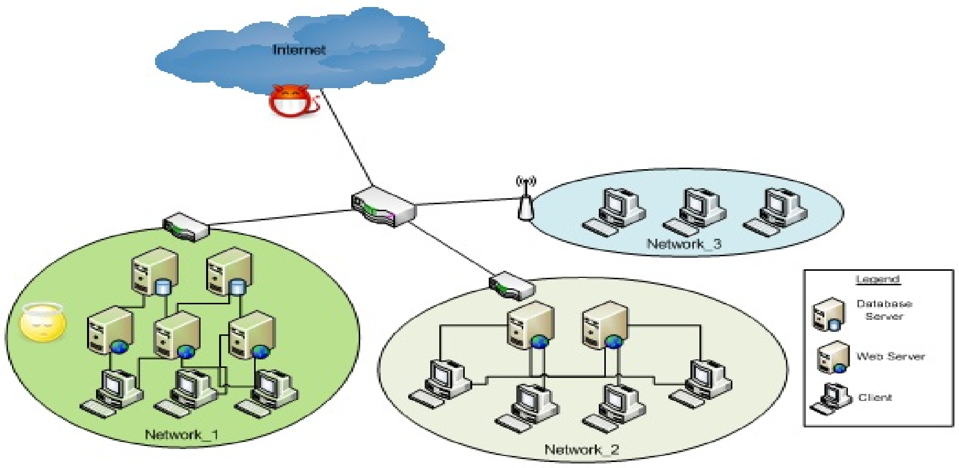

Fig. 1. Example Networked Application

or obtain administrator privileges. Unfortunately, the number and variety of possible local exploits available to adversaries is beyond enumeration at present.

A security problem occurs when an adversary can execute a sequence of operations that results in access to unauthorized data or excessive use of data processing resources. Remote adversaries use access to available networks to find hosts with vulnerabilities necessary to obtain these goals. Modern systems block many trivial vulnerabilities, yet adversaries often find short sequences of compromises, including combinations of the unprivileged networked processes and local exploits described above, that lead to security breaches. As a result, security practitioners must block all *attack paths* [24, **?**,32], but the variety of possible attack paths (even short ones) has proven too complex for manual configuration.

2.2 Network Defense Placement Problem

A common method for protecting hosts from adversaries is IDPS. IDS can see all the network connections being utilized[3] and examine the network packets that are being transmitted. Firewalls [5] now support powerful forms of deep-packet inspection to compare contents to attack signatures.

The common view asserts that proper defense placement depends on the type of network. For wired networks, IPS are often placed at network edges because all traffic must enter or leave via this choke-point. For wireless networks, IPS are placed on each node because other nodes in the wireless network may be untrusted. However, such placements may not effectively limit adversaries and/or may result in redundant monitoring. For wired networks, defenses at the edges ignores threats internal to the network, so if an adversary can compromise

[3] Some uses of IPsec hide addressing information, but IDS is often placed where such information can be obtained, such as gateways.

a single process on a single host they can then launch further attacks undetected. For wireless networks, per node monitoring may be unnecessary in some cases because certain kinds of attacks can be prevented at the edges. For example, a distributed denial of service attack can be thwarted external to the cell. In both types of networks, local exploits are invisible to the network based intrusion detection infrastructure, meaning the IDS placement has only a limited view of possible attack paths.

With the widespread deployment of mandatory access control (MAC) in commodity systems over the last ten years [28, 26, 37, 33], adversary access within hosts can be restricted, although such restrictions do not block all adversary attack paths on the host. This MAC enforcement has been used primarily to confine network-facing daemons to prevent compromised root process from compromising the system at large. However, such enforcement does not prevent local exploits. Windows Mandatory Integrity Control [18] (MIC) is designed to prevent untrusted code (e.g., downloaded from the Internet) from modifying privileged resources, but does not prevent adversaries from tricking victims into reading untrusted data or upgrading untrusted code. As a result, we advocate development of a method to compute IPS placements that account for the host and network configurations.

Related Work: Many efforts have been made to verify the data flow in a policy [**?**,**?**,**?**,**?**], especially to assist in designing the policies and to identify whether adversary can perform an unauthorized operation on the component. There are also many attack graph based methods to verify the network policies [11, **?**] which, as discussed above are heuristic based methods and may fail to represent the actual host behavior. However, the composed behavior of these arbitrary policies in hosts and networks is complex to analyze and may interact in unpredictable ways. Researchers have recently explored methods to place minimal but comprehensive network defenses automatically by solving graph problems, such as vertex cover and multicuts [2, 35, 4, 25]. While these problems are computationally complex in general, efficient greedy algorithms exist to produce effective solutions. Again, these methods make broad assumptions about the host that may misrepresent the actual attack paths within the host.

Project Goal: Our goal is to develop a method that computes comprehensive and minimal *mediation placement* for networks of hosts. Such method must utilize the host security policies in addition to those in the network to provide an accurate model of adversary access. Further, this method must account for the possible mediation capabilities in both network devices and on the hosts themselves. As discussed above, finding the minimal solution is impractical for known methods for general security policies, so the method must also leverage practical insights to produce effective approximate solutions.

3 Information Flow Problem

In this section, we show that the network and host mediation placement problem can be expressed as an information flow problem. Traditionally, an information flow problem is defined as follows:

Definition 1 *An* information flow problem*, $\mathcal{I} = (\mathcal{G}, \mathcal{L}, \mathcal{M})$, consists of the following concepts:*

1. *A* directed data flow graph $G = (V, E)$ *consisting of a set of nodes V connected by edges E.*
2. *A* lattice $\mathcal{L} = \{L, \preceq\}$*. For any two levels $l_i, l_j \in L$, $l_i \preceq l_j$ means that l_i 'can flow to' l_j.*
3. *A* level mapping function $M : V \rightarrow \mathcal{P}^L$ *where $\mathcal{P}^L$ is the power set of L (i.e., each node is mapped either to a set of levels in L or to $\emptyset$).*
4. *The lattice imposes security constraints on the information flows enabled by the data flow graph. Each pair $u, v \in V s.t.$ $[u \hookrightarrow_G v \wedge (\exists l_u \in M(u), l_v \in M(v).\ \ l_u \not\preceq_{\mathcal{L}} l_v)]$, where $\hookrightarrow_G$ means there is a path from u to v in G, represents an* information flow error*.*

It has been shown that information flow errors in programs [21] and MAC policies [12, 31, 3] can be automatically found using such a model.

However, resolving such information flow errors has been a complex manual task. In general, information flow errors can be resolved by changing the data flow graph (e.g., removing nodes and/or edges) or adding *mediation* to change the level of data propagated by information flows. However, changing the data flow graph is difficult in practice because it implies a change in the operations a system may perform, which may prevent one or more components from functioning correctly.

Researchers have developed methods to generate minimal mediation to automatically resolve information flow errors by independently proposing graph-cut-based solutions for MAC policies of programs [15, 16] and systems [30]. A graph-cut solution identifies a minimum cost set of mediators R defined as $R = \{((u, v), l) \mid (u, v) \in E \wedge l \in L\}$, where edge (u, v) is a *cut-edge* and l is the data *security level* sent by u to v due to mediation, resulting in the mapping $M_v = v \rightarrow l$ being assigned to the edge's destination v. That is, each cut-edge relabels the information received by v from u to l. This set of cut-edges in R resolves all information flow errors in the information flow problem $\mathcal{I}$, according to the Cut-Mediation Equivalence [16].

In a recent paper [19], we extended the basic graph cut problem to account for the integration of independent components into a composite and coherent data flow graph. The method also accounted for the constrained ability of partially trusted components to mediate information flows, and proposed a strategy for placing mediation that implemented a classical integrity model [6]. In general, the information flow policy may be a partially ordered set of permissions, and therefore an optimal mediation solution requires us to solve a multicut problem.

In light of the intractability of mutlicut [7], we apply a greedy method to produce an approximate solution.

We find that by solving this information flow problem, we can also produce a network defense placement. Further, by accounting for host and network data flows comprehensively, we produce a network defense placement that accounts for attack paths more accurately. However, the unwieldy size of the combined host and network data flows make obtaining these solutions computationally infeasible. Indeed, the data flow graph for each host consists of 2000–3000 nodes and 6000–12000 edges. Given that our greedy multicut graph cut algorithm runs in $O(|L| \cdot n^3)$ where n is the number of nodes in the data flow graph, only networks with a modest number of hosts can be considered in practice. In this paper, we develop an approximate (greedy) solution along with host summaries for network defense placement that accurately accounts for network and host data flows.

Assumptions. The key assumption in this work is that the devices, operating systems, and programs that enforce security policies, do so correctly. This is a significant assumption given the size and complexity of such components, but it is also a standard assumption in modern computing systems. Specifically, we assume that the devices, operating systems, and programs that enforce security policies satisfy the *reference monitor concept* [1], which requires that a reference validation mechanism (i.e., MAC enforcement) "must always be invoked" upon a security-sensitive operation, "must be tamper proof," and must be "small enough to be subject to analysis and tests, the completeness of which can be assured," which implies correctness. The reference monitor concept is certainly the goal of several commodity reference validation mechanisms, although they do not meet the latter of these requirements.

4 Design

In this section, we design a method for computing mediation placements for network and host data flows. First, we compute the conservative mediator placement required for each unique host configuration in the network. Second, we produce summaries of the resultant host information flows, accounting for the placed mediation. Third, these summaries are used to produce an information flow problem for the entire network whose solution is a mediation placement that replaces host mediation with network defenses when the latter reduces the number of mediators required overall.

4.1 Host Mediation

In the first step, we describe a method to compute a conservative host mediation that would protect the host when there is no network defense. Our fundamental task is to constrain the information flow problem to the union of sufficient subproblems, as we use a graph-cut method described elsewhere to compute these submediations [20].

A host is defined by its internal data flows and network connections. Many commodity systems are now deployed with *mandatory access control* (MAC) policies [28, 26, 33, 37]. MAC policies define the legal operations of subjects on objects in the host. We compute the internal data flow graph among subjects and objects of a host from its MAC policy using well-known techniques [36, 12, 31, 3].

In addition to the host data flows, some subjects in the host may have access to the network. The combination of the MAC policy data flows and network data flows for host subjects forms the host data flow graph $G = (V, E)$. The network access is represented by sets of input and output nodes, $I \subseteq V$ and $O \subseteq V$ respectively, and edges that identify which MAC policy subject nodes can access the network nodes. For the computation of individual host mediation, each element in I has an indegree of 0, and each element of O has an outdegree of 0. For many firewall rules, the port uniquely identifies the MAC policy subject that can access the network, but for some client ports such connections may be ambiguous. These must be identified before analysis.

To produce an information flow problem, we must produce a lattice policy $\mathcal{L}$ and map the lattice levels to the appropriate nodes in the data flow graph by defining a mapping function M, see Definition 1. For OS distributions which specialize in single applications (e.g., web server, database, etc.), we associate lattice levels with the kernel and application labels in the MAC policy by specifying such mapping functions [19]. The input nodes are mapped to the lattice level for the expected input data. Typically, we assign the input nodes to the level for remote adversaries because most network inputs are untrusted. The input node mapping must represent the worst-case scenario for the host, as there is no network based defenses at this stage.

In Section 3, we stated that a solution consists of a set of mediator edges $R = \{((u, v), l)$ where $(u, v) \in E$ and $l \in L\}$ and finding such a solution is a multicut problem [7] for a general lattice. Finding a minimal solution to the Information Flow problem is NP-hard. For this reason, we employ a greedy algorithm rather than attempting to obtain an exactly minimal solution. The greedy method for producing a solution to the information flow problem solves a graph-cut problem [9] for each prohibited pair of lattice level and union all the cuts. If we have k such pairs, then the greedy solution obtained is no greater than k times the optimal cut. The reason for this is that the optimal cut can be no smaller than the size of the minimal cut for an individual pair, and the algorithm makes k such cuts. We order the lattice problems to take advantage by reuse of solutions, as described in [19]. By solving the resultant information flow problem, we produce an approximation of the minimal host mediation R at the program entry points (i.e., program instructions that invoke the system call library [11]) necessary to protect host processes from the specified remote threats. The soundness can be argued to be true given we consider each pair and solve them individually to obtain the result, thus by construction of the solution there are no false negatives for any prohibited lattice level pair in the system.

4.2 Host Summaries

In this section, we use the host mediation placement to summarize the flows within the hosts to reduce the size of the network-wide information flow problem. A summary of a host data flow graph consists of the data flows from the host's input nodes to its output nodes and shows how data received by this host is propagated to other hosts via its output. We define the function $\mathit{Reach}(S,T) = \{(s,t) \mid s \in S, t \in T, s \neq t \wedge (s \hookrightarrow_G t)\}$. In general, the data flows through a directed graph G can be summarized as $G' = (V', E')$, where $V' = I \cup O$ and $E' = \mathit{Reach}(I,O)$.

To accurately capture information flows through a host, we must also account for the mapping function M, which defines where data of particular security levels host imports, and the host mediator placement R computed in the previous section, which defines where the security level of the data on a particular data flow is changed. Since the mapping function and mediators affect the security level of the data the host produces, these must be accounted for[4].

First, we leverage the knowledge that the mapping function $M : V \to \mathcal{P}^L$ identifies a set of nodes $A \subseteq V$ that are mapped to lattice levels in L. Secondly, application of a mediator also changes the information flows through the host. A mediator $((x,y),l) \in R$ has two effects: (1) it filters the flow through edge $(x,y) \in E$ in the graph and (2) it maps a new level $l \in L$ to the node y. That is, a mediator causes the receiving node of an edge to receive data mapping to the lattice level of the mediator. As a result, we retain the mediator edges (R_x, R_y) in the summary. This results in the following definition of a summary graph.

Definition 2 *For graph G given the input sets I, O, R_x, R_y and A such that R_x and R_y are sets of source and target nodes of mediator edges in R respectively. A* summarized data flow graph *G' is a directed graph $G' = (V', E')$, where $V' = I \cup O \cup A \cup R_x \cup R_y$ and $E' = \mathit{Reach}(I,O) \cup \mathit{Reach}(R_x,R_y) \cup \mathit{Reach}(I,R_x) \cup \mathit{Reach}(R_y,A) \cup \mathit{Reach}(A,O)$.*

That is, the outputs are either based on the input data (edges in $\mathit{Reach}(I,O)$) or based on the mapped data (edges in $\mathit{Reach}(A,O)$) which may be combined with some input data and mediators (edges in $\mathit{Reach}(I,R_x)$ and $\mathit{Reach}(R_y,A)$). Note that if there is a flow from node $i \in I$ and a flow from node $a \in A$ that merge at some node $x \notin O$, the correct output flows will still be produced. Either there is a path from x to node $o \in O$ causing $\mathit{Reach}(I,O) \cup \mathit{Reach}(A,O)$ to include edges (i,o) and (a,o) replicating the merge, or neither reaches a node in O meaning that no edges are produced. Figure 2 represents the summarization of a host graph where all the relevant paths from inputs to mediators and mapped nodes, and from mapped nodes to outputs are retained.

The claim is that the summarized data flow graph includes all the edges necessary to compute the output information flow of the host accurately.

[4] When data of different security levels is combined, the resultant security level is the least-upper bound of the input levels, as defined by Denning's Lattice Model [8].

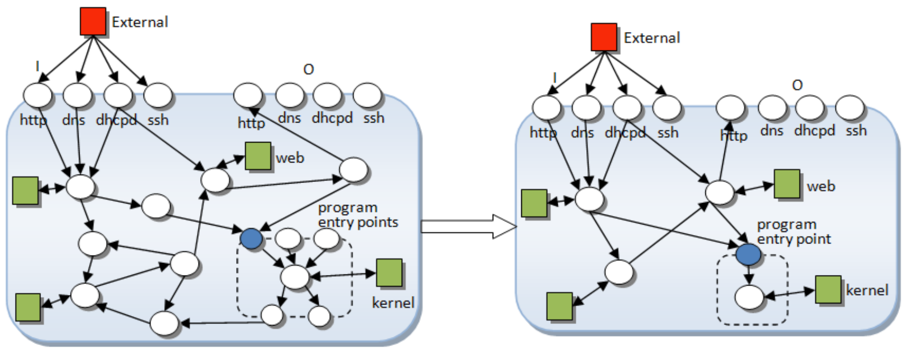

Fig. 2. Graph Summarization: Where the box represents the security level mapping to the node, green is trusted and red is untrusted. The blue nodes are nodes which can be selected for mediation(R_x)

Theorem 3 *The* summarized data flow graph G' *constructed using Definition 2 for a directed graph G with a particular mapping function M, lattice $\mathcal{L}$, and set of mediators R produces the same information flows to all nodes in O as G, regardless of the security levels mapped to the inputs nodes I.*

Proof Sketch: The information flows that reach nodes in O in G are a combination of flows propagated from the inputs to the outputs ($Reach(I, O)$), flows from the mapped nodes and mediators to the outputs ($Reach(A, O)$), flows from inputs to the mediator nodes ($Reach(I, R_x)$), and flows from mediator nodes to other mapped nodes ($Reach(R_y, A)$). As long as we include all the relevant flows that alter the security level while computing the summary graph G', the flows $Reach(I, O)$ and $Reach(A, O)$ will capture only the flows to the nodes in O in summary graph G'.

4.3 Network-Wide Mediation Placement

Given the summarized data flow graphs for each host and the network policies that define the data flows among hosts, it is possible to compute a IPS placement. The goal is to replace the host mediation with network IPS wherever possible such that this replacement reduces the overall number of mediators required. To do so, we compute a multicut for an information flow problem built from the composition of network and summarized host data flows. The result is guaranteed to require no more than the number of mediators in the conservative host mediation for all network hosts.

We solve this problem by building a second information flow problem covering the network flows and summarized hosts (see Definition 1). The data flow graph of this information flow problem is now a combination of summarized host data flow graphs (from the previous section) and the network data flows that connect hosts. The network data flows are derived from the possible network connections in the particular network type (e.g., wired or cellular) and the firewall policies of the network (e.g., edge servers) and hosts.

The recursive method for composing the hierarchical model for network and host data flows is presented in Algorithm 1. The algorithm performs a postorder precessing where the parent node can only be processed after the child nodes in order to add the required edges. As networks of hosts are organized hierarchically and flows between hosts or networks are encapsulated, we use an encapsulated, hierarchical graph model to represent the composite data flow graph [23, 10]. An advantage of such a graph model is that we can plug summaries of hosts and even networks into a data flow graph easily. The input and output nodes of child host are projected at the parent node to provide an interface with the external world. The child elements are connected as per the network configuration provided. For instance in the wireless network $Network\_3$ all hosts are interconnected, while in the wired network $Network\_1$ all child hosts are not necessarily interconnected and communicate as defined by the topology.

Algorithm 1 Generate Hierarchical Network Graph

```
Input: $host$ contains the allowed policy flows in a host and $net\_conf$ contains the
    allowed flows by firewall and network configuration files
Output: Hierarchical network graph model $Network\_M$
 1: function $Gen\_Network\_Graph(host,\ net\_conf)$
 2: //Recursively build all child hosts
 3:     $N = host.Children.Count$;
 4:     for $i = 0\ to\ N$ do
 5:         $Network\_M.child[i] = Gen\_Network\_Graph(host.child[i])$;
 6:     end for
 7: //Process parent node: Generate Network Edges
 8:     $Add\ interface\ for\ child\ I/O\ ports$
 9:     for $(all\ u, v\ in\ Network\_M.V)$ do
10:         if $((u, v) \in host.flows$ and $(u, v) \in net\_conf.flows$ then
11:             $Network\_M.E = Network\_M.E \cup (u, v)$;
12:         end if
13:     end for
14:     $Network\_M.Graph = M(Network\_M,\ L)$; //lattice mapping function.
15:     $Summarize(Network\_M)$; //Summarize host graph
16: end function
```

We configure the lattice and mapping function for the information flow problem as follows. Since network devices are not really visible to the hosts, they do not introduce any new lattice levels or mappings[5]. However, we do need to identify the sources of network adversaries to map the threats (i.e., adversarial security levels) to their actual network locations.

We note that not all network devices may be capable of mediating all host requirements. For example, a network IPS may scan for known malware, but a host process that is accessible to an adversary must protect itself from any malicious input, known or unknown. To express such limitations in solving information flow problems, we express constraints on edges [19]. Such constraints

[5] Of course, we may want to place mediation to protect the network devices, but that is not the focus in this paper.

associate an edge with a lattice level and prevent any mediator assigned to that edge from declassifying data above that lattice level.

As a result, the set of network devices may not be capable of solving the information flow problem. Thus, we retain the possibility of using the host mediators in addition to network IPS to solve the information flow problem. These two sets of locations are the only possible mediation locations for this information flow problem. In the worst-case, the conservative host mediation produced in the previous section will be used to protect the system, but we apply network IPS where it reduces the cost of host mediation.

To compute a network-wide mediator placement, we solve the information flow problem above by computing a set of mediators that resolve all information flow errors for the network-wide data flow graph. Using the summarized data flow graphs and the conservative host mediations computed in Section 4.1, we see that the method shifts host mediation to network IPS where possible. $\mathcal{R}$ is the union of all the conservative host mediations for all the systems hosts in the network. Since we computed $\mathcal{R}$ using the worst-case input mapping, $\mathcal{R}$ is a solution to the network-wide information flow problem. The claim is that any network IPS that leverages the above solution to this information flow problem only reduces the amount of overall mediation required.

Theorem 4 *Given a conservative host mediation for a set of hosts $\mathcal{R}$ consisting of T edges, a mediation placement can be found that solves the information flow problem containing these hosts in a network data flow graph constructed as described above and the number of mediators in that solution is less than $|T|$.*

Proof Sketch: A cut problem is created for each lattice level $l_i \in L$ mediated by the conservative host mediations $\mathcal{R}$, resulting in $|L_{\mathcal{R}}|$ cut problems for $L_{\mathcal{R}} \subseteq L$ levels mediated total. A conservative host mediation has a set of edges T_i that mediate to level l_i. Note that this set forms a valid cut solution of size $|T_i|$ for the cut problem to l_i. Thus, the union of these cut solutions is a valid multicut that resolve all information flow errors of size $\sum_{i=1}^{|L_{\mathcal{R}}|} |T_i|$. However, any solution of edges S_i for lattice level l_i must be of size $|S_i| \leq |T_i|$, because the cut problem solution is exact. Thus, the sum of these sizes of the cut solutions for each level l_i, $|S_i|$, must be no greater than $|T|$.

5 Evaluation

In this section, we describe the prototype implementation for a sample networked application. We base the analysis of individual host mediation on our previous work [19] with a new hierarchical modeling and host summarization module implemented in C++. Figure 1 shows the experimental setup with 16 hosts in a sample network configuration. The network hosts include a collection of web servers, database servers, and web clients. Each of the network host is a VM that runs the Linux 2.6.31-23-generic kernel and enforce SELinux refpolicy 2.20120725 [28] with different module configurations. Each host policy is different

Host type	Nodes (G)	Edges (G)	Nodes (G')	Edges (G')	Reduction Nodes	Reduction Edges	Time (sec)
Web Server(5)	2050	6660	309	2509	85%	62.3%	192.1
Database Server(2)	2578	10071	359	2179	86%	78.3%	267.26
Web Client(8)	2978	11499	479	2332	84%	79.7%	302.57

Table 1. Impact of summarization on individual host graphs.

Network	Nodes	Edges	Mediators	Time(min)
Before Summarization	39,235	145,476	4,745	34.19
After Summarization	6,176	36,031	4,745	4.27
Reduction Percentage	84.2%	75.2%	N/A	87.5%

Table 2. Performance gain with summarized hosts for sample networked application

and supports distinct set applications. The web server VM runs an Apache web server and web application, database VM runs MySQL and, client VMs run a web browser. Network 1 and 2 are both wired networks with one at a higher security level than the other. The hosts in Network 2 are not directly connected to each other and communicate via the network node, while some hosts in network 1 can directly talk to each other. Network 3 is a wireless network and hence we have modeled as all nodes communicating with each other since potentially nodes can listen to all transmitted data. This type of modeling can address the attacks like RP poisoning.

We use individual firewall (iptables) rules to determine a host's interaction with the outer world and the network configuration files to determine the overall organization of hosts in this networked application. We now describe the impact of summarization on the computation of network-wide mediation placement considering the optimizations from the conservative host mediation. We also demonstrate the joint optimized analysis of host mediations and network IPS.

We compute the summary graph for our hosts as described in Section 4.2. Table 1 shows the reduction in the graph for individual hosts on computing the host summary with conservative mediation placement. For each of the hosts, the number of nodes are reduced by over 80% and the number of edges are reduced by over 60%. We incorporate the conservative mediation placement, since only about 10% of the nodes can be removed without it, as nearly all processes may be capable of performing some kind of mediation. Table 1 specifies the average time needed for generating the summary graph for each kind of host. The summarization needs to be computed only once after which the summarized host can be reused multiple times for analyzing in different network environments.

Table 2 shows the impact of summarization on computing time for mediation placement. We compute a mediation placement before and after host summarization and find an 87.5% reduction in the placement analysis time for the example network. The total number of mediation placements is 4745, out of which only 613 mediators are unique. A unique mediator is counted only once even if it is used in different hosts. A program may appear in multiple hosts, thus filtering such repetition gives us a reduction of 87% in terms of the effort required for deploying the mediators.

Table 3 shows the average reduction in the host mediation for each type of host when network IPS is available. It also shows average time required to

Host type	Mediators (Conservative)	Mediators (Network)	Time-Conservative mediators (sec)
Web Server(5)	258	214	62.86
Database Server(2)	308	229	80.55
Web Client(8)	429	283	93.55

Table 3. Mediator placement: conservative vs in a network.

generate the conservation mediations for each type of host. In this example, we aim to compute the necessary mediation to block denial of service attacks as an information flow problem. The conservative host mediation shows the number of program entry points necessary to block paths from the network inputs to protect the application (e.g., web server, database, web clients) and system resources (e.g., critical kernel files). We assume that the network IPS can block application-specific malware directed at the application over the network. As only a small number of paths exists directly from the network to these applications, and because malware may compromise system processes by passing through applications, we can block only a fraction of the threats with network IPS, even with complete malware detection at the IDS. This result can help system administrator identify what defenses are needed in the host and what can be entrusted to network in a particular configuration. The experiment can be performed on various network states at static time and it is part of future work to adapt the analysis to handle dynamic network. The method determines the placement, such that no untrusted data can reach a trusted object without going through the appropriate mediator, enforcement of which has to satisfy the reference monitor concept as expressed in section 3.

6 Conclusion

In this paper, we proposed a method for computing a minimal placement for network defenses among network and host flows. While networks with multiple hosts can have many flows, we demonstrated a feasible approach which views network IPS as a replacement for certain host mediations. We designed an algorithm based on summarization of host flows and a conservative estimate of the required host mediation that can reduce the size of the information flow problem by more than 80% and mediation placement computation time by 87.5% for the considered sample. Thus, our method provides automated, network-wide defense placements which comprehensively enforce security. We also observe that hosts performing similar functionality with similar security requirements use mediations at the same entry points. By the insight gained while performing this experiment, we found that the redundancy in networks of systems offers future opportunities for host based summarization.

References

1. J. P. Anderson. Computer security technology planning study, Volume II. Technical Report ESD-TR-73-51, Deputy for Command and Management Systems, HQ Electronics Systems Division (AFSC), October 1972.

2. Y. Breitbart, F. Dragan, and H. Gobjuka. Effective monitor placement in internet networks. In *Journal of Networks, North America*, 2009.
3. H. Chen, N. Li, and Z. Mao. Analyzing and comparing the protection quality of security enhanced operating systems. In *NDSS*, 2009.
4. Xian Chen, Yoo-Ah Kim, Bing Wang, Wei Wei, Zhijie Jerry Shi, and Yuan Song. Fault-tolerant monitor placement for out-of-band wireless sensor network monitoring. *Ad Hoc Networks*, 10(1):62 – 74, 2012.
5. William R. Cheswick, Steven M. Bellovin, and Aviel D. Rubin. *Firewalls and Internet Security; Repelling the Wily Hacker*. Addison-Wesley, Reading, MA, second edition, 2003.
6. D. D. Clark and D. Wilson. A comparison of military and commercial security policies. In *IEEE Symposium on Security and Privacy*, 1987.
7. E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM J. Comput.*, 23:864–894, August 1994.
8. D. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–242, 1976.
9. L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.
10. Douglas G. Fritz and Robert G. Sargent. Hierarchical control flow graph models. In *In*, pages 1347–1355, 1996.
11. Michael Howard, Jon Pincus, and Jeannette Wing. Measuring relative attack surfaces. In *Proceedings of Workshop on Advanced Developments in Software and Systems Security*, 2003.
12. T. Jaeger, R. Sailer, and X. Zhang. Analyzing integrity protection in the SELinux example policy. In *USENIX Security Symposium*, August 2003.
13. Trent Jaeger, Kevin Butler, David H. King, Serge Hallyn, Joy Latten, and Xiaolan Zhang. Leveraging IPsec for mandatory access control across systems. In *Proc. 2nd Intl. Conf. on Security and Privacy in Communication Networks*, August 2006.
14. S. Jha, O. Sheyner, and J. Wing. Two formal analyses of attack graphs. In *Proceedings of the 15th IEEE Workshop on Computer Security Foundations*, pages 49–, Washington, DC, USA, 2002. IEEE Computer Society.
15. Dave King, Susmit Jha, Trent Jaeger, Somesh Jha, and Sanjit A. Seshia. Towards automated security mediation placement. Technical Report NAS-TR-0100-2008, Network and Security Research Center, Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA, November 2008.
16. Dave King, Susmit Jha, Divya Muthukumaran, Trent Jaeger, Somesh Jha, and Sanjit A. Seshia. Automating security mediation placement. In *ESOP*, 2010.
17. J. Morris. New Secmark-based network controls for SELinux. `http://james-morris.livejournal.com/11010.html`.
18. MSDN. Mandatory Integrity Control (Windows). `http://msdn.microsoft.com/`.
19. D Muthukumaran, S Rueda, N Talele, H Vijayakumar, T Jaeger, J Teutsch, and N Edwards. Transforming commodity security policies to enforce Clark-Wilson integrity. In *ACSAC*, 2012.
20. Divya Muthukumaran, Trent Jaeger, and Vinod Ganapathy. Leveraging "choice" to automate authorization hook placement. In *CCS'12: Proceedings of the 19th ACM Conference on Computer and Communications Security*, Raleigh, North Carolina, USA, October 2012. ACM Press.
21. Andrew C. Myers and Barbara Liskov. A decentralized model for information flow control. *ACM Operating Systems Review*, 31(5):129–142, October 1997.

22. Nessus Vulnerability Scanner. `http://www.nessus.org/`.
23. James Noble, Robert Biddle, Ewan Tempero, Alex Potanin, and Dave Clarke. Towards a model of encapsulation. In *2003. Presented at the ECOOP 2003 IWACO Workshop on Aliasing, Confinement, and Ownership. Available at: http://www.mcs.vuw.ac.nz/comp*. Publications, 2003.
24. S. Noel and S. Jajodia. Advanced vulnerability analysis and intrusion detection through predictive attack graphs. In *Critical Issues in C4I, Armed Forces Communications and Electronics Association (AFCEA) Solutions Series*. International Journal of Command and Control, 2009.
25. Steven Noel, Sushil Jajodia, Brian O'Berry, and Michael Jacobs. Efficient minimum-cost network hardening via exploit dependency graphs. In *ACSAC*, 2003.
26. Novell. AppArmor Linux Application Security. `http://www.novell.com/linux/security/apparmor/`.
27. NetLabel - Explicit labeled networking for Linux. `http://www.nsa.gov/selinux`.
28. Security-enhanced linux. `http://www.nsa.gov/selinux`.
29. Xinming Ou, Wayne F. Boyer, and Miles A. McQueen. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 336–345, New York, NY, USA, 2006. ACM.
30. Lee Pike. Post-hoc separation policy analysis with graph algorithms. In *Workshop on Foundations of Computer Security (FCS'09). Affiliated with Logic in Computer Science (LICS)*, August 2009.
31. Beata Sarna-Starosta and Scott D. Stoller. Policy analysis for Security-Enhanced Linux. In *WITS*, April 2004.
32. Oleg Sheyner, Joshua W. Haines, Somesh Jha, Richard Lippmann, and Jeannette M. Wing. Automated generation and analysis of attack graphs. In *IEEE Symposium on Security and Privacy*, pages 273–284, 2002.
33. Sun Microsystems. Trusted Solaris operating environment - a technical overview. `http://www.sun.com`.
34. Nirupama Talele, Jason Teutsch, and Trent Jaeger. Using security policies to automate monitor placement. Technical Report NAS-TR-0163-2012, Network and Security Research Center, Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA, oct 2012.
35. Yongping Tang and Thomas E. Daniels. On the economic placement of monitors in router level network topologies. In *The Workshop on the Economics of Securing the Information Infrastructure*, 2006.
36. Tresys. SETools - Policy analysis tools for SELinux. available at http://oss.tresys.com/projects/setools.
37. R. N. M. Watson. TrustedBSD: Adding trusted operating system features to FreeBSD. In *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference*, pages 15–28, 2001.