SAT-Solving Approaches to Context-Aware Enterprise Network Security Management

John Homer and Xinming Ou
Kansas State University
Manhattan, KS, U.S.A.
{jhomer,xou}@ksu.edu

***Abstract*—Enterprise network security management is a complex task of balancing security and usability, with trade-offs often necessary between the two. Past work has provided ways to identify intricate attack paths due to misconfiguration and vulnerabilities in an enterprise system, but little has been done to address how to correct the security problems within the context of various other requirements such as usability, ease of access, and cost of countermeasures. This paper presents an approach based on Boolean Satisfiability Solving (SAT Solving) that can reason about attacks, usability requirements, cost of actions, *etc.* in a unified, logical framework. Preliminary results show that the approach is both effective and efficient.**

***Index Terms*—Boolean Satisfiability Problem (SAT), Computer Network Management, Computer Network Security, Risk Analysis, Security**

I. Introduction

Enterprise networks continue to grow in both size and complexity and with this increase, concerns for security grow apace. Vulnerabilities are regularly discovered in a wide variety of software applications, so even a network of moderate size can have dozens of possible attack paths, overwhelming a human user with the amount of information. With all of this complexity, it is near impossible for a human to fully and accurately identify which configuration settings should be changed to address security problems.

To make things more complicated, requirements for usability are often at odds with those for security. Configuration management would be a trivial problem if one only needed to consider security requirements; simply shutting down the whole network would resolve any security issues. But configuration changes aimed at correcting security flaws must be made in a context-aware manner, carefully balancing the system's security and usability.

Existing works in enterprise network security analysis, such as MulVAL [19], [20], can identify all possible attack paths in an enterprise system and output them in a graph structure. This structure provides a good foundation for addressing how to automatically find the best way to correct the security problems presented in the analysis results.

We have developed a systematic approach, shown in Figure 1, to aid a human in confronting these difficulties.

This work is supported by the U.S. National Science Foundation under Grant No. CNS-0716665. Any opinions, findings and conclusions or recomendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

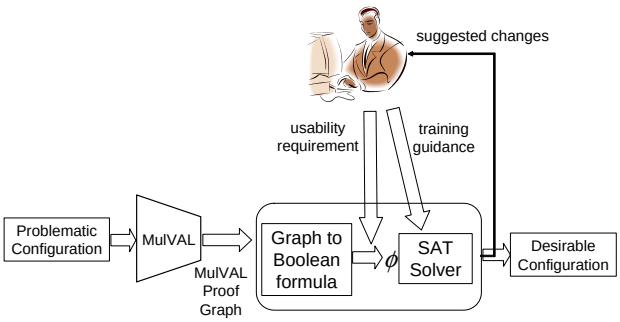

Fig. 1. SAT-based configuration generation

The current (problematic) network configuration settings are passed into the MulVAL toolkit, which produces a logical proof graph identifying all potential attack paths by which an attacker might exploit system resources. This proof graph is converted into a Boolean formula in conjunctive normal form that relates configuration settings and attacker actions with potential effects, such as an attacker being able to execute arbitrary code on a computer in the network. Security and usability requirements, provided by the human user, are also converted into conjunctive normal form and added to the Boolean formula, and this combined formula ϕ is processed by a SAT solver.

A human user can further train the SAT solver as to the relative value of various system resources and usages. Working interactively, the human user is able to quickly identify and resolve network security issues without unknowingly lessening the system usability. As the tool is trained, the degree of automation should increase, producing sound and desirable reconfiguration suggestions with minimal human involvement.

In this approach, we use two SAT solving techniques:

1) MinCostSAT can utilize user-provided discrete cost values, associated with changing a given configuration setting or allowing an attacker a given amount of access, to find a mitigation solution that minimizes the cost in terms of both security risk and usability impairment.
2) By examining the UnSAT core, a minimal set of configurations and policy requirements that conflict, we narrow the complexity of a reconfiguration dilemma to a straightforward choice between options. Past policy decisions by the human user are placed in a partial-

order lattice and used to further reduce the scope of the decisions presented to the user.

By this approach, the human user is not expected to fully comprehend the effects, both good and bad, of all aspects of network configuration, but only to make decisions on the immediate relative value of specific instances of usability and security. In this way, we reduce an extremely complex problem to one of more manageable proportions, automating the verification of both security and usability policies while introducing a method by which conflicts can be quickly and verifiably resolved.

II. MulVAL Security Analyzer

We use the MulVAL tool suite [19], [20] for our work. MulVAL is a security analyzation tool that, given initial network configurations (machines, active services, inter-host reachability, *etc*.) and a database of known vulnerabilities, can identify all potential attack paths by which an attacker can exploit the system. These attack paths are assembled in a logical proof graph, showing how potentially successful attacks into the network are enabled by initial attacks on the outer edges. MulVAL's reasoning engine is specified declaratively in Datalog [1], providing inherent soundness of the results as well as an efficient $O(N^2)$ running time [19].

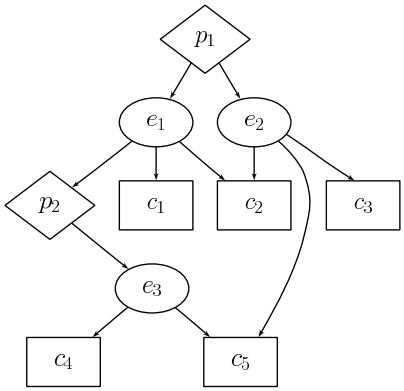

Fig. 2. A MulVAL proof graph

Figure 2 shows part of the proof graph for an example enterprise network we studied. The diamond-shaped nodes in the graph represent privileges an attacker can gain through the exploits depicted as the elliptical nodes. System configuration data are represented by the rectangular nodes, such as c_1, c_2, c_3, c_4, c_5. These can be both administrator-defined configuration settings, like host access permissions, and unintentional facts, such as an existing vulnerability in a specific application. The potential exploits - e_1, e_2, e_3 - link the causality relationship between a privilege that an attacker can gain and the preconditions that make this possible. For example, node e_1 could correspond to a remote buffer-overflow attack on a service. It links the effect of the attack, p_1 (which means the attacker can gain privilege on the victim machine), to pre-conditions for the attack, such as c_1 (which could mean the existence of a buffer-overflow vulnerability in the service program), and p_2 (which could mean the attacker's ability to send a maliciously crafted packet to the vulnerable service). All the arcs coming out of an exploit node like e_1 form a logical AND relation, requiring all of its children to be true before this exploit can be used. The arcs coming out of a privilege node like p_1 form a logical OR relation, in which multiple descendant nodes indicate alternative exploits by which an attacker can gain this privilege.

Although we have chosen to build our implementation based on the MulVAL proof graph, our approach can be based easily on other, similar tools for the production of network attack graphs (or fault propagation models) [6], [7], [11].

III. Reconfiguration Using SAT Solving

Since any network misconfiguration is technically resolvable (if only by removing all inter-machine access), reconfiguration decisions must be made in consideration of the cost of the changes needed and of usability requirements. We have developed two approaches based on advanced SAT solving techniques that can automatically suggest optimal configuration changes to address the security problems presented in a proof graph. Our approaches allow a user to provide feedback to the SAT solver so that constraints on usability, cost of deployment, and potential damage due to successful attacks can all be optimized in a unified framework.

A. Transforming proof graphs to Boolean formulas

We first extract the causality relationships represented in a MulVAL proof graph and express them as a Boolean formula. This is best explained through an example. In the dependency proof graph of Figure 2, the AND node e_1 means that the remote exploit is successful, since all of its children nodes p_2, c_1, c_2 are enabled, and the result of the exploit is that the attacker gains privilege p_1.
This can be expressed by the following formula,

$$p_2 \wedge c_1 \wedge c_2 \Rightarrow p_1$$

Or, equivalently,

$$\neg p_2 \vee \neg c_1 \vee \neg c_2 \vee p_1$$

We similarly convert the other exploit nodes to construct the following formulae:

$$\begin{aligned} e_1 &= \neg p_2 \vee \neg c_1 \vee \neg c_2 \vee p_1 \\ e_2 &= \neg c_2 \vee \neg c_3 \vee \neg c_5 \vee p_1 \\ e_3 &= \neg c_4 \vee \neg c_5 \vee p_2 \end{aligned}$$

Let $\phi = e_1 \wedge e_2 \wedge e_3$, then ϕ is a Boolean formula in conjunctive normal form (CNF) whose size is linear in the size of the proof graph[1]. ϕ encodes all the causality relationships between configuration data and potential attacker privileges shown in the proof graph. For example, if all of c_1, c_2, c_3, c_4, c_5 are assigned the truth value T (as in the current configuration), then p_1, p_2 must be assigned T to make a satisfying assignment for ϕ. Therefore, if one wishes p_1, p_2 to be false (meaning an attacker can gain neither of these privileges), at least some of c_1, c_2, c_3, c_4, c_5 must be assigned

[1]A MulVAL proof graph's size is quadratic in the size of the network (number of hosts) [19].

F, meaning some of the current configuration settings need to be changed. Let $\psi = \phi \wedge \neg p_1 \wedge \neg p_2$; then seeking a satisfying assignment to ψ amounts to finding configuration settings that can prevent an attacker from gaining privileges p_1, p_2.

Every variable representing a configuration setting will be assigned T (meaning that the setting is "enabled") or F ("disabled"). Since every configuration setting is T ("enabled") when the proof graph is constructed, removing or "disabling" that setting will negate the associated variable. For example, if c_1 represents the existence of a software vulnerability on the web server, the negation of that node means patching the vulnerability; if c_5 represents a reachability relationship between the Internet and the VPN server, disabling that node means blocking that access. If we feed ψ to a SAT solver, we can get a satisfying assignment by simply disabling all the configuration nodes c_1, c_2, c_3, c_4, c_5. This is certainly not an optimal solution; we need a secure configuration that maintains basic network usability.

A careful observation of the proof graph shows that by disabling c_5 without altering c_1, c_2, c_3, c_4, we can prevent all the attack paths in the system, but we must consider the effects of this decision. It is not necessarily the case that a minimal number of system changes represents the optimal reconfiguration. Suppose again that c_5 represents accessibility of the VPN server from the Internet. Removing this access would certainly block an attacker, but it would also prevent legitimate users from remotely logging into the network via the VPN server. This type of trade-off between security and usability is often present in system configuration management.

In configuring an enterprise network, we want to compare not only the potential cost in damage from a successful attack, but also the potential losses arising from decreased network usability. If the cost of completely securing the network against attackers is much higher than the potential losses from attacks, it could be a better solution simply to acknowledge and tolerate the possibility that an attacker can obtain some minor privileges on the enterprise system. In this example, we may decide that an optimal solution would not force p_2 to be false, so we can redefine our goal to be $\psi = \phi \wedge \neg p_1$.

We must now re-examine the proof graph in light of this new ψ. Suppose that c_1 and c_3 represent vulnerabilities present in system applications. By patching these two vulnerabilities, we can disable these two nodes and thus eliminate all attack paths that could enable an attacker to gain privilege p_1. This configuration would negate p_1 without violating ϕ, so it satisfies ψ.

Though it is relatively easy to examine and reconfigure this small example, a reliable and automated approach is needed to address security concerns in real-size enterprise networks. We now introduce two applications of SAT solving to resolve network misconfigurations by balancing costs and potential damage.

B. MinCostSAT

MinCostSAT is a SAT problem which minimizes the cost of the satisfying assignment [9]. Mathematically, given a Boolean formula ψ with n variables $x_1, x_2, \ldots, x_n$, each with cost $c_i \geq 0$, find a truth-value assignment $X \in \{0,1\}^n$ such that X satisfies ψ and minimizes

$$C = \sum_{i=1}^{n} c_i x_i$$

where $x_i \in \{0,1\}$ and $1 \leq i \leq n$.

MinCostSAT has been thoroughly studied by the SAT solving community [2], [5], [9], [14]. Although the problem is NP-hard, modern SAT solvers have been very successful in practice, being able to handle Boolean formulas with millions of variables and clauses in seconds. We use the MinCostChaff solver [5] which is a MinCostSAT solver based on the zChaff SAT solver [13].

The MinCostSAT problem minimizes the cost for variables that are assigned T. This matches the semantics for privilege variables, whose T assignment means an attacker can gain some privilege and thereby cause some damage. But for configuration variables, the cost would be incurred when it is disabled, or assigned F. To model this correctly, we first transform our formula to use the negation of a Boolean variable to represent each configuration node. This way, when the variable is assigned T, it means that the corresponding configuration node is disabled, which will incur some cost. For the example we used, the new formula derived from the attack graph will be $\tilde{\phi} = \widetilde{e_1} \wedge \widetilde{e_2} \wedge \widetilde{e_3}$, where

$$\begin{aligned} \widetilde{e_1} &= \neg p_2 \vee \widetilde{c_1} \vee \widetilde{c_2} \vee p_1 \\ \widetilde{e_2} &= \widetilde{c_2} \vee \widetilde{c_3} \vee \widetilde{c_5} \vee p_1 \\ \widetilde{e_3} &= \widetilde{c_4} \vee \widetilde{c_5} \vee p_2 \end{aligned}$$

Here $\widetilde{c_1}, \widetilde{c_2}, \widetilde{c_3}, \widetilde{c_4}, \widetilde{c_5}$ are the new Boolean variables such that $\widetilde{c_i} = \neg c_i (1 \leq i \leq n)$. For $\tilde{\psi} = \tilde{\phi} \wedge \neg p_1$, the MinCostSAT solution to $\tilde{\psi}$ would be the desired solution for the enterprise system's reconfiguration, if we have correctly defined the costs for all variables.

With the expressiveness of Boolean formulas and the power of a SAT solver, a system administrator can ask questions like "what is the best way to reconfigure my system if I want to guarantee that the file server will not be compromised?" This can be done by forcing the Boolean variable x that corresponds to the privilege `execCode(fileServer, someUser)` to be false(i.e., conjoining $\neg x$ to the original formula). He can also ask questions like "Can I make the file server secure while allowing the web server to be accessed from the Internet?" We have implemented mechanisms that allow a system administrator to specify those additional constraints for the various queries he would like to conduct. Those constraints can be straightforwardly specified in Datalog and automatically transformed into additional clauses in the Boolean formula to be solved by the MinCostSAT solver. This kind of constraint can also become a part of the configuration policy. For example, a user might decide that the web server must be accessible from the Internet. If the variable representing this configuration setting is forced true in the Boolean formula, MinCostSAT will never return a suggested reconfiguration that requires this access to be removed. Similarly, potential attacker privileges can be forced

to be always false; for example, a user might decide that an attacker should never access the data historian, and so this access could be forced to be false, meaning that MinCostSAT will never allow it to be true. This effect could also be simulated by assigning unrealistically high costs for those variables; however, forcing them to be true or false will ensure that no reconfiguration suggestion will reverse this decision.

C. Scalability

To test the scalability of our approach, we constructed simulated enterprise networks with two different sizes:

I: 100 host machines, evenly divided in 10 subnets
II: 250 host machines, evenly divided in 25 subnets

We also tested using two different cost functions:

A: All clauses were assigned an equal cost. The effect of this cost policy would simply be to minimize the number of configuration changes made plus the number of compromised machines.
B: Clauses representing code-execution privileges on a machine were assigned costs based on the machine's position in the network. The effect of this cost policy would be to have increasingly high costs for penetrations deeper into the network. The costs for blocking network access to hosts or disabling network services were significant. All other changes had equal, low cost.

The test was conducted on a Linux machine with Opteron Dual-Core 2214 2.2 GHz CPU, with 16GB memory, and running Gentoo Linux with kernel version 2.6.18-hardened-r6.

Sz	Cfn	# variables	# clauses	time (sec)
I	A	11,853	12,053	0.11
I	B	11,853	12,053	0.21
II	A	70,803	72,553	3.03
II	B	70,803	72,553	6.49

The simulated networks on which we performed the above tests were certainly not representative of realistic enterprise network settings, but the performance indicates that modern SAT solvers are likely to be powerful enough to handle the configuration management problem we describe in this paper. Also, a highly correlated network configuration may produce nontrivial runtimes. A full-scope understanding of the scalability of this approach will require extensive real-world testing, currently planned for future work.

D. Iterative UNSAT Core Elimination

We now introduce the second SAT solving technique, in which the concept of UNSAT core is leveraged for the identification and resolution of conflicts in the network policies.

Definition 1. An unsatisfiable core is a subset of the original CNF clauses that is unsatisfiable in itself [4].

When a SAT solver finds a set of clauses to be unsatisfiable, a byproduct of this decision is the UNSAT core. Logically, given an unsatisfiable Boolean formula ψ in CNF, the UNSAT core $\mu = u_1, u_2, \ldots, u_m$ is a subset of all the clauses in ψ (shorthanded $\mu \subseteq \psi$ hereafter) such that ψ will remain unsatisfiable while μ remains unchanged. We generate the UNSAT core using the zChaff SAT solver's zcore function [13].

In this approach we will not rely on cost assignments, but rather on the balance between security and usability policies. Returning to the example from section II, let security policy $\delta = \neg p_1$; then our security policy specifies that an attacker should not be able to gain privilege p_1. Let usability policies $\gamma_1 = c_1 \wedge c_2 \wedge c_4 \wedge c_5$ and $\gamma_2 = c_2 \wedge c_3 \wedge c_5$; then our usability policies together specify that all current configuration settings are necessary to maintain basic network usefulness. So $\psi = \phi \wedge \delta \wedge \gamma_1 \wedge \gamma_2$.

An unsatisfiable formula arises when one policy demands that a certain fact be true while another policy demands that it be false. Let the UNSAT core be $\mu = \phi_u \wedge \delta_u \wedge \gamma_u$, where $\phi_u \subseteq \phi$, $\delta_u \subseteq \delta$, and $\gamma_u \subseteq \gamma$. Each UNSAT core, then, will have some collection of derivation clauses based on MulVAL logic rules as well as security and usability policies specified by the user. Together, these constitute an unsatisfiable instance.

Obviously, the user cannot change the logical foundations of the MulVAl derivation rules, so the μ cannot be resolved by altering any of ϕ_u. To make the necessary reconfiguration decision, we request from the user an immediate decision of the relative importance among the elements of δ_u and γ_u. In this example, the user would be prompted to choose the more desirable of δ and γ_1. Let us suppose that the user decides that the security of the system is more important, and chooses to relax the γ_1 constraint.

Relaxing γ_1 (removing it from ψ) does not necessarily mean that its preconditions will all be disabled; instead, only the configuration variables that conflict with δ will be disabled. The other configuration variables will remain true (unchanged). It is possible, of course, that multiple UNSAT cores exist in a single Boolean formula. In our approach, we iteratively present each UNSAT core, prompting the user to decide for each core which policy constraint can be relaxed. In this way, a satisfiable configuration solution will eventually be reached. Note that if multiple UNSAT cores exist, the final, satisfiable configuration can be affected by the order of presentation of the UNSAT cores.

In this example, the new $\psi = \phi \wedge \delta \wedge \gamma_2$ is again submitted to the SAT solver to check for a satisfiable solution, and again we find an UNSAT core. In this μ, we find a conflict between δ and γ_2, and the user is again prompted to decide which of the two may be relaxed. Suppose that the user again decides in favor of strong security, and chooses to relax γ_2. So $\psi = \phi \wedge \delta$ and we can now find a satisfiable solution.

Utilizing the UNSAT core in this way precludes the need to assign costs to each network configuration setting beforehand, as is required for the MinCostSAT solution. So long as security and usability policies do not conflict, the user is not asked to decide between any two policies or attempt to assign discrete values to them. These decisions are only faced when an actual conflict has arisen, so the human user makes only necessary choices about system resource valuations.

Partial-order lattice: To further reduce the breadth of decisions faced by a human user, we have implemented a partial-order lattice to store the relative priorities between

pairs of policies. Each time the human user is presented with the causes of an unsatisfiable conflict and selects one or more of those constraints to be relaxed, this decision is recorded in the partial-order lattice to be used as a reference for deciding future conflicts. We assume that the constraints that the user allows to be relaxed have a lower overall priority than any clauses that were not relaxed, and this ordering is recorded in the lattice. In future decisions where two conflicting constraints appear for which an ordering is already known, the constraint with higher priority will not be offered to the user as a possibility for relaxation. In this way, conflicts are reduced to comparisons between configuration settings or policy requirements for which relative priorities are not known. Once known, these decisions need not be faced again.

IV. Experiments

Figure 3 shows an example enterprise network that is based on a real (and much bigger) system. We are not able to disclose the real system due to the sensitivity of the information. However, the specific problems we encountered are the same for both the real and the adapted system.

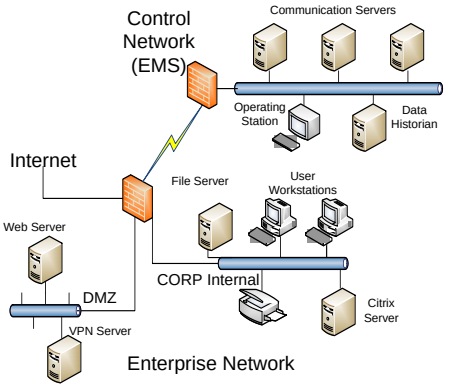

Fig. 3. An example enterprise network

Subnets: There are three subnets: a DMZ (Demilitarized Zone), an internal subnet (CORP Internal), and an Energy Management System (EMS) subnet, which is a control-system network for power grids.

Hosts: The functionalities of most of the hosts are self-explanatory. One functionality of the EMS network is to gather real-time statistics, such as voltage, load, *etc.*, from the power generation and transmission facilities. This information, stored on the data historian, triggers various control signals through the communication servers to maintain proper operation of the power grid.

Accessibility: Both the web server and the VPN server are directly accessible from the Internet. The web server can access the file server through the NFS file-sharing protocol; the VPN server is allowed access to all hosts in the internal subnet. From the internal subnet, only the Citrix server is allowed access to the EMS subnet, and then only to the data historian.

Threats: An attacker with privileges on the EMS machines could potentially take control of physical infrastructures and drive them to a failed state, such as causing the turbine of a power generator to spin at a high speed until self-destruction.

Vulnerabilities: All the machines in the system may have software vulnerabilities in their various services and software applications. The users of the enterprise network may not be careful to protect their log-in credentials and might leak their user name and password to an attacker through various means, including social engineering. The data historian and the communication servers run a proprietary communication protocol that can be easily sabotaged.

A. Application of MinCostSAT approach

We began with a simple configuration policy that assigned an equal cost to all changes in basic settings and to all potential attacker privileges. Based on this policy, running MinCostSAT produced recommendations to remove access from the Internet to the web server and VPN server as well as removing network services from the Citrix server, VPN server, and workstation, essentially cutting off all outside connections to the network. We then started refining the cost functions in awareness of realistic requirements on usability. In our testing, it took only three to four iterative steps to produce a realistic suggestion. We began by assigning high costs to changes in desired network access permissions and network services, as well as varying costs for allowing an attacker access to machines in various subnets, with highest costs in the control network subnet, lower in the internal subnet, and still lower in the DMZ. In this way, the policy asserted that there were higher potential costs in allowing an attacker access further into the system. After several more iterations of reassessing and reassigning costs, the suggested changes are to patch the existing vulnerability in webServer and either remove one employee's account on the VPN server or ensure that the employee's log-in information will not be compromised. Acting on these suggestions, an administrator could patch the vulnerability and spend some time ensuring that the employee understands good security procedures, such as using strong passwords. Based on the costs we used in our policy, MinCostSAT has determined that an attacker might be allowed minimal access to the webServer or vpnServer but will be prevented from accessing any machine in the internal and EMS subnets.

The policy thus produced can be saved for re-use in the future. We added to our example a second web server and file server, with a remote exploitation vulnerability in the newly-added web server. Running MulVAL and MinCostSAT again, using the same cost policy, immediately showed that this vulnerability must be patched. In this way, it took much less effort to produce useful results once a baseline policy has been well established.

It is important to note that assigning different costs can easily produce different suggestions. For example, if the cost assigned to patching the vulnerability in the web server was sufficiently high, the suggested solution might be to change the accessibility from the web server into the internal subnet

and/or to remove the file sharing relationship between the Citrix server and the file server. The cost for changing a specific configuration parameter and the cost caused by a potential attacker privilege will vary from one organization to another. There is no one-size-fits-all cost function suitable for all enterprise systems, and the user of the tool will have to define costs based on local requirements and policies.

B. Application of UNSAT Core approach

In applying the UNSAT core approach to this example, we began with an empty partial-order lattice. Our security policies stated that an attacker should not gain privilege to execute code on any host in the system, and for our usability policies, we required that each configuration setting remain unchanged.

Not surprisingly, we immediately encountered an UNSAT core. We were prompted to choose from among several constraints, some of which must be relaxed[2]:

1) Attacker must not be able to execute code on the web server
2) Allow access from Internet to web server via HTTP, port 80
3) The service httpd is running on the web server
4) A vulnerability exists in httpd on the web server

Suppose that we look into the vulnerability on the web server and find that no patch currently exists for it. We decide that the benefits of the web server outweigh the potential damage of an attack at this level in the network, so we relax the security constraint stating that an attacker should not gain privileges on the web server.

With this updated policy, we run the SAT solver and again encounter an UNSAT core. In this instance, we are asked to select from among the following constraints:

1) Attacker must not be able to execute code on the VPN server
2) Allow access from the Internet to the VPN server
3) The VPN service must be running the VPN server
4) User ordinaryEmployee is classified as "incompetent"
5) User ordinaryEmployee has an account on the VPN server

This decision is slightly harder, since an attack through the VPN server could cause more damage. We decide that the VPN server must remain accessible to employees, so we cannot impair its usefulness by blocking access from the Internet or removing the VPN service. User ordinaryEmployee is classified as "incompetent," meaning that he should not be expected to use strong, unique passwords and that his login data could easily be compromised. We decide, then, that his account on the VPN server should be disabled until such time as he can be trusted to follow strong security guidelines.

Running the SAT solver with the updated policy constraints, we now find a satisfiable configuration for the system. By adjusting our security policy (in acknowledging that an attacker might gain some privileges on the web server) and usability policy (by disabling ordinaryEmployee's VPN account) we can identify a system configuration that adheres to the remaining policy constraints.

[2]We give here the natural language meanings of the Boolean formulas presented by the application. For example, the first policy is the meaning of a constraint wherein the privilege to execute code on the web server is negated.

V. Discussion

MinCostSAT requires cost functions that assign numeric values to every configuration setting and security or usability policy; these cost valuations may be difficult to assign fairly to all resources. Once decided, however, a minimum-cost configuration can be determined at any time, simply by comparing the relative costs of potential security breaches to usability requirements. The biggest challenge with the MinCostSAT approach is determining the basis for the cost functions. Although any metric can be utilized, we suggest an approach wherein a uniform monetary amount is assigned, indicating, for example, the potential liability if an attacker gains access to a specific server or the cost of applying a fix to a known vulnerability.

The UNSAT core for conflict resolution requires no up-front cost assignments, relying instead on immediate decisions made only when a conflict is discovered. We believe that introducing a form of machine learning may assist the reasoning engine in ordering priorities of clauses that have not been previously directly compared, to further reduce the decisions a user faces. The nature of the UNSAT core, however, necessitates that a human user carefully examine all conflicts and make decisions as needed for all occurring conflicts.

Both approaches to configuration resolution, MinCostSAT and the UNSAT core, carry advantages and disadvantages. The optimal approach may be a combination of the two. As these approaches are refined, we believe that many of the low-level configuration settings will no longer be examined or weighed by the human user, who will deal only with policy requirements. A more automated reasoning engine can compare security and usability policies, identify conflicts, prompt for human prioritization between the two, and alter the necessary underlying configuration settings needed to resolve this conflict without introducing new conflicts into the system.

VI. Related Work

SAT solving has been used to address general configuration management problems in the ConfigAssure project at Telcordia [16], [17]. The general configuration problem concerns various requirements on service availability, performance, fault-tolerance, and security. The requirements are specified in a Prolog-like language and a Prolog partial evaluator carries the resulting constraints in the form of quantifier-free formula which is subsequently solved by a SAT solver. This technique supports both configuration synthesis and problem diagnosis and resolution in a logical framework. Our work is closely related to ConfigAssure, and we present a Datalog proof analysis technique that can convert a complete proof-graph into a linear-sized Boolean formula for configuration generation and policy specification. Besides, our focus is on attack modeling and how to resolve security threats in the context of usability requirements. It is likely that the two technologies can be integrated into a unified framework to address a complete set of requirements regarding enterprise network configuration management.

Dewri, *et al.* [3] formulates the security hardening problem as a *multi-objective optimization* problem, in which the cost for security hardening and the cost for potential damage caused by successful attacks are two objectives in the multi-objective optimization problem, whose solution is searched for by a genetic algorithm (GA). We adopt a different approach, MinCostSAT, to find a *provably* minimum-cost solution to the configuration problem. GA, on the other hand, cannot always guarantee to converge to the global optimum. MinCostSAT is a specific optimization problem that has been studied extensively and thus is likely to outperform a general optimization algorithm such as GA for the specific problem. Adopting the SAT solving approach also allows the user to make the various queries we discussed in this paper and resolve potential conflicts in policy specification, in addition to finding the optimal solution for reconfiguration. The benefit of multi-objective optimization is that a user can be presented multiple optimal trade-offs between the two objectives. It will be interesting to study whether the MinCostSAT techniques can be extended to provide multiple trade-off solutions for the optimization problem.

Wang *et al.* [22] developed a graph search-based method for network hardening, with previous work by Noel *et al.* [18]. Their approach is different from ours. An attacker's goal is expressed as a propositional formula on the initial conditions through recursive substitution during graph traversal. The formula is then converted to disjunctive normal form (DNF) which will give all possible hardening options, from which an optimal one is chosen. The converted DNF could be exponential in the size of the attack graph, as admitted in the paper. In our approach, we formulate the optimization problem as a MinCostSAT problem on a Boolean formula whose size is *linear* in the size of the proof graph, and applies a well developed modern SAT solver to find the solution. This is likely to yield better performance since SAT solvers have been very successful in efficiently finding solutions to large Boolean formulas arising from practical problems.

A number of other works addressed the problem of how to use attack graphs to better manage the security of enterprise networks [8], [10], [12], [15], [21], [23]. The observations and insights from these previous works helped us develop the approach in this paper, and our work either complements or improves upon them. Our contribution is the development of formal, logic-based approaches where proof graphs from security analysis are used to compute reconfiguration suggestions automatically, taking into account not only security requirements, but also requirements on usability and trade-offs between costs of security hardening, costs of possible damage due to successful attacks, and costs of loss in usability.

VII. Conclusion

We have introduced a methodology where the system security requirements can be converted to a Boolean formula and, using SAT solving techniques, one can quickly correct misconfigurations that may lead to multi-step, multi-host attacks in enterprise networks. This approach can account for both security and usability requirements, through the adoption of modern SAT solving techniques such as MinCostSAT and UNSAT core elimination. We presented a unified framework in which the competing requirements can be specified in a Boolean formula and an optimal solution can be searched for that provides a reasonable trade-off between the various requirements for practical security administration. Preliminary experimental results on both realistic and synthesized enterprise network settings indicate that the SAT solving approach is effective and scalable.

VIII. Acknowledgement

We would like to thank Zhaohui Fu and Sharad Malik for providing us the zChaff and MinCostChaff SAT solvers for our research. Raj Rajagopalan gave valuable comments on an earlier draft of this article. We would also like to thank the anonymous reviewers for the helpful comments.

References

[1] Stefano Ceri, Georg Gottlob, and Letizia Tanca. What you always wanted to know about Datalog (and never dared to ask). *IEEE Transactions Knowledge and Data Engineering*, 1(1):146–166, 1989.

[2] Olivier Coudert. On solving covering problems. In *33rd Design Automation Conference (DAC'96)*, pages 197–202, 1996.

[3] Rinku Dewri, Nayot Poolsappasit, Indrajit Ray, and Darrell Whitley. Optimal security hardening using multi-objective optimization on attack tree models of networks. In *14th ACM Conference on Computer and Communications Security (CCS)*, 2007.

[4] Zhaohui Fu and Sharad Malik. On solving the partial MAX-SAT problem. In Armin Biere and Carla P. Gomes, editors, *Proceedings of Theory and Applications of Satisfiability Testing - SAT 2006*, pages 252–265, 2006.

[5] Zhaohui Fu and Sharad Malik. Solving the Minimum-Cost Satisfiability problem using SAT based branch and bound search. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, San Jose, CA, USA, 2006.

[6] Kyle Ingols, Richard Lippmann, and Keith Piwowarski. Practical attack graph generation for network defense. In *22nd Annual Computer Security Applications Conference (ACSAC)*, Miami Beach, Florida, December 2006.

[7] Sushil Jajodia, Steven Noel, and Brian O'Berry. Topological analysis of network attack vulnerability. In V. Kumar, J. Srivastava, and A. Lazarevic, editors, *Managing Cyber Threats: Issues, Approaches and Challanges*, chapter 5. Kluwer Academic Publisher, 2003.

[8] Somesh Jha, Oleg Sheyner, and Jeannette M. Wing. Two formal analyses of attack graphs. In *Proceedings of the 15th IEEE Computer Security Foundations Workshop*, pages 49–63, Nova Scotia, Canada, June 2002.

[9] Xiao Yu Li. *Optimization Algorithms for the Minimum-Cost Satisfiability Problem*. PhD thesis, North Carolina State University, Raleigh, North Carolina, 2004.

[10] Richard Lippmann, Kyle Ingols, Chris Scott, Keith Piwowarski, Kendra Kratkiewicz, Mike Artz, and Robert Cunningham. Validating and restoring defense in depth using attack graphs. In *Military Communications Conference (MILCOM)*, Washington, DC, U.S.A., October 2006.

[11] Richard Lippmann and Kyle W. Ingols. An annotated review of past papers on attack graphs. Technical report, MIT Lincoln Laboratory, March 2005.

[12] Richard P. Lippmann, Kyle W. Ingols, Chris Scott, Keith Piwowarski, Kendra Kratkiewicz, Michael Artz, and Robert Cunningham. Evaluating and strengthening enterprise network security using attack graphs. Technical Report ESC-TR-2005-064, MIT Lincoln Laboratory, October 2005.

[13] Yogesh S. Mahajan, Zhaohui Fu, and Sharad Malik. Zchaff2004: An efficient SAT solver. In *Lecture Notes in Computer Science SAT 2004 Special Volume*, pages 360–375. LNCS 3542, 2004.

[14] Vasco M. Manquinho and Jo ao P. Marques-Silva. Search pruning techniques in SAT-based branch-and-bound algorithmsfor the binate covering problem. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21:505–516, 2002.

[15] Vaibhav Mehta, Constantinos Bartzis, Haifeng Zhu, Edmund Clarke, and Jeannette Wing. Ranking attack graphs. In *Proceedings of Recent Advances in Intrusion Detection (RAID)*, September 2006.

[16] Sanjai Narain. Network configuration management via model finding. In *Proceedings of the 19th conference on Large Installation System Administration Conference (LISA)*, 2005.

[17] Sanjai Narain, Gary Levin, Vikram Kaul, and Sharad Malik. Declarative infrastructure configuration synthesis and debugging. *Journal of Network Systems and Management, Special Issue on Security Configuration*, 2008.

[18] Steven Noel, Sushil Jajodia, Brian O'Berry, and Michael Jacobs. Efficient minimum-cost network hardening via exploit dependency graphs. In *19th Annual Computer Security Applications Conference (ACSAC)*, December 2003.

[19] Xinming Ou, Wayne F. Boyer, and Miles A. McQueen. A scalable approach to attack graph generation. In *13th ACM Conference on Computer and Communications Security (CCS)*, pages 336–345, 2006.

[20] Xinming Ou, Sudhakar Govindavajhala, and Andrew W. Appel. MulVAL: A logic-based network security analyzer. In *14th USENIX Security Symposium*, 2005.

[21] Reginald Sawilla and Xinming Ou. Identifying critical attack assets in dependency attack graphs. In *13th European Symposium on Research in Computer Security (ESORICS)*, October 2008.

[22] Lingyu Wang, Steven Noel, and Sushil Jajodia. Minimum-cost network hardening using attack graphs. *Computer Communications*, 29:3812–3824, November 2006.

[23] Lingyu Wang, Anoop Singhal, and Sushil Jajodia. Measuring network security using attack graphs. In *Third Workshop on Quality of Protection (QoP)*, 2007.