Lockdown: Distributed Policy Analysis and Enforcement within the Enterprise Network

Andrew Blaich, Qi Liao, Aaron Striegel, and Douglas Thain
Department of Computer Science and Engineering
University of Notre Dame
Email: {ablaich, qliao, striegel, dthain} @cse.nd.edu

***Abstract*— Security policy is a multi-faceted problem that needs to be enforced at different levels of a system. Attempts to resolve the problem of how to enhance network connectivity with information for better security based decision making while keeping an emphasis on simple deployment and management have been insufficient. Existing solutions take the approach of hammering network-centric mechanisms towards the policy. These solutions are not simple to deploy or manage, sometimes requiring reconfiguration of the network resources or a completely new structure. Some of the solutions in the form of intrusion detection systems are passive, only notifying of an intrusion after the fact. Our project, Lockdown, looks to simplify the deployment and management issues of enhancing network connectivity information while improving upon intrusion detection by providing an active intrusion prevention system.**

I. Introduction

IT is often said that the most secure computer is one that has no connection to the network. Unfortunately, access to the Internet is an essential component of day to day operations for the workforce, especially in today's fast-paced highly connected world. Traditional methods for maintaining system security (firewalls, intrusion detection, etc.) face continued pressure from both benign and malicious entities that seek to circumvent the prescribed policy for the network.

Take for instance the humble firewall such as iptables under Linux policing traffic based on foreign/local port/address tuples. While the intention of a rule to allow outbound port 80 may be to only allow web traffic, the coarseness of the mechanism can be exploited to introduce protocol-abiding but undesirable behavior (ex. Gnutella over HTTP). Conversely, consider legitimate but rarer cases, ssh to a remote site on port 2020 or telnet to port 9005 to check a service. While approaches such as proxy-wise inspection of packets or extremely sophisticated firewalls can attempt to close these gaps, the approaches are a band aid for symptoms of the larger enterprise control problem.

Our work in development, Lockdown, seeks to bring application and user-awareness into network traffic from a top-down perspective rather than hammering network-centric mechanisms towards the policy. We view Lockdown as enhancing network connectivity with information for better decision making while keeping an emphasis on simple deployment and management. To illustrate one small but non-trivial example, rather than silently discarding a packet leaving the application to timeout, the actual network call would return with a security failure.

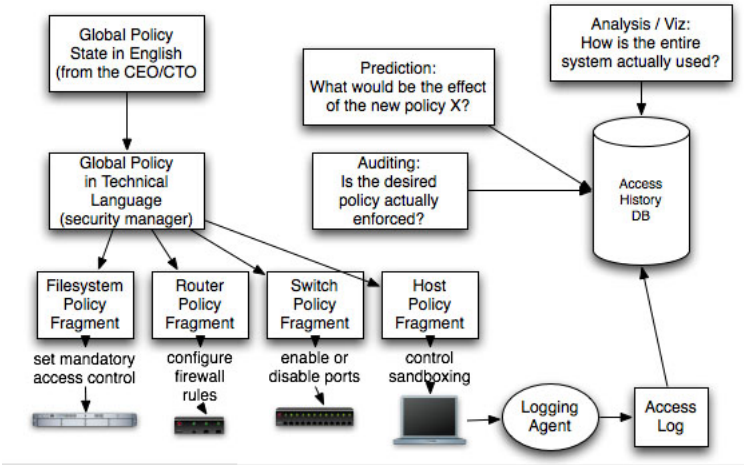

Fig. 1. Lockdown System Architecture

In some sense, Lockdown can be viewed as a practical version of SANE [2] for today's Internet, trading assurance gained from the clean slate approach for deployment capacity with extensive auditing and analysis components.

Lockdown wants to do away with the notion of traditional firewall rules, by adding an additional layer of security on top of them to stop or allow such applications that try to bypass firewall restrictions by following the exact firewall rule and not the intended spirit of it. The end goal is to have a policy in place on the hosts running within an enterprise network that would allow rules written in English to be fully supported on the host. An example of such a rule would be as follows: Floyd is allowed to access the company intranet server using only Internet Explorer from machines installed in room 302, and may not copy data to removable media. From the example rule above the Lockdown policy for network connectivity contains the notions of user id, group id, application used, foreign/local port/IP address, network protocol, and everything in between. With all of this additional information the Lockdown policy is much more robust and capable in preventing/allowing network connectivity and is easier to deploy and manage that pre-existing systems.

The rest of the paper is outlined as follows: Section V discusses the related work in this area. Section II discusses the different parts of the system. Section III presents some of the open ended research questions the designers have come across. Section IV provides an update on the current status of

Lockdown and the remaining work. Section VI concludes this proposal.

II. System Components

The Lockdown system consists of five different parts: data collection, data analysis, data visualization, policy translation (English to computer usable), and finally policy enforcement. See Figure 1 for Lockdown's overall system architecture. Currently work has been focused on the data collection/analysis with recent progress being made on the visualization and policy enforcement.

A. Data Collection

The data collection is done via an agent script[1] that is installed on approximately 200 hosts within our department. The agent gathers at regular intervals all network connectivity taking place on that particular host: the user(s) involved, the files they are touching, and the processes responsible. This data is then sent out from each host to a central server where a collector sits, sifting through the data and logging it into a SQL database. The collector keeps track of when hosts last checked in and presents the details on an internal web-page for the system administrators to monitor.

The agent running on the linux hosts is a bash script using the common tools netstat, lsof, and ps to gather data. In addition when the agent starts up it sends out the linux kernel version, version numbers for each tool, /etc/passwd, /etc/group, cpuinfo, meminfo, uptime, and iptables in a startup file. The agent is currently distributed via our system-admins up-to-date script. After each iteration of the tool there is a series of awk, sed, and diff commands to formulate and size down the data to reduce the amount of network bandwidth used when uploading the data.[2] For example, with netstat, a diff is performed comparing the data collected from the previous netstat command with the current data just collected. With this information we are able to determine if a connection is new and needs to be added to the data file to be sent to the server, or if it has finished and needs to be sent to the server as well. If the same connection appears in the new and old temp files, diff will not add redundant connections to the file. Data is uploaded roughly every 15 minutes.

The agent has evolved over its development to include several safe-guards and checks. Since our campus and the hosts are using AFS[3] the agent has immediate access to certain ğloballyävailable files on the network. For example, at a configurable interval the agent script checks a signal file stored in one of the admin's AFS spaces. The file is readable, but not writable, and contains a value that when set will cause all hosts checking it to sleep and then recheck the value when the sleep is finished until the value in the signal file returns to zero, in which case everything is green and the agent can continue collecting. There has also been built into the agent a way to, disable any combination of the three tools monitoring a specific host in case that the tool becomes unstable. [4]

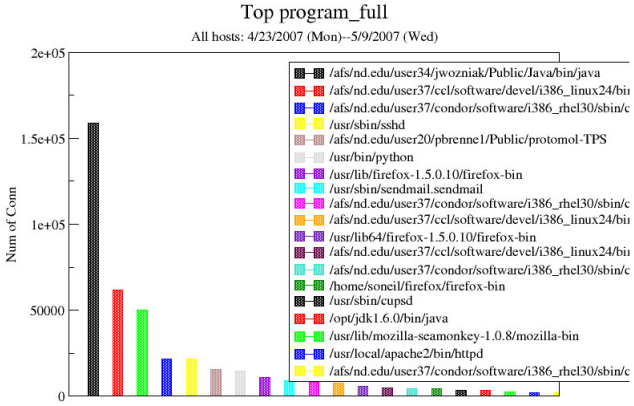

Fig. 2. Top applications by number of connections across all monitored hosts.

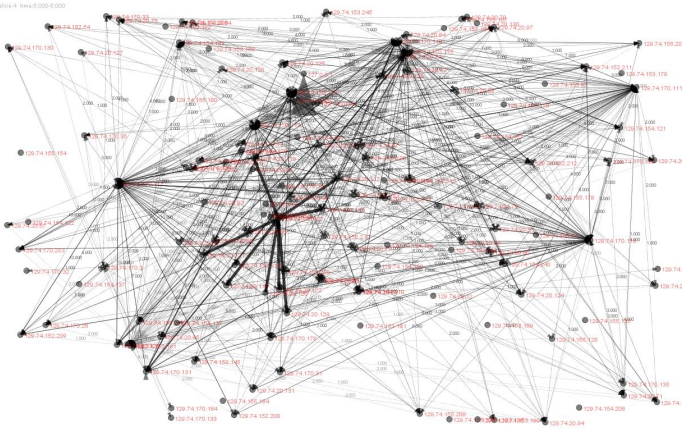

Fig. 3. Network Visualization

B. Data Analysis and Visualization

The data that has been collected and inserted into the database can be mined through using traditional datamining techniques. The results are displayed visually in the form of graphs showing the number of connections per host, the top users per host, the top processes per host/user, etc. Almost any combination of the data can be pulled out and displayed on the system's web-page. In essence given a graph that simply shows the number of connections per host we can drill down to a level of more detail revealing the users and applications involved. Overtime this creates a great snapshot of what is going on within the network. The current results that our internal website displays for us include Node Status (the start time and last check in time in the database), Distinct Fields (Fields other than the unique hosts listed in Bipartite Matching 1, i.e. local/foreign port numbers, uid/gid, application names, etc.), Bipartite Matching 1 (list of unique local / foreign addresses), Bipartite Matching 2 (Local vs

[1]The agent script is currently a linux only bash script. Future work is focusing on creating a Windows service to gather data from a local cluster of Windows hosts on campus.

[2]The agent in performance testing consumes very little resources, roughly 2-5percent of CPU usage.

[3]Andrew File System

[4]A cluster of machines had been running fine for upwards of 40 days until lsof began to cause issues.

```
# Allow connections to the corp www via iexplore.
ALLOW * : iexplore : * : * -> www : apache : www : 80

# Allow Condor-owned programs to use the network.
ALLOW condor : condor_* : * : * -> condor : condor_* : * *

# Allow ssh from anywhere into an sshd.
ALLOW * : *: *: * -> root : sshd : * : 21

# But, no other root owned programs can accept conn.
DENY * : *: *: * -> root : * : * : *
```

Fig. 4. Sample Global Policy Language

Foreign Identity for the connections established on each host), Graphical Visualization of Data 1 (average file sizes, top hosts / users / applications, etc), and Graphical Visualization of Data 2 (Instantaneous number of connections, etc). Figure 2 shows the number of connections across all hosts for the top 19 applications.

Through our data-mining we are also able to infer a certain amount of host-level chaining and trust inference. This is currently displayed in the form of a directed graph, created with the Social Network Image Animator. SONIA allows us to play back snapshots of the system over time to see how connections form and are shutdown between all of the hosts within our web of monitoring. Figure 3 shows an example of the directed graph we have produced for all the hosts being monitored.

C. Policy Language and Distribution

Figure 4 demonstrates a sample global policy that could be enforced with Lockdown with the English translation above the actual description required for the computer to interpret the rules. The symbol * means to allow or deny all.

D. Policy Enforcement

Policy Enforcement is handled via the Linux Security Module framework, standard in the 2.6 kernel, but available as a patch for the 2.4 version. The LSM framework has several hooks placed within an assortment of system calls. See linux/security.h for a complete listing of the hooks. The Lockdown LSM module can be inserted dynamically at anytime, without the need for a kernel re-compile, and is responsible for enforcing the policy that is pushed out onto each of the hosts. The primary focus of our LSM is with the socket hooks, since we can determine whether certain sockets should be created before they are, or if an incoming connection on a listening host should even be established before it does, then either pass or fail the connection. If it is passed, returning a zero, then the normal firewall rules can apply; however, if it is failed, then that attempt has been prevented.

LSMs provide a terrific amount of auditing in addition to policy enforcement without the need to alter the actual linux kernel in anyway. Via the Lockdown LSM a decision can be made to allow or deny a socket_create call with as much information as possible to make an informed decision on the validity of the call.

III. Open Research Questions

There are still some open research questions in regard to policy enforcement that need to be answered. There are still some open research questions in regard to policy enforcement that need to be answered. Applications such as firefox that allow users to load plug-ins and extensions would appear undetectable to an LSM. As the web-browser migrates toward the most commonly used application for users and with web-applications gaining dominance, it is likely to think that everything may migrate over to a browser in the form of web-apps or even modules and/or extensions of the browser. From a user standpoint this is nice, but from a security viewpoint it is tricky to deal with since even the extensions and/or modules run within the browser process as that process.

- Plug-ins: Can these be identified successfully at the OS level? In applications such as firefox can it be detected that a certain plug-in is being used?
- Scripts: Is the appropriate enforcement point on the script or on the individual executables? Does the combination of a set of executables need policy?
- Java/Interpreted Languages: What about applications written in Java or Python or Perl?
- Network ACL: Is there a need for the concept of an ACL for network activity? Is that network ACL directional (in/out), is it dependent on the application, does it change with application arguments?
- Security to application obscurity: To what sense should the configurations of the host and/or network be kept obscure to the application? Is it a black box to probe and timeout leaving distributed systems debugging in its current state akin to something of a black art? Conversely, if information is exposed, how verbose should it be and should it be dependent who or what sent the query?

IV. Current Status

Lockdown is currently in development, but portions of it are deployed within our department's computing clusters. The data collection/analysis which had been the early focus still remains the most complete portion of the system. The agent script is constantly reporting data to the database and our web-server is feeding us up-to-date results on what is going on based on the most recently parsed data-set. The visualization portion of Lockdown, which is simply producing static images currently, will in the future be producing interactive movies allowing the administrators to roll-back over time and observ network connection behavior during any point of the monitoring period. The policy enforcement LSM is in the middle stages of development. As the appropriate hooks are being determined, logging is taking place to determine what information we can collect from the LSM's point of view to enforce our global policy. Finally, the policy distribution component needs to be developed so as to allow simple and efficient distribution of the global rules to all of the hosts running within the intranet.

V. Related Work

In a broad sense, the work in this paper touches on the vast array of research already being conducted with regards

to firewall/policy analysis and intrusion detection. The MAC portacl module in FreeBSD is used to limit binding to ports and in essence is close to what Lockdown can accomplish with the LSM enforcement module. A project out of the University of Italy [10] is attempting a similar distributed approach, but based theirs on the 2.4 linux kernel and in addition they allow for an anomaly detection engine to be plugged in to help establish the policy, resulting in passive detection. Whereas in Lockdown the policy is established beforehand and the analysis is used to determine the rule(s).

Lockdown relies on the LSM framework [12] [11] to do the policy enforcement because of the speedup resulting from working in kernel space rather than user-space. LSM work is also good in that it does not require writing a module to do system call interception which can be costly in terms of CPU time as well as buggy in terms of stability, relying on the programmer to implement each system call properly [6]. There is also LOMAC [7] , which was recently wrapped into a Linux Security Module. LOMAC works on the basis of jailing application that try to access resources they shouldn't be, but is more geared for monitoring memory rather than network connectivity.

There exist several commercial solutions produced by Elemental Security, Cisco, and Endforce to name a few. The commercial solutions, which may be robust, remain costly in terms of deployment and management.

The closest related academic work, SANE [2], grew out of the Clean Slate program. The idea for SANE is nice, but it is impractical to have an enterprise re-do their entire network infrastructure and have everything relying on a centralized server to handle all routing and access control decisions. By making a trade off between the Clean Slate approach and extensive auditing/analysis, Lockdown is able to achieve a similar result to that of SANE without requiring any restructuring of the network and/or its resources.

Lockdown focuses on auditing to look at the richness of the data, showing a deeper understanding of what is taking place on the network so that the rule designers can make better informed decisions on how to enforce the policy given that rules are not limited by simply the standard four-tuple of the traditional linux iptables.

VI. Conclusion

Using a system such as Lockdown global policy on an enterprise network can be distributed to the hosts to respond to various threat levels. When policy is pushed out onto the hosts the analysis engine can determine if the policy is in fact being followed or if there is a further need to lock down users or applications that might be misbehaving. While there do exist commercial alternatives that solve a similar problem. Lockdown relies on using commonly available tools and features of linux to create and deploy an intrusion prevention system that is easier to maintain and less costly in terms of maintenance fees and/or subscription fees. Where Lockdown is conceived to provide robust security in the form of more accurate firewall rules it also has the advantage of being able to do the trust-inference and host-chaining that exists among a monitored network to give designers and administrators an interactive visual of what is happening over time on the enterprise intranet. Lockdown's beauty is in its simplicity of not having to re-work a network's infrastructure and by using commonly available tools and techniques inherent to linux to enhance network connectivity information allowing for better policy mapping within an enterprise.

Acknowledgment

The authors would like to acknowledge the research undergraduates Greg Allan and Brian Sullivan for their work with the data visualization and web-based query components.

References

[1] E. Al-Shaer and H. Hamed, Discovery of policy anomalies in distributed firewalls, in IEEE INFO- COM, Mar. 2004, pp. 2605 2616.

[2] Martin Casado, Tal Garfinkel, Aditya Akella, Michael J. Freedman, Dan Boneh, Nick McKeown, Scott Shenker. SANE: A Protection Architecture for Enterprise Networks. 15th USENIX Security Symposium. Vancouver, Canada, July, 2006.

[3] Suresh N. Chari and Pau-Chen Cheng. BlueBoX : A Policy-Drive. Host-Based Intrusion Detection System

[4] Crispin Cowan, Steve Beattie, Greg Kroath-Hartman, Calton Pu, Perry Wagle, and Virgil Gligor. SubDomain: Parsimonious Server Security. LISA 2000

[5] Thomas E. Daniels and Eugene H. Spaffor (CERIAS Purdue University). A Network Audit System for Host-based Intrusion Detection (NASHID) in Linux. CERIAS Tech Report.

[6] Tal Garfinkel. Traps and Pitfalls: Practical Problems in System Call Interposition Based Security Tools. Network and Distributed Systems Security 2003

[7] Jinhong K. Guo, Stephen Johnson, David Braun, and Il-Pyung Park (Panasonic Information and Networking Tech Lab). Applicability of Low Water-Mark Mandatory Access Control Security in Linux-Based Advanced Networked Consumer Electronics . Consumer Communication and Network Conference 2004

[8] Jeffrey Horton and Rei Safavi-Naini. Detecting policy violations through traffic analysis. ACSAC 2006

[9] Moody, James, Daniel A. McFarland, and Skye Bender-deMoll. 2005. Visualizing Network Dynamics. American Journal of Sociology, (January 2005). http://www.stanford.edu/group/sonia/ .

[10] Pier Lica Montessor and Davide Pierattoni. Towards effective Intrusion Prevention: developing an open framework for integrated network security . (Tech Report)

[11] Markus Quaritsch and Thomas Winkler. Linux Security Modules Enhacnements: Module Stacking Framework and TCP State Transition Hooks for State-Driven NIDS. Secure Information and Communication 2004

[12] Chris Wright, Crispin Cowan, James Morris, Stephen Smalley, and Greg Kroah-Hartman. Linux Security Module Framework. Ottawa Linux Symposium 2002