TRACING ATTACKS AND RESTORING INTEGRITY WITH LASCAR

Alexandre Aellig, Philippe Oechslin
LASEC EPFL

Abstract: We present a novel method to trace the propagation of intrusions or malicious code in networked systems. Our solution is aimed at large numbers of loosely managed workstations typical of a research environment as found in CERN. The system tags events which have a potential to become harmful. On a given machine all processes that results from the tagged event are marked with the same tag and the tag is carried on to others machines if a tagged process establishes a connection. Tag creation logs are stored in a central database. When an intrusion is detected at a later time, all machines and processes that may have lost their integrity due to this incident can easily be found. This leads to a quick and effective restoration of the system. Our implementation of the system is designed to incur very little overhead on the machines and integrates easily with many flavors of the Linux operating system on any type of hardware.

Key words: Recovery, Intrusion Propagation, Intrusion Detection

1. INTRODUCTION

Undesired intrusions are always a problem but intrusions in large groups of workstations are particularly serious. This is especially true if the workstations are managed by their users or many groups rather than by a single central entity. An intrusion showing up at a given workstation may be the result of the compromise of various intermediate machines. All implied machines may not show the same symptoms with some compromised machines not showing any symptoms at all. Cleaning up all machines that show symptoms is thus not enough. Malicious code may still be lingering deep in a machine and strike again when all other machines have been restored. Restoring all systems is not an option when there are too many machines or too many managers.

2. THE LARGE SCALE ATTACK RECORDER (LASCAR)

The large scale attack recorder provides a mean to find out exactly which machines have been hit by a given attack. LASCAR is divided into three parts, the recorder, the database and the analysis tool. The recorder consists in a module loaded on every hosts of the workstation cluster. This module tracks all the events in each workstation and logs the needed information to the central integrity database. Two types of events are logged by the recorder: process events and network events. The analysis tool parses all the events in the integrity database and reconstructs all propagation paths within and between the workstations. Once a potential integrity breach has been detected, the analysis tool is able to trace back the origin and the evolution of the breach in the cluster protected by LASCAR. In particular, it can determine which are the corrupted machines in the cluster and visualize the path and the propagation of the attack.

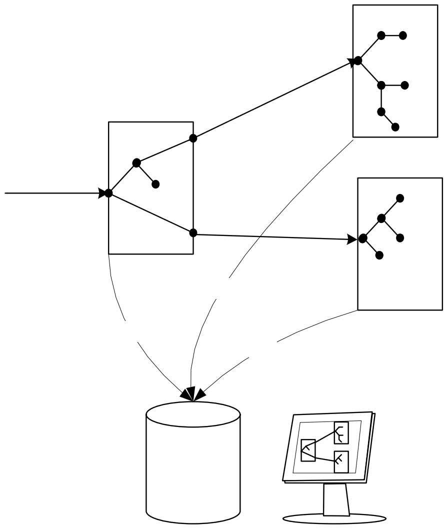

Figure 1. LASCAR at work

2.1 Existing Work

Our approach of recording the propagation of suspicious activities across processes and workstation is quite unique. Previous work that is related to our approach includes:

- The EMERALD system [2] is an advanced intrusion detection system aimed at large scale networks. It is composed of classical IDS systems distributed on all elements of the network and of an intelligent method to merge and correlate all information gathered. It is a heavy system in the sense that complex statistics have to be gathered in many places of the network and it suffers from typical shortcoming of IDS, namely that they generate false alarms and that they are not able to detect new classes of attacks that were not imagined by their conceivers. Our contribution is much more lightweight and humble, in the sense that we do not actually detect attacks but we gather evidence for later damage assessment and integrity restoration when some other system or person has detected an attack.
- For database security it has been proposed (e.g. in [1]) to tag data in order to indicate whether it is correct, damaged to some extend or even unsafe to use. The tags propagate when datasets are combined in calculations. The advantage of the method is similar to our case: In case of loss of integrity only the damaged data needs to be reconstructed rather than the whole database.
- In [3] the propagation of intrusions across multiple workstations is formally stuidied. The results of the paper could allow us to find out the intrinsic properties of an intrusion by matching our data to the mathematical models.

2.2 Tagging inside a workstation

Inside every machines of the LASCAR cluster, the recorder monitors network and processes activities. Using the Linux Kernel modules (LKM) ability to intercept system calls, it can log every incoming or outgoing connection on the machine and every new listening socket or session id (SID) change.

When the machine receives an incoming connection, LASCAR checks whether this connection is considered as trusted or not. In the latter case, the recorder logs the event and tag the process handling this connection with a “suspicious” flag. This flag indicates that the corresponding process is launched by an untrusted connection and thus can be potentially dangerous for the machine integrity. All child processes on the machine will automatically inherit this flag. In order to limit the volume of information logged, the recorder does not log every new process that inherits a “suspicious” flag but only processes that are created within a new session. A new session is typically created when a user launches a process in the background or when multiple interactive sessions are run in different windows. Lastly, the recorder logs all outgoing connections which are established by flagged processes.

A suspicious activity (and a potential loss of integrity) can thus be traced on the machine from the point where it entered the system up to all points where connections were made to other machines.

2.3 Tagging on the network

The previous method can be generalized to the entire LASCAR cluster. However in this case, the "suspicious” flag must now be remotely transmitted amongst machines of the LASCAR cluster.

The simplest way of propagating the flag is to set a bit in the IP header of the packets transmitted by a flagged process. This could for example be one of the Type Of Service (TOS) bits which are not usually used within a local network. (An alternative would be the “evil bit” suggested in [4] published on april fools day this year). The best way of using such a bit would be to use the default value of the bit for the flagged processes and to use a non-default value for all other processes. Thus, packets providing from machines not running the LASCAR recorder would automatically appear as suspicious. Since connections from outside the cluster are considered suspicious anyway and administrator privileges are required to set IP header bits, only machines within the cluster where the administrator account has been compromised could avoid logging by resetting the bit. A more secure approach would be to use a set of bits to carry the result of a cryptographic calculation. For example, the calculation of the IP packet ID field (16 bits) could include a secret key owned by the LASCAR recorder. Finally, to prevent manipulation of the flag in transit or creation of spoofed packets, the IPsec protocol could be used to authenticate all packets in a cryptographically secure way.

In summary, an incoming connection will be logged by the recorder only if it comes from a machine outside the LASCAR cluster or if its IP header indicate that the connection was generated by a process previously marked as “suspicious”. All connections internal to the cluster and originating from clean processes won't be logged by LASCAR.

3. EXPERIMENTAL RESULTS

To illustrate the behavior of LASCAR we have chosen two scenarios that have been recorded by our implementation of LASCAR: a normal user activity and a computer worm.

3.1 Normal user activity

Figure 2 shows an output of LASCAR analyzer generated from the integrity database. It shows the connection of a user to the LASCAR cluster composed by lasecpc12, lasecpc15 and lasecpc16. The user launches an xterm window and from that window he connects to two other machines, maybe to launch two calculations. In the original session he starts the lynx browser to access a web site on the Internet (198.133.219.25)

Lasecpc12 receives the incoming ssh connection from an outside host at the IP address 128.178.73.68. It logs the connection attempt and its two subsequent sessions: xterm and lynx. We see that each session then connects to others machines.

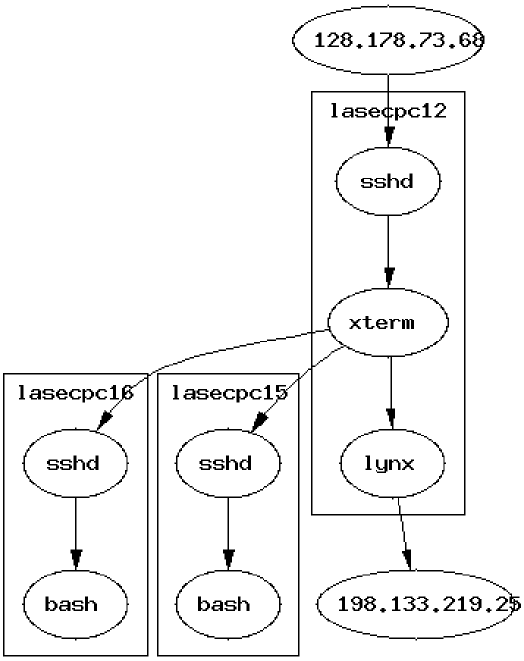

Figure 2. Propagation graph of a normal user behavior generated automatically by LASCAR

Lasecpc15 and lasecpc16 consider the incoming ssh connections as untrusted, even though they come from a machine inside the cluster. Indeed, connections were launched by sessions that were carrying the “suspicious” flag.

3.2 Computer worm behavior

The second example illustrates the evolution of a computer worm inside a cluster where machines are exposed to a common vulnerability of the secure shell program (SSH) on port 22. In this case, the worm enters through an external connection and corrupts a first machine of the cluster. It then tries to initiate each time two outgoing connections to replicate itself To simulate a work behavior we have manually connected a host by ssh and recursively opened two new sessions to random hosts from every ssh session.

With the help of LASCAR, we are able to trace back the evolution of the worm replication inside the cluster. In particular, we see that machines lasecpc15 and lasecpc16 are infected twice. When the attack has been detected, the LASCAR analyzer provides an exhaustive description of the attack path and scope with the id of the machines involved. This allows cluster administrators to quarantine only the corrupted machines of the cluster. In particular, the analysis of the propagation path may identify machines that have been infected but are not showing any symptoms of infection.

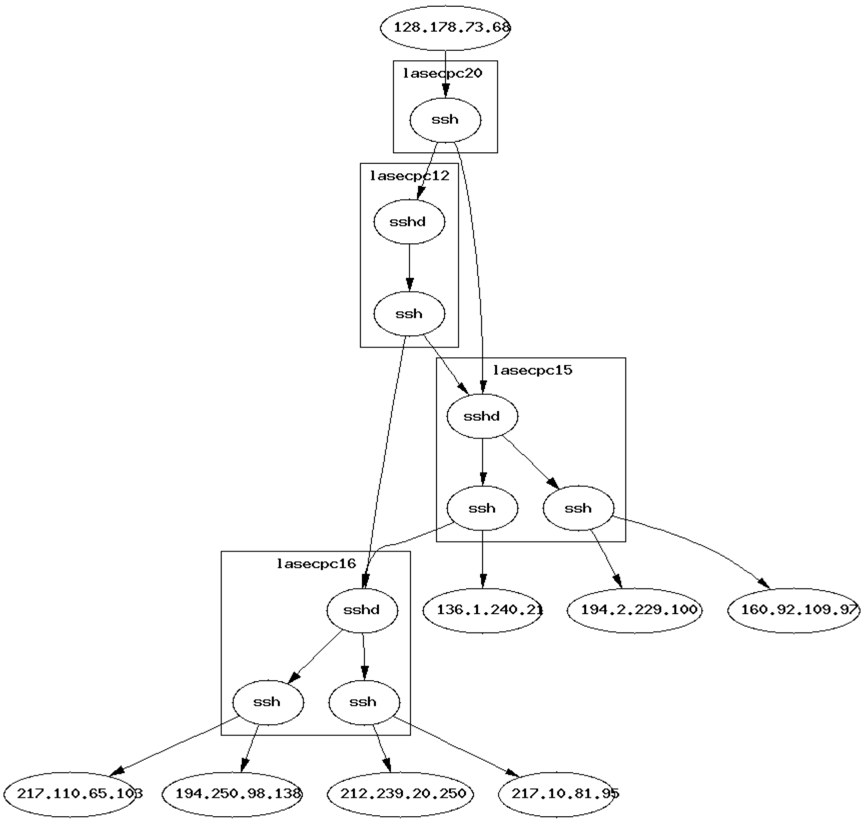

Figure 3. Propagation graph of a simulated computer worm plotted by LASCAR

4. IMPLEMENTATION DETAILS

LASCAR has been implemented on Linux using Linux Kernel Modules (LKM). These modules allow administrators to load and unload features on the fly inside the operating system kernel. LKM was the ideal solution for LASCAR since it combines the power and speed of being directly integrated into the kernel together with the flexibility of an independent program. Similar solutions are available on others UNIX systems.

The first implementation of the recorder was made specifically for the CERN network as a connection logger. After further research it has evolved to the actual LASCAR. Two functionalities were needed in order to implement the recorder: interception and modification of the connections and tagging of processes.

When a network event occurs or a new session is created, the corresponding system call is intercepted and LASCAR output is generated accordingly. The output can be customized, but it contains at least minimal security information such as connection IP addresses, ports numbers and processes info (session id, process id, user id) for the LASCAR analyzer. In the case of a process, we set a "suspicious” flag on the process called by the incoming connection. Every child process then automatically inherits this flag.

All the logs generated by the recorder are forwarded by the logging daemon to a remote integrity database, which centralize the logs of the LASCAR cluster needed by the analyzer to generate graphs of the LASCAR reports as shown in section 3. The analyzer currently uses timestamps to link the events between different machines (e.g. an outgoing connection and the matching incoming connection on another machine), since the concept of flagging trough TOS bits has not yet been implemented. Additionally, it has the ability, as a forensic tool, to regenerate graphs of a whole attack based on several criteria or to perform some basic intrusion detection using a blacklist.

5. DISCUSSION: LASCAR AND INTRUSION DETECTION

LASCAR is not an intrusion detection tool by itself. Its main function is to record traces of potential intrusions. Still, it can provide basic intrusion detection capabilities.

5.1 The suspicion criteria or identifying the potential troublemakers:

LASCAR only tags potentially harmful connections and processes. To do so it must have some criteria to identify potential harmfulness. In our case the criteria simply is the fact that a connection origins in a machine outside the cluster. This reflects the belief that the malicious activity that we want to trace enters the cluster from the internet, either because an attacker is targeting the cluster or because the attack propagates automatically and arbitrarily hits the cluster. If desired, the tagging criteria could be extended to include any process that acquires root privileges. More complex criteria could be made available by running a complete intrusion detection system on each workstation and using its output. The important point about our approach is that there is no need for such a complicated way of finding potentially harmful connections or processes. Indeed, we could even try to tag and log all connections and processes. The use of the suspicion criteria only serves to relieve the load on workstations and the main database.

5.2 The detection criteria or finding out when you have been hit:

Tagging and logging is only one part of LASCAR. The other part is the analysis of the gathered information in order to identify the machines that have lost integrity. To start an analysis we first need to know that we have been attacked. The most evident way of detecting an attack is when its first symptoms become visible (e.g. data loss, deterioration of service). Of course it would be more useful to detect an attack when it first appears. This is the goal of all intrusion detection systems. Alas, there is no way yet to build an IDS which will catch intrusions early and will not create a large number of false alarms. In a large setting like ours the amount of false alarms could be prohibitive. This is why we resort to logging suspicious activity such that when an attack is later confirmed we can go back and quickly get rid of its effects. Still, we have built a very pragmatic IDS capability into LASCAR. A criteria for attack detection used at CERN is a blacklist of IP addresses of servers hosting malicious code. Although not every attack will generate a connection to these addresses, any process that connects to the addresses of the black list must be malicious. Since connections made by potentially harmful processes are logged by LASCAR anyway, attempts to connect to addresses from the black list can be signaled with no overhead.

5.3 Using LASCAR as an IDS tool by itself:

The graphs that can be created from data gathered by LASCAR describe the propagation behavior of potentially harmful code or actions. This data contains metrics which are closely related to malicious activity and which could lead to an IDS system that would not have the high false alarm rates of other known systems. Interesting metrics would be the maximum depth in a propagation tree (how many times a suspicious activity has jumped to a next workstation), the maximum out degree of nodes in the trees (e.g. how many different workstations an activity has jumped to from a single workstation) or simply the number of nodes in a graph (how many machines at all have been hosts to the same activity).

6. CONCLUSIONS

Using a combination of process and connection tagging we have been able to create a system that can record the propagation of an activity through a cluster of machine and trace back all machines and processes that contributed to a loss of integrity. This information makes it possible to completely retrace the propagation of an attack and to eradicate compromised malicious code and compromised systems without having to restore large numbers of unconcerned hosts. Our system is particularly well suited for large networks of loosely managed systems as found in academic or research environments.

REFERENCES

1. P. Ammann, S. Jajodia, C. D. McCollum, and B. T. Blaustein, ``Surviving information warfare attacks on databases," *Proc. IEEE Symp. on Research in Security and Privacy*, Oakland, Calif., May 1997, pages 31-42.
2. Peter G. Neumann and Phillip A. Porras. Experience with EMERALD to date. In *Proceedings of the 1st Workshop on Intrusion Detection and Network Monitoring*. 1999, USENIX, pages 73–80.
3. Sotiris Nikoletseas, Grigorios Prasinos, Paul G. Spirakis, and Christos D. Zaroliagis. Attack propagation in networks. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 67–76, 2001.
4. S. Bellovin, The Security Flag in the IPv4 Header, *RFC 3514*, Internet Engineering Task Force, April 1st 2003