CS4520 Project Proposal

Background

Intrusion Detection Systems (IDSs) are applications that monitor the state of a local (host-based) or remote (network-based) machine and analyse specific activities in order to detect unauthorised intrusions. Charles Pfleeger identifies 8 major tasks of IDSs in Security in Computing (1):

- Monitoring users and system activity
- Auditing system configuration for vulnerabilities and misconfigurations
- Assessing the integrity of critical system and data files
- Recognizing known attack patterns in system activity
- Identifying abnormal activity through statistical analysis
- Managing audit trails and highlighting user violation of policy or normal activity
- Correcting system configuration errors
- Installing and operating traps to record information about intruders

Pfleeger then concedes that "no one IDS performs all of these functions", giving the general description that "an IDS is a sensor, like a smoke detector, that raises an alarm if specific things occur." In contrast to this, an Intrusion Prevention System (IPS) is an application that actively tries to enforce preventative measures to protect against these intrusions. While the purpose of an IPS is simple and intuitive, a recent study by the University of South Wales found that under controlled test conditions, less than 2% of attacks against systems defended with modern IPSs were capable of successfully evading intrusion, while the detection rate of such intrusions was over 95% (2). This highlights an important concern in modern computer security: that while it is often possible to detect the presence of unauthorised intruders, systems that attempt to automate the process of intruder prevention are largely unsuccessful and flawed. Lee Allen enumerates a number of techniques that are used by intruders to avoid detection (3), and amongst the most common of these is the manual cleaning of log files to avoid leaving traces of the intruder's presence, and the modification of configuration files in order to make the system less hostile to their presence.

Research proposition

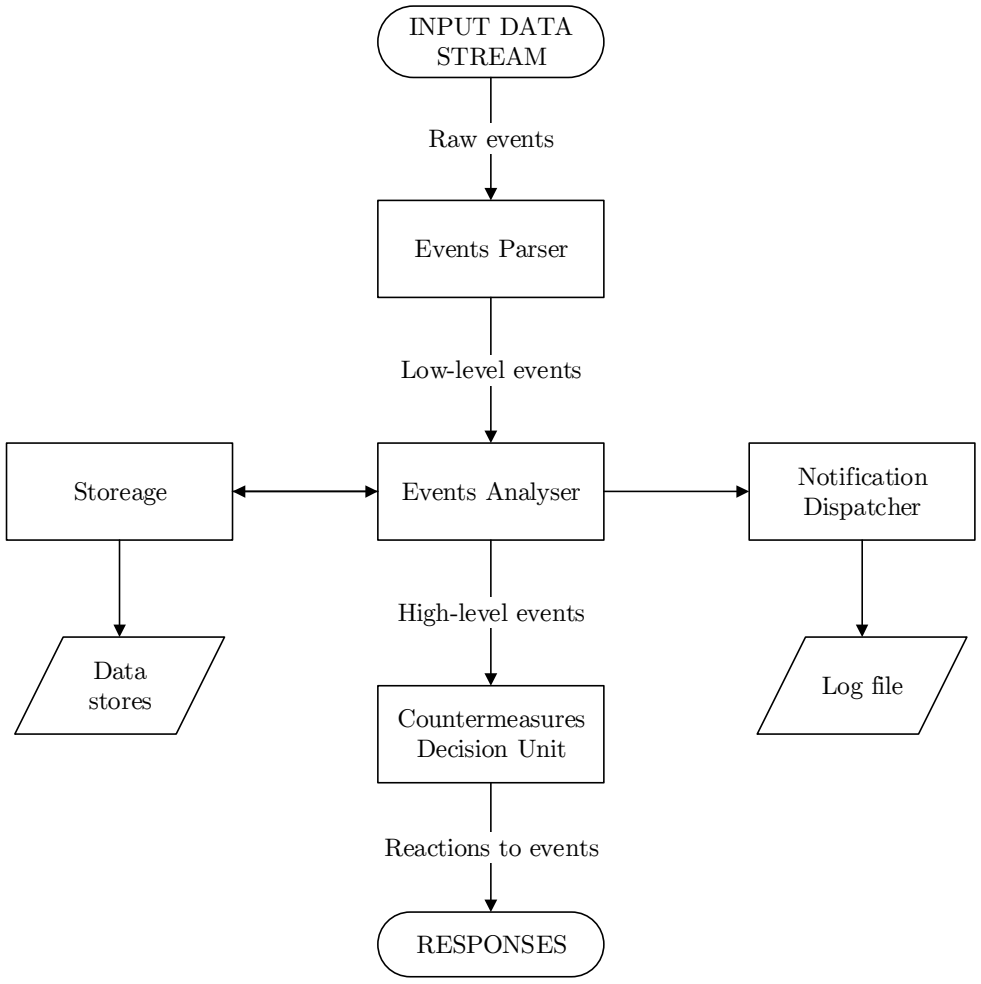
There are a number of IDSs (both commercial and open source) which attempt to detect intrusion through monitoring the state of the network and its traffic for known attack patterns, and the sophistication of these signature based systems means that real-time detection of intrusions is often successful. By contrast, host-based IDSs for internal system states tend to be much less reactive; usually taking the form of non-interactive applications that perform one-time sweeps of the system in order to generate a report. Two of the most popular of these applications are Tripwire (4) and Logwatch (5), which both provide a report of the system state which can be automated so as to be generated at set intervals. Unlike network IDSs, these tools do not provide real-time data, and this can cause a latency between the time of an intrusion and the time at which one of these systems detects it. Additionally, neither system offers any form of active correction mechanism, so it is up to the user to attempt to fix problems that are detected. This leads to an interesting potential research proposition – the case for an IDS which monitors for signs of tampering of local files in real-time. The Linux kernel added an `inotify` subsystem in its 2.6.13 release which provides a mechanism for dispatching notifications when specific files or directories are modified (6), and this could be used to create an event-driven real-time IDS, especially when combined with the UNIX "everything is a file" philosophy, which allows for probing information about currently executing processes through /proc/*.

Research objective

To develop a host-based intrusion detection system daemon for Linux platforms which performs filesystem and processes signature and heuristics-based anomaly detection and recovery in real-time. The table below lists 8 possible functions of the ISD, along with the type of monitoring (signature-based or heuristic-based), and the possible response type (Passive for detection, Active for prevention).

Component	Action	Type	Response
Processes	Monitor the number of active processes	Heuristic	Passive
Processes	Maintain a blacklist of banned processes	Signature	Active
Processes	Monitor the CPU usage	Heuristic	Active
Filesystem	Verify integrity of key system files	Signature	Passive
Filesystem	Track changes to key system files	Signature	Active
Filesystem	Detect tampering of log files	Heuristic	Passive
Filesystem	Monitor the hard disk usage	Heuristic	Passive
Filesystem	Monitor the hard disk activity	Heuristic	Passive

Feedback to the user could be provided through either a web-accessible interface, system log, or by automatically dispatching emails. In the case of active response types, the responses could range from the killing of suspect processes, to restoring tampered files to their original state, and showing the changes made. The program would be implemented as a system daemon, with multiple candidates programming languages being suitable for the purpose, such as C, Python, Perl, Bash, etc. A high-level overview of the system components is included below, with a modular system layout allowing for incremental development:



Resources

Reference	(1) Pfleeger, C. P. and Pfleeger, S. L. **Security in Computing**. Prentice Hall, 2006.
Overview	An overview of computer security attacks and countermeasures.
Points of Interest	Chapter 7.5 (p.484) includes a section on IDSs, giving an overview of common components, purposes, and the Intrusion Detection Framework.

Reference	(2) Xynos, K., Sutherland, I. and Blyth, A. **Effectiveness of blocking evasions in Intrusion Prevention Systems**. Stonesoft. 2013. Online link.
Overview	A study of the effectiveness of various IPS and IDSs in preventing common attacks.
Points of Interest	The report highlights the low success rate of network based IPSs, and emphasises the importance of local IDSs in auditing the damage caused during intrusions.

Reference	(3) Allen, L. **Advanced Penetration Testing for Highly-Secured Environments: The Ultimate Security Guide**. Packt Publishing Ltd., 2012.
Overview	A penetration-testing handbook for exploiting security vulnerabilities in remote systems.
Points of Interest	Multiple sections include instructions for avoiding detection by signature based ISDs, and the necessary steps required to minimise the evidence of intrusions, such as log file cleaning.

Reference	(4) itripn and stephd. **Open Source Tripwire**. SourceForge. 2013. Online link.
Overview	One of the most popular and established host-based Linux ISDs.
Points of Interest	Tripwire uses a fairly common method for detecting file integrity violations (similar to other systems such as `debsums`). A database backend is used to store a set of hashes for a list of files which act as checksums. When the tool is ran a second time, the checksums are re-computed and any discrepancies against the baseline readings are flagged as anomalies.

Reference	(5) bjorn1, kirkbauer and mtremaine. **Logwatch**. SourceForge. 2013. Online link.
Overview	A popular log analysis system.
Points of Interest	Logwatch parses a user-configurable set of system logs using Perl scripts for each type of log, and summarises the events in a report. The idea is that these reports would be checked by the system administrator, and any anomalous entries would be flagged up as possible signs of intrusion.

Reference	(6) Love, R. **inotify - a powerful yet simple file change notification system**. Linux Cross Reference. 2005. Online link.
Overview	Linux kernel documentation for the inotify filesystem subsystem.
Points of Interest	This document provides an overview of the inotify subsystem, and contains usage details.

Reference	(7) Burgiss, H. **Security Quick-Start HOWTO for Linux**. The Linux Documentation Project. 2002. Online link.
Overview	An overview of the basic steps required to secure a Linux installation from intrusion.
Points of Interest	Chapter 6 details intrusion detection techniques, including the use of Intrusion Detection Systems and a list of symptoms and behaviours which can indicate that a system has been compromised. Of particular interest is a list of common system files which are the target of tampering during an intrusion, and ways in which tampering can be detected.