ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

Detecting IT Infrastructure Policy Violations in a LAN Environment Using Triggers

Nickson M. Karie
Dept. of Computer Science, Kabarak University, Private Bag - 20157, Kabarak, Kenya

Abstract
Many organisations today make use of Acceptable Use Policy (AUP) to specify the actions prohibited to the users of an organisations' IT infrastructure. All users are usually required to adhere to all the policies specified in the AUP document without exception. Despite the use of Intrusion Detection Systems (IDSs) to detect any malicious activities, some users still find a way to violate some of the laid down IT infrastructure AUPs. For this reason, this paper presents a framework that can help network administrators detect IT infrastructure AUP violations in a Local Area Network (LAN) environment using triggers. In the context of this paper, triggers are predefined procedural codes that are automatically executed whenever a user tries to violate any of the laid down organisation IT infrastructure AUPs. Once executed, triggers have the ability to log certain activities from the client computer among other security threat events executed during the AUP violation period. The goal of the proposed framework is to assist network administrators in detecting IT infrastructure AUP violations as well as be able to log certain crucial information which can later be used to prove that a particular incidence took place. The framework also allows network administrators to pass commands to the client computers and capture additional information such as screenshots of the clients' activities as well as perform aborting operations. A framework for detecting IT infrastructure AUP violations in a LAN environment using triggers constitutes the main contribution of this paper.

Keywords
Acceptable Use Policy, I.T Infrastructure, AUP Violations, Intrusion Detection Systems, LAN Environment, Triggers, Digital Forensic Evidence

I. Introduction

The majority of organisations use different types of network-based and host-based security software to detect malicious activities, protect systems and data, and support incident-response efforts [1]. However, unlike network-based intrusion detection systems, host-based intrusion detection systems are normally used to analyse data that originates from host computers [2]. The analysis may include, but is not limited to, examining events like which files have been accessed and which applications were executed in the host computer. This is specifically evident when investigating insider attacks or for when regular security checks are conducted. Host-based intrusion detection systems are also useful in that they can keep track of the behaviour of individual users [2].

In an effort towards fighting IT infrastructure policy violations and collect relevant incidence information in a LAN environment, law enforcement agencies have also started incorporating the collection and analysis of digital evidence data into their infrastructures [3]. For this reason, developing tools and techniques that can be part of an IT infrastructure to assist network administrators, in detecting Acceptable Use Policy (AUP) violations is of uttermost importance. Such tools and techniques can further help in gathering relevant evidence data from host computers and other IT infrastructures as well. Moreover, with the rapid changing technology environment [4] it is clear that, more rapid changes in the way security incidences are detected and the way security data is analysed needs to be done.

In this paper, therefore, the concept of triggers is introduced in an attempt to help automatically detect AUP violations in an organization's LAN environment. Besides, triggers have been used in many other different areas of computer science in an attempt to automatically effect changes in the state of devices, databases, electronic circuits, remote detection of events, among other applications. In this paper, however, triggers are programmed in such a way as to automatically log activities from host computers for security analysis as well as for digital forensic purposes. Note that security analysis is used here to refer to analysing the host computer events in a bid to identify security loopholes in a network environment. Besides, the host activities and other digital traces mostly found in computers, among other digital devices, can be useful to courts and law enforcement agencies in identifying valuable evidence for further forensic investigation purposes [5, 6].

As for the remaining part of this paper, section II introduces the background followed by a detailed description of the proposed framework in section III. Section IV presents a case scenario to test the feasibility of the framework followed by a critical evaluation of the framework in section V. Finally, conclusions and future work are provided in section VI.

II. Background

In today's complex technological world, network administrators face many difficult LAN security issues. For this reason, every network administrator needs to understand and be concerned about network security. Knowing that connecting computers to networks significantly increases security risks, the need to monitor the network environment then arises. Network monitoring can be described as the use of a system to constantly monitor a computer network for any network security threats and then notify the network administrator of such incidences [7]. While an intrusion detection system may be used to monitor a network from outside threats, a network monitoring system, on the other hand, monitors the network for problems caused by such events as running application that overloaded or crashed network servers, malicious network activities, network connections, among others [7].

While the traditional ways of using log files data from host computers to analyse host network activities is still valid [9, 10], the evolution in digital technology, however, calls for new, efficient, effective and alternative approaches for detecting AUP violations in network environments. This also includes securely logging of network events data from the host computers. The proposed framework in this paper, thus, offers an alternative way of detecting

IT infrastructure AUP violations in a LAN environment using triggers. As mentioned earlier, triggers are predefined procedural codes that are automatically executed whenever a user tries to violate any of the laid down organisation IT infrastructure AUPs. Triggers, therefore, can help detect events while they are happening or possibly stop an activity before it affects the system. In addition, triggers can also gather information about which process initiated an event, and the user associated with that event, especially when the user identification changes [8].

The data gathered using the proposed framework, therefore, may contain profile information associated with certain host events and can serve as a proof of what happened in a host computer. This is useful especially when log files and registry entries have been erased by perpetrators in a bid to hide their tracks. Besides, the data gathered can further be used for security analysis as well as for digital forensic purposes. The next section elaborates on the design of the proposed framework.

III. The Proposed Framework

Figure 1 shows the design of the proposed framework in this paper. The framework has three layers arranged from top to bottom where the first layer depicts the application layer. The network layer and the database or network storage layer in the second and third layers respectively follow this. Organising the framework into three layers was necessary to simplify the understanding of the framework as well as to present specific finer details of the framework.

Infer from Figure 1 that, running authorized programs or applications in a LAN environment as shown in the application layer do not invoke triggers. The triggers are usually installed in all host machines that are deemed susceptible to possible attacks and are automatically executed whenever a user tries to violate any of the laid down IT infrastructure AUPs. Note also that, the term "host" used throughout this paper refers to an individual computer system, thus a trigger would be needed for every machine. Running any malicious program or application that violates the AUP in a host computer, however, does invoke the triggers and the process of logging starts automatically. The logging process, however, is explained in more details using Figure 2.

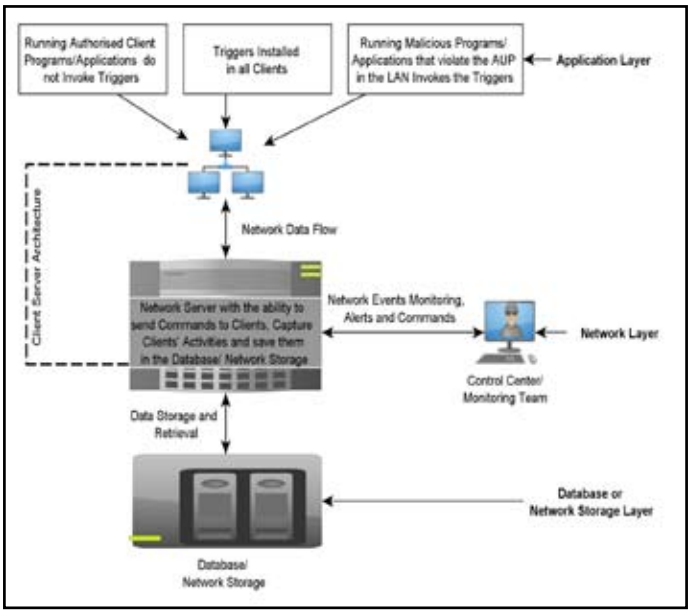

Fig. 1: Framework for Detecting AUP Violations Using Triggers

The network layer shown in Figure 1 is responsible for handling the logging process. This is true because the network data gathered is primarily channelled through the organization's network environment. As mentioned earlier, this process is explained using Figure 2. Figure 2 is also used in this paper to explain the activities of the database or network storage layer as depicted in Figure 1. In addition, Figure 2 also simplifies the understanding of the core logical design and flow diagram of the triggers and their associated processes.

However, before explaining the network layer and the database or network storage layer shown in Figure 1, a practical scenario used to test the feasibility of the proposed framework is considered. This scenario is used in this paper as an example to explain, in simple terms, an incidence that is adopted to show how a violation of the AUP in a LAN environment can be detected.

IV. Case Scenario

Hypothetically speaking, assume that using the notepad program in Windows is a violation of the AUP. A practical scenario was simulated in a lab environment to test the feasibility of the framework, where it is assumed that opening a program like notepad is a violation of the AUP. Furthermore, this simulation was done in a university setting during the writing of online, computerized tests. Having any program open other than, the student's test window was perceived as a violation of AUP. Any student opening the notepad program, therefore, violates the AUP and invokes a trigger, which then sends an AUP violation alert to the monitoring team.

Once the alert has been propagated to the monitoring team, the logging process begins automatically. This is shown as step 4 of Figure 2. Based on the alert received, the monitoring team can further send commands to the client computer to gather more information related to the violated AUP where necessary. Using this scenario we, therefore, explain how the proposed framework shown Figure 1 handles this incident.

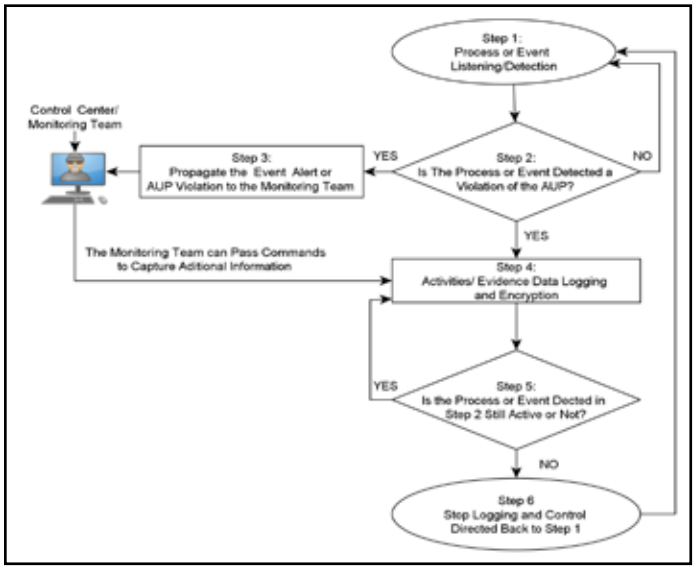

Fig 2: Flow Design of how the Proposed Framework is implemented.

The entire process as depicted in Figure 2 begins when an event considered as a violation of the AUP or security threat event is detected in step 1 of Figure 2. Using the scenario described in section IV, opening the notepad program in windows during the writing of online computerized tests, for example, will be detected

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

as a violation of the AUP and an alert is sent to the monitoring team. Figure 3 below shows a view of how alerts are displayed after being propagated in the network layer and viewed at the security center and/or by the monitoring team.

Note that to detect that using the notepad program in windows during the writing of online-computerized tests is a violation of AUP in step 1 of Figure 2; such an event must first be pre-programed in the triggers. The triggers are also designed in such a way that they have the ability to listen to events being executed by a user on the host computer and determine whether such events are safe or violate the laid down AUP. Steps 2 to 6 as shown in Figure 2, form the activities of the second and the third layers of the framework shown in Figure 1. These steps are explained in more detail in the paragraphs to follow.

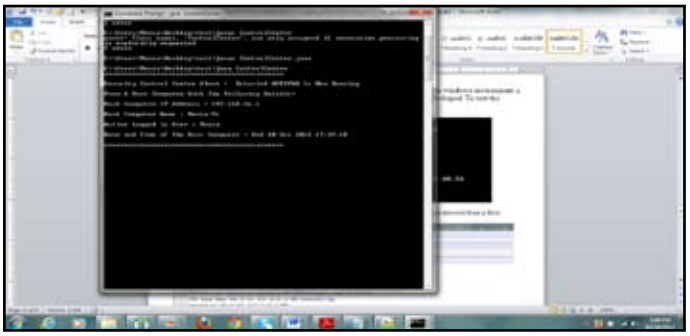

Fig 3: How propagated alerts are displayed at the Security Center for a detected event

Once an event is detected in step 1, step 2 has the ability to check if that event or activity is what has been pre-programmed as a violation of AUP or security threat event to the organization's IT infrastructure. A 'NO' from step 2 is interpreted as not being a security threat event in the network and/or not an AUP violation. The control goes back to step 1 where the listening process continues. At this point no logging of evidence data is done. However, the default legacy log files will still be captured by the host computer as this is by default the work of any operating system.

A 'YES' from step 2 will cause an alert signal to be sent to the security center or monitoring team as shown in Figure 2. This process happens in the network layer of the proposed framework shown in Figure 1. Considering the scenario described earlier in section IV, during an online-computerized tests, having any other program open other than the student's test window is a violation of AUP. Any student opening the notepad program, therefore, will invoke a trigger and send an AUP violation alert to the monitoring team.

Step 3 of Figure 2 is basically meant to propagate the alert so as to notify the monitoring team of the event in the network. Further, step 3 can be used to initiate an investigation process in the computer network in the case of a forensic analysis.

Having known the magnitude of the AUP violation and/or the security threat event through the propagated alert, the security team can then send commands to the client computer to gather more evidence information as shown in Figure 2. Step 4 is where the process of logging and encrypting network events data caused by a violation of the AUP or security threat event automatically starts. Figure 4, hence, shows a sample text file with the details of the evidence information logged after violating the AUP.

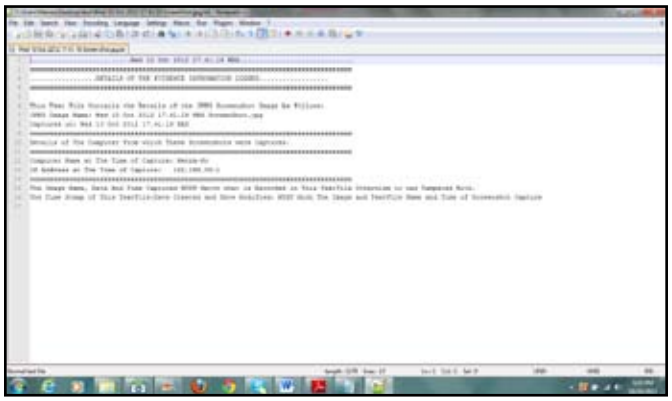

Fig 4: A text files showing the details of the evidence information logged

The data logged can be stored in the host computer, however, a copy of the same data is sent to centralized network storage as shown in database or network storage layer of Figure 1. The stored data is made available for later analysis and/or examination. For integrity checks using the proposed framework, the logged data is usually hashed before being stored in the host computer or sent to the centralized network storage. Figure 5 shows a sample of the details of screenshots and text files captured and stored in the network storage or database. The database or network storage is where the logged data is finally stored for future analysis.

Step 5 checks if the detected event and/or process from Step 2 is still active or not. A 'YES' from step 5 causes the control to be passed back to step 4 where the process of logging and encrypting the evidence data continues in a loop. This looping process was done to ensure that the framework captures as much evidence data as possible within the short time interval while the detected event is still active. However, a 'NO' from step 5 causes the control to automatically stop the logging process at step 6. This is because the detected event will no longer be active hence the logging process stops. The control goes back to step 1 where listening continues and ready to detect the next event that would potentially violate the AUP.

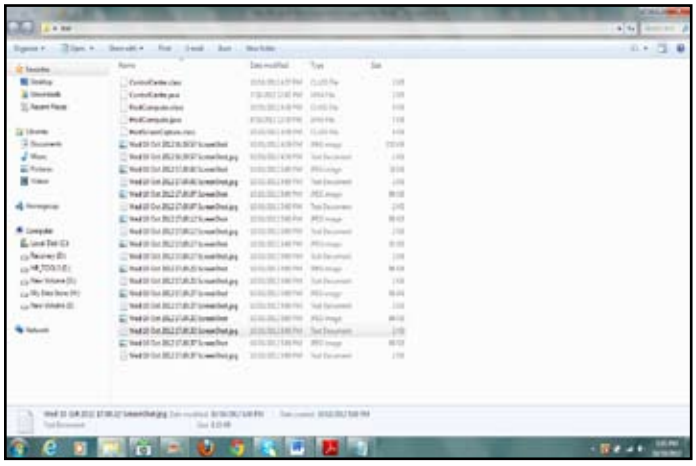

Fig 5: Details of screenshots and text files in the network storage/ database where the logged data is finally stored

The processes shown in Figure 2 are, however, independent in every host. The detection of an event in one host computer does not invoke a trigger in another host computer. This also implies that, every computer is treated independently and do not in any way depend on the activities detected in a neighboring host. In the

case of multiple events detected in the same host, each event is treated as a new thread, hence, handled independently. Termination of one thread (detected event) does not affect another thread that is still active in the host computer, thus logging of evidence data will continue.

As mentioned earlier, the goal is to be able to securely log and retaining network evidence data from host computers even if the legacy log files and registry entries may have been erased. For the purpose of this paper, during a lab simulation to test the feasibility of the proposed framework, only common events (not all events) were pre-programed as triggers, as a proof-of-concept only. Future implementations of the framework, however, will consider implementing real-world organizational IT policies and security threat events, e.g. Denial-of-Service events. This will also include features that assist network security administrators to customize any events and IT infrastructure AUPs they might want to monitor and securely log evidence data. The next section presents a critical evaluation of the proposed framework in this paper

V. Critical Evaluation of The Framework

The goal of the proposed framework in this paper is to help network administrators detect AUP violations in an organisation LAN environment using triggers. Once executed, triggers have the ability to collect data about events taking place on the host system being monitored. However, the framework can also help establish that an event took place, that a crime has been committed in the LAN or can even provide a link between a crime and its victim or a crime and its perpetrator.

The framework can also be used in any organisation LAN environment to implement network-monitoring systems that can automatically detect AUP violations. This can further ensure the confidentiality, integrity and availability of the network infrastructure at all time. This is true because any security threat events will automatically be detected and resolved early enough to avoid any loss or damage to the network infrastructure.

However, institution of higher learning, schools, shopping malls, among other organisations can also benefit from this framework especially for monitoring network activities in their LAN environment.

In the lab simulation of the proposed framework, it was evident that, no too much network traffic is generated because most of the activities are handled at the users' host computer. Moreover, in all the experiments, the authors did not see any irregularities in the performance of the host computer; however future developments will take this into more consideration to see if there can be any effects in the performance of the host computers as a result of using the proposed framework in any LAN environment.

Note also that, for every AUP defined using the proposed framework, there are specific evidence data that may be required. Therefore, the depth of data logged may as well depend on the organizational IT AUP violated or the security threat events detected. This is because the relevance of data collected may not be the same for different pre-defined organizational IT infrastructure AUPs or security threat events. For example, consider the scenario discussed in section IV, the data logged as a result of running a notepad program may be different when compared to a security threat event like Denial-of-Service in the LAN.

Treating the logged evidence data like any other data that is found in the legacy log files of any computer system, it becomes clear that such data can also be used as potential digital evidence in the case of a digital forensic investigation. This is true when the legacy log files are unavailable for use as a result of deletion. For this reason, in the authors' opinion, the logged data using the proposed framework can be significant for IT security analysis as well as for digital forensic purposes.

Another advantage of the proposed framework is that, because the logged data is stored in a different (Network Storage) location and away from the large volumes of legacy log files, investigators can save time. This is true because, investigators can avoid spending too much time examining large volumes of host log files during investigations (unless for exceptional cases). This helps them to concentrate on the logged evidence data for the specified violated AUP or the security threat event detected. Besides saving on time, organizations can also save on resources like money that would have been used for lengthy investigation processes. The next section presents the conclusion of this paper.

VI. Conclusion

In this paper, a framework for detecting IT infrastructure AUP violations in a LAN environment using triggers is proposed. The framework can reduce, for example, investigation time because potential digital evidence data can be gathered separate from large volumes of legacy log files. In addition, unlike the legacy log files additional data like screenshots, can also be captured as part of the logged data.

The lab-simulated experiments to evaluate the proposed framework have shown that the framework is practical and feasible. It is also possible to log certain specific information based on the IT infrastructure AUP violated in a LAN environment. This framework can, thus, be used in different organisations, for example, as a way to implement digital forensic readiness as well as in detecting AUP violations in a LAN environment.

However, more research needs to be done to further extend the proposed framework in this paper so as to be able to cater for wider variety of IT infrastructure AUP violations. This may also include detecting AUP violations in remotely administered networks. As part of the future work, the authors is now developing an enhanced prototype to test the feasibility of this framework in other real world implementation.

References

[1] *Karen K and Murugiah S., Guide to Computer Security Log Management. Recommendations of the National Institute of Standards and Technology. NIST Special Publication 800-92. September 2006.*

[2] *Bace, Rebecca: An Introduction to Intrusion Detection & Assessment. Infidel Inc., prepared for ICSA Inc. Copyright 1998.*

[3] *Digital Evidence and Forensics, National Institute of Justice, Available at: http://www.nij.gov/nij/topics/forensics/evidence/digital/welcome.htm [Accessed July 18, 2012]*

[4] *Anon, Rapid Change in Technology Sparks New Environment for Higher Education. Available at: http://www.globalizationofhighereducation.com/rapid-change-in-technology-sparks-new-environment-for-higher-education, March 24 2014 [Accessed May 11, 2014]*

[5] *E.Casey, Handbook of Computer Crime Investigation: Forensic Tools & Technology, Academic Press, pp.1.16,2002*

[6] *Markus H. et al, Social Snapshots: Digital Forensics for Online Social Networks*

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

[7] *Mittal H., Jain M., Banda L. Monitoring Local Area Network Using Remote Method Invocation. International Journal of Computer Science and Mobile Computing. Vol. 2, No. 5, May 2013, pp.50-55*

[8] *Bace, Rebecca Gurley: Intrusion Detection. Copyright 2000 by Macmillan Technical Publishing, ISBN 1-57870-185-6*

[9] *Kozushko, H., Intrusion Detection: Host-Based and Network-Based Intrusion Detection Systems. September 11, 2003. Available at: http://infohost.nmt.edu/~sfs/Students/HarleyKozushko/Papers/IntrusionDetectionPaper.pdf [Accessed May 11, 2014]*

[10] *Krotoski, M.L., and Passwaters, J., Using Log Record Analysis to Show Internet and Computer Activity in Criminal Cases. NOVEMBER 2011 UNITED STATES ATTORNEYS' BULLETIN.*

Nickson M. Karie is currently a PhD candidate in Computer Science at the University of Pretoria South Africa focusing on Digital Forensics. Mr. Nickson has presented several research papers at different international conferences and published others in world known scientific journals. His research interests include network security, intrusion detection, information security, and digital forensics