Analysis of Security Data from a Large Computing Organization

A. Sharma, Z. Kalbarczyk, J. Barlow, and R. Iyer
University of Illinois at Urbana-Champaign
1308. Main St. Urbana, IL 61801, USA
aashish@lbl.gov, {kalbarcz, jbarlow, rkiyer}@illinois.edu

Abstract — **This paper presents an in-depth study of the forensic data on security incidents that have occurred over a period of 5 years at the National Center for Supercomputing Applications at the University of Illinois. The proposed methodology combines automated analysis of data from security monitors and system logs with human expertise to extract and process relevant data in order to: (i) determine the progression of an attack, (ii) establish incident categories and characterize their severity, (iii) associate alerts with incidents, and (iv) identify incidents missed by the monitoring tools and examine the reasons for the escapes. The analysis conducted provides the basis for incident modeling and design of new techniques for security monitoring.**

Keywords - incident/attack data analysis, security monitoring, alerts, large scale computing systems.

I. Introduction

In this paper, we analyze security incidents that have occurred over a period of 5 years across 5000 machines monitored at the National Center for Supercomputing Applications (NCSA) at the University of Illinois. The monitored systems include i) eight high-performance computational clusters (each consisting of 2k – 12k processors) participating in grid computing and accessed by users world-wide; ii) smaller research clusters; iii) production infrastructure systems, e.g., mail, web, domain name, and authentication servers, certification authority; iv) file servers (which include NFS, AFS, GPFS, and Luster file systems); and v) over 1000 desktops and laptops located in a class B (/16) network. The target system has naturally continued to evolve over the years, and hence the analysis is an aggregate over the measurement period. The usage of the network studied is not different from any other network with heterogeneous set of computing systems.

To illustrate the complexity of the target system, Fig. 1 shows a section of a five-minute snapshot of traffic for systems within NCSA. The light-colored ovals represent the IP addresses, and the lines correspond to network connections. Fig. 2 shows the connections of a system involved in a security incident. The red ovals here represent the IP addresses that were determined to be malicious[1]. The example shows that it is not trivial to navigate in this sea of traffic and data to identify compromised hosts and the information relevant to a given attack.

Our goal is to understand how attacks progress as seen by the monitoring tools and traces recorded in the data logs. Reconstructing the attack steps – from an alert to a compromise – provides the foundation for categorizing the security incidents, identifying missed incidents, and characterizing the incident severity. While analysis of real security data is of value in and of itself, the actual studies are rare (see Section III). This is due to (i) privacy issues in accessing the security data and (ii) difficulty in comprehending and correlating large quantities (terabytes) of multimodal data from various logs (often with different data formats) and alerting tools.

This paper makes an important step in overcoming these challenges. Specifically, this study (i) introduces and illustrates a methodology (combination of automated analysis and human intervention with potential for further automation) to extract and process relevant data from different logs in order to identify the progression of an attack, (ii) associates alerts with incidents, and (iii) identifies incidents missed by the monitoring tools and analyzes the reasons for the escapes.

The key findings of this work are as follows:

- While over half (57%) of incidents are detected by IDS-Bro (31%) and NetFlows (26%) monitors, a significant fraction of incidents (27%) are not detected by any alert but identified by the third party external notification.
- Almost 26% of the incidents analyzed involved credentials stealing. For this incident category, an attacker usually (97% of the time) enters with already-stolen credentials of a legitimate user [20] and hence the behavior is the same as that of a malicious insider[2].
- Association of alerts with incidents reveals that usually one (or at most two) alert(s) is(are) responsible for identifying an incident, and the same alert can be triggered by different attacks.
- Nearly 50% of the incidents are detected in the last phase of an attack, when attackers start misusing the system.
- Anomaly-based detectors are seven times more likely to capture an incident than are signature-based detectors. However the signature-based detectors (due to their specialization) have fewer false positives compared to the anomaly-based detectors.
- Alerts detecting the most incidents are not necessarily the most efficient alerts; e.g., TopN alert detects

[1] A detailed connectivity map also based on a five minute snapshot is given in (www.ncsa.illinois.edu/~jbarlow/graphs/nfdump-5-min-snapshot.gif).

[2] Our earlier study (using a subset of the same data logs) showed that in all but one credential stealing incident, attackers obtained access to the host using a stolen password (78%), a public key (16%), or combination of multiple authentication means (e.g., password + publickey) (6%) [20].

15% of the incidents of low-to-medium severity, but it exhibits a 33% false positive rate.

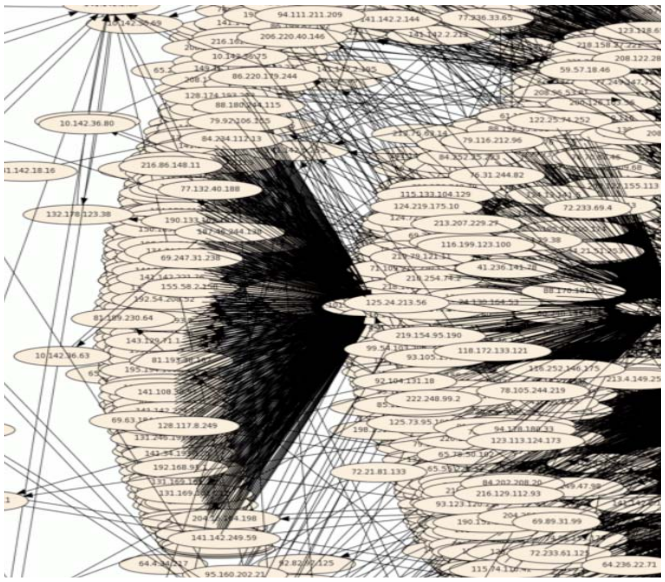
Figure 1: Five-Minute Snapshot of In-and-Out Traffic within NCSA.

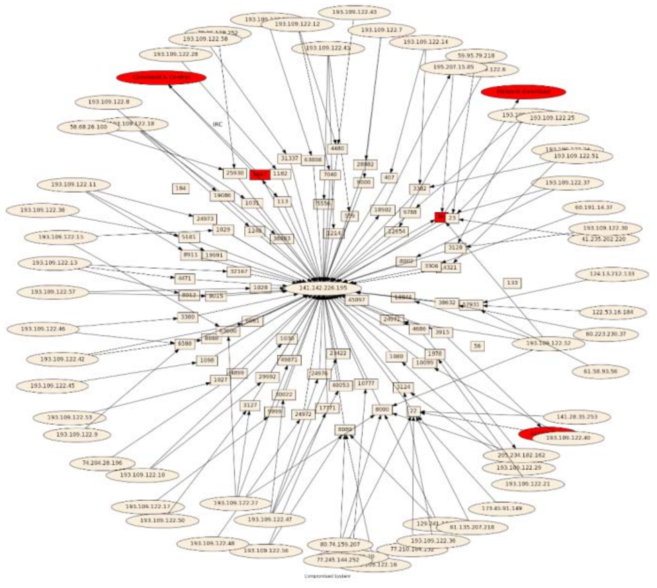
Figure 2: Connections of a system involved in a security incident.

The study, in addition to providing insights into the attacks' evolution and efficiency of alerts over a long period of time, brings an understanding of incidents missed by the alerts. The tangible benefit would be in designing better detection tools. For example, learning the phase of an attack in which the alerts are triggered provides important information to guide design of protection mechanisms. The intangible benefit is to promote research in the area of designing methods that can directly improve network security.

Although this work focuses on a NCSA network, a good portion of the analyzed incidents pertain to other campus or open research networks. In a fully closed environment, e.g., a corporate network with firewalls on perimeter, types of attacks and type of misuses identified may be different. However, a significant fraction of incidents (about 26%) we analyzed are due to compromised credentials when an attacker penetrates the system using the already stolen credentials (e.g., user password) of a legitimate user. Such incidents could happen in virtually any network which is accessible from the Internet (e.g., social networking sites, email systems, corporate networks allowing VPN access) or from an intranet or a business network managed by an IT department within a corporation. Therefore, insights from this work are representative of many different environments and can be used to guide better design and organization of defense mechanisms.

II. Monitoring Tools

NCSA employs a variety of monitoring tools and corresponding detection techniques to provide comprehensive detection coverage [2]. Fig. 3 depicts the monitoring and alert generation architecture used to collect the measurements at both the network and host layers. At the network layer *Bro IDS (www.bro-ids.org)* [16] is used to perform deep packet inspection of network traffic going through the border router for detection of anomalous activity. Additional network traffic analysis is performed using *network flow logs*. A network flow is a 7-tuple record of network transmission data: IP address, ports, protocol, bytes and packet count for both source & destination hosts. The network flow (netflow) collectors are distributed such that flows are monitored and logged from the border router using *Argus* (*www.qosient.com/argus/*) and from the internal routers using *nfdump* (*sourceforge.net/projects/nfdump*). The measurements give visibility to traffic within the internal subnets (nfdump) as well as to the traffic going in and out of the network (nfdump + Argus).

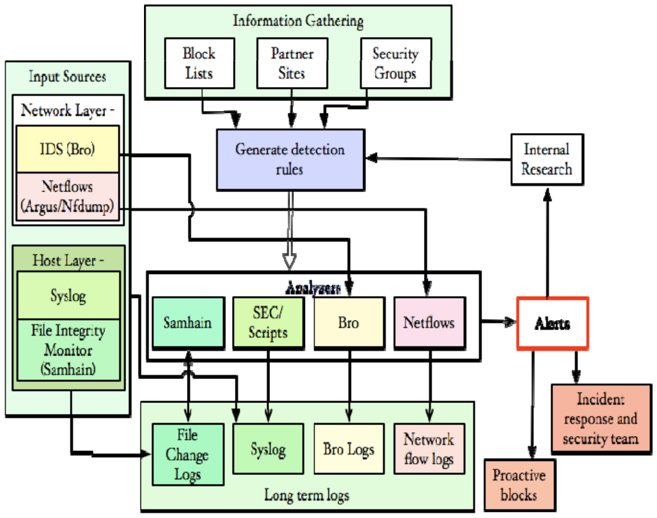

Figure 3: Monitoring Architecture Deployed at NCSA.

Host layer input sources in Fig. 3 consist of *syslog* data and file change logs generated by the *File Integrity Monitor (FIM)* (Samhain Labs: *la-samhna.de/samhain*). The central syslog collector has built-in redundancy in order to execute in a failsafe mode. The installation of FIMs and central syslog servers is limited for reasons of performance, data collection cost, and scalability. However, network monitoring can somewhat compensate for the selective deployment of FIMs to indentify an attack and provide relevant forensic information.

Additional detection rules are generated from information gathered from partner sites, security mailing groups or blacklists (watchlists) of malicious IP addresses provided by the peers.

Finally, it should be noted that anti-virus and anti-malware tools are deployed within the network. However, these tools are used as a part of post-incident investigation/cleanup rather than for proactive alerting. Very decentralized architecture and management of the system with no mandate for a standard software stack and high false positive rate are primary reasons to not use these tools as primary alert mechanisms.

A. Analyzers and alerts

Data collected by all four monitoring tools (network-based: Bro IDS and NetFlow analyzers; and host-based: syslog server and file integrity monitor) is analyzed for anomalies and signature matches. For instance, Bro IDS alerts are generated when a protocol-based rule or policy is violated - when a DNS query is made from/to a hostile domain or when an internal host downloads a binary from a blacklisted URL (using HTTP protocol). Custom policy rules designed to monitor NCSA's cluster are based on network traffic characteristics of cluster nodes, e.g., no compute nodes of the cluster should be downloading anything. *Netflows* trigger alerts based on the predetermined network traffic characteristics (e.g., number of bytes transferred, traffic to/from a watchlisted IP address, and connection to/from specific ports). File Integrity Monitor watches for unauthorized modification, creation, deletion or permission changes in specified system files, e.g., configuration files and OS binaries. A Simple Event Correlation engine (SEC) [18] uses syslog data to generate alerts on anomalies based on rule sets, such as new user creation, user authentication and command history profiles and privilege escalations by new users, etc.

In addition to the generated alerts, the data from all monitoring sources is archived. This can be valuable in conducting forensic analysis while tracing specific attacks.

The *Security Team* determines the response action warranted for the alerts generated by the monitoring tools. Some of the alerts result in proactive actions, e.g., sending reset packets to a scanned IP address (or proactive blocks). Other actionable [3] alerts (those determined to be true positives) are forwarded to the *Incident Response and Security Team (IRST)* for investigation and mitigation of the incident. At the end of every investigation, the Security Team performs its own internal analysis to calibrate the monitoring tools based on the lessons learned.

While each of the monitoring tools provides important data in its own right, it is also limited in scope by its nature and by its deployment, in terms of access to network and system infrastructure and the quality and amount of data gathered. A combination of these tools provides a comprehensive view of activity inside the network and the systems.

III. Security Implications of an Open Network Architecture

NCSA is an open network architecture that supports services ranging from high-performance scientific computations and web, email and file servers to small research clusters and user desktop/laptops. Because of very high bandwidth requirements (support of multiple 10 Gb links), traditional firewall strategies are not sufficient for providing security protection [1]. Researchers and operational staff are allowed to run their own flavors of operating systems and software, which results in a very heterogeneous computing environment. There is no mandatory software stack requirement for the systems except for certain production and critical infrastructure machines. Due to stringent dependency requirements of exclusive scientific computing software (e.g., compilers and proprietary code) and grid-based file systems (e.g., GPFS), often kernel upgrades are delayed.

Furthermore, the Grid computing component of the NCSA network environment brings the challenge of having a very broad base of users, each one of which needs a way to authenticate to systems and has his/her own specific requirements for running jobs. The widespread user base often makes the network boundaries blurred and any successful adversarial action at NCSA may affect peer sites. Likewise, security breaches at peer sites directly affect the security state at NCSA. Additionally, the NCSA security team doesn't have much control over the security of the end users' computers. Different administrative boundaries for grid computing environments make it difficult to have a unified security policy that not only addresses concerns of users locally but also reaches across the grid.

IV. Related Work

While many techniques to protect against attacks are available, relatively little has been published on comprehensive measurement-based analysis of different types of attacks, from alerts to identifiable incidents. Databases like those provided by CERT *(www.cert.org)* and CVE *(cve.mitre.org)* document vulnerabilities and possible exploits. These data are often used for analyzing and modeling vulnerabilities. For example, [21] uses data mining techniques to study traffic data during an attack for identifying signatures of intrusion detection; [4] uses vulnerability data to develop a finite-state machine model for analyzing vulnerabilities and attacks at the code level. [19] describes the analysis of a single denial-of-service attack on a server. Several studies [12], [13], [23] have analyzed attack

[3] Of roughly 140 alerts seen every day, each alert is actionable and is investigated until determined to be a false positive (vast majority of the daily alerts turnout to be false positives).

data to build formal models involving both single and multiple nodes. Several authors have proposed models that correlate alerts to incidents. Two examples include [10], where a capability-based model is proposed and verified using real attack data, and [15], where prerequisites and consequences model are discussed. In [22] the authors view attacks as a set of capabilities that provide support for abstract attack concepts. Using this notion the paper introduces a model for computer attacks, and a language for specifying the model and shows how it can be used in vulnerability analysis, intrusion detection and attack generation.

Other sources that collect attack data include honeypot experiments [6], [11] and the DETERlab Testbed (*www.isi.edu/deter*). *Red teams* have often been used to collect network vulnerability data; these data are generally unpublished. Several studies focus on classification of security flaws, vulnerabilities, and attacks; examples include [3], [14], and [5]. In [17] an attack is described as a natural progression through seven unique phases; we use this classification in our analysis. In addition, the Bro team (*bro-ids.org*) conducted extensive studies of attacks for the purpose of developing monitoring tools. While [24] is a comprehensive report on the analysis of security incidents between 1989 and 1995, it does not concentrate on how incidents were detected. Most recently, the Verizon Business RISK Team investigated 90 security breaches in 2009, encompassing 285 million compromised records [24]. This report also concentrates on the effect of the incidents instead of their detection.

Studies show that modeling can be useful in understanding attack patterns and building more efficient protection mechanisms. For example, [4] introduces a finite state machine model methodology to analyze operations involved in exploiting application vulnerabilities and to identify the security checks to be performed at the elementary activity level to foil an attack. More recently, [9] proposes a state machine-based attack model that is verified in an emulated environment using real security monitors. In this approach the model is limited in scope and an independent state machine is built for each incident. Likewise, [26] proposed a state machine model-based language to describe computer penetrations as sequences of actions that an attacker performs in order to compromise a computer system.

Reference [25] presents a comprehensive analysis of security incidents on the Internet for the period of 1989-1995. However, during the last 20 years, computing environments and threats (motives, models) have changed significantly. While [25] presents solid classification schemes which are highly useful, it is important to note that some observations are somewhat outdated. Quoting from the report, "Estimates based on this research indicated that a typical Internet domain is involved in no more than one incident per year and that a typical internet host in no more than around 1 incident per 45 years." This conclusion clearly does not represent the state of security in 2010 as indicated by our study of data on security attacks, which occurred in a large computing organization over the last 5 years. Both [24] and [25] concentrate on the effect of the incidents and their classifications while we also attempt to analyze alerts and incident data to develop an understanding of (i) how incidents are getting detected and (ii) how incidents progress.

An important recent development is establishing (with support from DHS) of PREDICT, a repository for security-relevant network operation data. The data can be used by researchers to develop new models and technologies to assess cyber threats to the computing infrastructure and increase cyber security capabilities (*www.predict.org*).

V. Analysis of an Example Incident

In this section, a *credential compromise incident* is used as an example to illustrate (i) the type of alerts and ancillary data provided by the security monitoring systems and (ii) the analysis process starting from an alert to incident identification. Data snippets from a real log are used to walk through the analysis process. In this example incident, the attacker logs-in the victim host using a stolen credential (password) and obtains root privileges using a local root escalation exploit. The end goal of the attacker is to harvest passwords and authentication keys of other users, using trojaned versions of SSH and SSHd.

(1) An IDS alert shows suspicious download on a production system (victim: *xx.yy.ww.zz*) using the HTTP protocol from remote host *aa.bb.cc.dd* (the hostname and IP address are anonymized).

May 16 03:32:36 %187538 start xx.yy.ww.zz:44619 > aa.bb.cc.dd:80 May 16 03:32:36 %187538 GET /.0/ptrat.c (200 "OK" [2286] server5.bad-host.com)	IDS

The data snippet above, shows an alert generated by the *Bro HTTP analyzer* due to the violation of a predefined bro-policy, i.e., download of an unauthorized file (*ptrat.c*). This file is suspect because: (a) this particular system is not generally expected to download any code except patches, system updates, and other relevant binaries, but then only from authorized sources, (b) the downloaded file is a C language source code, and (c) the server from which this source was downloaded is not an authorized software distribution repository.

While the alert generated suggests the download of a suspicious source file, it does not give a sufficient context of events prior to the download (such as a login, execution of exploit, abnormal number of bytes transferred, or a scan). The alert does not reveal what caused the potentially illegal download request, e.g., an exploit code, a potentially malicious user, or a web application making a request, malicious or otherwise.

(2) Correlation with network flows. The network flows reveal connections with other hosts in close time proximity to the occurrence of the download: (i) SSH connection (port 22) from IP address *195.aa.bb.cc*, and (ii) multiple FTP connections to *ee.ff.gg.hh, pp.qq.rr.ss*.

09-05-16 03:32:27 v tcp 195.aa.bb.cc.35213 -> xx.yy.ww.zz.22 80 96 8698 14159 FIN 09-05-16 03:33:36 v tcp xx.yy.ww.zz.44619 -> aa.bb.cc.dd.http 8 6 698 4159 FIN 09-05-16 03:34:37 v tcp xx.yy.ww.zz.53205 -> ee.ff.gg.hh.ftp 1699 2527 108920 359566 FIN 09-05-16 03:35:39 v tcp xx.yy.ww.zz.39837 -> pp.qq.rr.ss.ftp 236 364 15247 546947 FIN	Flows

While the SSH login could explain the exploit download, the connection record does not reveal whether authentication was successful or what credentials were used to authenticate.

(3) Manual correlation with syslog alerts. The *syslog* confirms a user login from *195.aa.bb.cc*, which is unusual, based on the user profile and behavior pattern (user logs in for the first time from this IP address).

May 16 03:32:27 host sshd[7419]: Accepted password for user from 195.aa.bb.cc port 35794 ssh2	Syslog

Four data points are established from the analysis: (1) a suspicious source code was downloaded, (2) the user login occurred at nearly the same time as the download, (3) the first time login from IP address *195.aa.bb.cc*, and (4) additional communication to other ports (FTP).

(4) Additional manual analysis. Search of all the files owned and created by this user found a footprint left behind by a credential stealing exploit – a library file, *libno_ex.so.1.0,* created at the time of the download, which is symptomatic of a credential-stealing exploit. This signature, located in the *tmp/* folder, confirms that the account was indeed used to download malicious code as identified below.

-rwxrwxr-x 1 user user 3945 May 16 03:37 /tmp/libno_ex.so.1.0	Directory listing

The *libno_ex.so.1.0* library file is known to be created when a code exploiting vulnerability cve-2009-1185 is successfully executed. Upon further investigation, it is determined that the attacker (i) successfully obtained root privileges in the system, (ii) replaced the SSHd daemon with its trojaned SSHd version, and (iii) captured passwords in the file */lib/udev/devices/S1*.

[root@host dir]# ls -l /proc/10589/fd lrwx------ 1 root root 64 2009-05-16 11:21 2 -> /dev/null lrwx------ 1 root root 64 2009-05-16 11:21 3 -> /lib/udev/devices/S1 *Note: Trojaned sshd is running as process id 10589 with root privileges and writing to a file in /lib/udev/devices/S.1*	Bash output

The example shows that:

- By systematic analysis of logs and correlation of different data points one can see an increasing level of suspicion of an imminent attack. Our conjecture is that if we can develop techniques to preempt the attack action and potentially, let the user progress under probation until the real intentions are clear, we may have a greater chance of preventing system misuse. The concept of execution under probation has been used in HP Non-stop systems to cope with accidental failures [7].
- The incidents analysis requires correlating data from varied monitors and system logs with human expertise. Using the process outlined with the example, we conduct an in-depth analysis of data on security incidents with the primary objective to characterize incident types and severity, extract the attack stages, correlate phase of an attack and the time when an alert is generated, and quantify the detection coverage of different alerts. Insights resulting from this analysis can be used to (a) drive design and strategic placement of defense mechanisms, (b) establish heuristics for automated (or semi-automated) analysis and correlation of data across the different logs to enable an early detection of potential intruders, and (c) construct a model to capture attacker's behavior at the network layer.

VI. Terminology

Before proceeding, we introduce basic terminology used in this paper: *alert, attack, incident, misuse, and severity*. An *alert* is a warning indicating a security violation within the network that warrants a response. An alert is generated when monitoring tools detect an anomaly or a known attack signature. An *attack* is an action taken by an attacker to obtain unauthorized access to a system or information. An *incident* occurs when an attacker is successfully able to exploit one or more vulnerabilities in the system and obtains unauthorized access. A *misuse* is an intentional, improper, or unauthorized action performed using computing resources. *Severity* rating is defined in terms of the adverse effects of an incident on the operations, assets, or individual with the loss of integrity, confidentiality, and availability [8][4]. The choice of [8] for representing severity of incidents was made due to its simplicity in categorizing incidents and that it is the Department of Energy-recommended way of incident reporting.

VII. Alerts

In this section, we present the types of alerts responsible for identifying a given incident. Data on 150 incident investigations (resulting in 124 actual incidents and 26 false positives, where a false positive is synonymous with full investigation resulting in no *incident found*) is studied to characterize both the attacks and the corresponding alerts that led to incident discovery. All statistics provided in the paper are concerning the 124 actual incidents (unless stated otherwise). While there are approximately 140 actionable alerts per day from the NCSA network, we only focus on the ones that resulted in the discovery of incidents analyzed in this study. For selecting incidents we went in reverse chronological order. Incidents left out from the study are those for which there is no proper documentation and forensic data. 150 incidents constitute more than 80% of the total incidents and are representative of all the incidents in last 5 years at NCSA

Table I provides an overview of the number of alerts generated by each monitoring tool. Rows marked as *INC* and *INV* provide the number of alerts corresponding to all actual incidents and all incident investigations, respectively. Overall about 87% (108/124) of incidents are detected by

[4] The choice of [8] for representing severity of incidents was made due to its simplicity in categorizing incidents and that it is the Department of Energy recommended way of incident reporting.

anomaly-based detectors and only 13% (16/124) by signature-based detectors. Table I shows also that the IDS and Flows monitors detected 31% and 26% of incidents, respectively. Note that 27% of incidents went undetected by any alert (more discussion follows in Section VIII-D.)

A. *IDS Based Alert*

IRC (Internet Relay Chat) Analyzer triggers an alert when the Dynamic Protocol Detection mechanism of Bro IDS sees a suspicious IRC connection (e.g., IRC traffic on a non-standard port). The *IRC analyzer* caught 15 incidents while exhibiting a very low false positive rate (1/16 = 6%). This is because the IRC alert uses a highly-customized detection mechanism finding connections to command and control channel derived based on the characteristics of known attacks.

Scan (internal, external scans and SSH). Even though most incidents start with a reconnaissance phase with scan being an integral part of it, the vast majority of scan alerts are false positives or inconclusive at best, and hence, most of them are ignored. In the analyzed data only one incident was discovered with the help of pre-attack scan activity. However, seven incidents were discovered due to the post-attack scan in the attack-relay/misuse phase where attackers (more precisely viruses installed by the attacker) were probing other networks/hosts in order to propagate further.

HTTP and FTP analyzer alerts are generated if a known exploit signature is matched or an attacker is downloading exploits directly from publicly available exploit repositories. This is a high-maintenance alert and requires constant updating of a list of the exploit signatures. These alerts caught six high severity incidents in the early stages of the attack (penetration phase), when attackers were trying to download an exploit on a host (see Section VIII-B for the discussion on the attack phases).

Malware analyzer alerts are based on the use of public availability of a malware hash registry which provides pre-computed MD5 and SHA1 hashes for known identified malware. Often attackers rename the file types (e.g., Linux binary downloaded as *.jpg* file) to defeat signature-based matches on filenames. These alerts detected 6 incidents, all related to spam-bots and virus downloads in the network.

Virus signature matching can be effective in detecting malware; however it can be defeated by polymorphic viruses. Thus, customized alerts are implemented as a *Bro policy* to detect anomalies corresponding to the behavior of some of the most popular virus/worms (e.g., *blaster* and *nimda*), instead of relying solely on known signatures.

B. *Network Flow- Based Alerts*

TopN is triggered when network traffic for the host crosses over a certain threshold. While this alert caught the maximum number of incidents (18), it exhibits a high false positive rate (33% = 9/27), and the detected incidents are of moderate to low severity. This is a somewhat counterintuitive because one would expect that an alert that detects a large number of incidents would also cover a proportional number of high severity incidents.

Undernet flows work in conjunction to the IRC analyzer and trigger on very specific IP address connections, e.g., an IRC connection to the channels on Undernet IRC servers used to host channels for *botnet* command and control.

Watchlist is triggered when an attacker connects from a known bad IP address. This alert caught 11 incidents. Four of those incidents were of high severity (see Section VIII-C for the discussion on incident severity). Although it is a common belief that attackers can easily change IP addresses, the fact that the blacklisted IP watch list is successful in detecting attacks shows that many attackers either have limited resources (compromised systems) or are careless in hiding trails. This also shows the importance of sharing relevant information between peers.

Darknet flows: alerts are generated when connections are made to an unused IP block within the network specifically by an internal IP address. Since no traffic should be sent to unused IP subnets, a connection to such address space is usually an indication of an automated scan performed by an attacker, virus, or worm.

TABLE I. ALERT TYPES GENERATED BY THE MONITORING TOOLS

#	IDS alerts (31%)								Flows alerts (26%)				Syslog (11%)		FIM (4%)	Other (1%)	No alerts (27%)				
	Packet/Protocol Analyzers								*Traffic Anlyzers*				*Profile*		*Host*		*3rd Party Notification*				
	IRC	**Internal Scan**	**HTTP**	**Malware**	**FTP**	**SSH Scan**	**Virus/ worm**	**External Scan**	**TopN**	**Undernet**	**Watchlist**	**Darknet**	**Login**	**Command**	**File change**	**Google alert**	**Mailing list**	**External**	**Peer**	**User**	**Admin**
INC 124	**15**	**6**	**6**	**5**	**3**	**1**	**1**	**1**	**18**	**2**	**11**	**1**	**10**	**4**	**1**	**5**	**4**	**14**	**2**	**0**	**14**
Anomaly (108/124)	*15*	*6*	*2*	*4*	*0*	*1*	*1*	*1*	*18*	*0*	*11*	*1*	*10*	*4*	*0*	*0*	*4*	*14*	*2*	*0*	*14*
Signature (16/124)	*0*	*0*	*4*	*1*	*3*	*0*	*0*	*0*	*0*	*2*	*0*	*0*	*0*	*0*	*1*	*5*	*0*	*0*	*0*	*0*	*0*
INV 150	**16**	**6**	**7**	**6**	**3**	**1**	**1**	**1**	**27**	**2**	**13**	**1**	**11**	**4**	**1**	**5**	**4**	**19**	**3**	**1**	**18**

C. *Custom Alerts*

Google Alert maintains an index of words found on web pages it crawls to build the search database and update the cache. Google Alert can be configured to send notifications when certain terms are found in the web page. In the NCSA network, Google alerts are configured to catch spam uploaded by miscreants in the comments section (world writable) of web pages or wikis.

D. *Notifications*

A total of 34 (27%) incidents were discovered based on third party notifications provided to the security team. In our analysis we consider these notifications as incidents missed by the security monitors. Section IX provides a detailed discussion on missed incidents. We break down notifications into the following groups:

Mailing list notification. Various information-sharing and analysis groups (REN-ISAC, CERT, FIRST) transmit notifications of a malicious activity that originates from our network.

External notifications. External sources notify the security team when they notice or attribute the origin of malicious activity or abuse to the NCSA network. A total of 14 (11%) incidents were discovered in this manner. However, these notifications also resulted in five false positive investigations. Nearly 78% of the external notifications (11/14) were related to an internal host scanning outside IP blocks or sending out spam.

Peer. A peer notification highlights a possible security issue or an incident which may have an adverse effect on our network because of shared resources among the users. The two notifications from peer sites corresponded to high severity incidents.

Users weary of unexplained behavior on their systems may request to have their system examined by security experts. Only one incident was reported by a user and turned out to be a false positive. This is an interesting finding, which shows that often users continue to use their systems unaware of the breach.

Administrator notification is received when something suspicious or anomalous is observed in a system, e.g., an unexplained account has been created on the system or a hard disk becomes suddenly full (without a plausible explanation). Along with external notifications, administrator notifications detected 14 (11%) incidents.

VIII. Classifying and Analyzing the Incidents

For every incident/attack, described in this paper, data logs are correlated to extract the following information: (i) incident type, (ii) alert generated (in most cases, a single alert was responsible for detecting an attack; where there were multiple alerts raised, we used the first one as the detector), (iii) exploit used, (iv) misuse of compromised host, (v) relevant monitoring logs, (vi) privilege attained by the attacker, (vii) attack phase, and (viii) severity.

The choice of using first alert was specifically made to avoid the bias towards favoring well-understood incident types for which multiple alerting mechanisms were developed over time. Counting first alert helps us normalize all incidents and provides uniform look. Ignoring secondary alerts (which was not a common occurrence in our dataset) does not affect our findings.

Table II summarizes the characteristics of three sample incidents. *Record 1* corresponds to an *application compromise* incident where the attacker obtained root privilege by exploiting the *xp_cmdshell MSSQL Server* vulnerability and used the system as an unauthorized media server (*warez*). *Record 2* represents a virus infection on the host. Finally *Record 3* shows credential compromise incident where the attacker penetrates the system using stolen credentials.

An accurate categorization of incidents is essential to guide a recovery strategy for an incident. For example, a virus infection incident (*infected system* category) can be recovered by cleaning up the system using antivirus software, whereas a local root compromise (*credential compromise* category) may require a more comprehensive cleanup strategy, such as complete re-installation of the system and a mandatory change of user passwords. This section classifies all incidents in our data in three ways: *incident type/category, attack phase, and incident severity.* The *incident type* classification sorts the incidents into nine broad categories based on the outcome of the investigation, the similarity of the attack type, and the attacker goal (intended misuse). The *attack phase* classification attempts to break down an attack lifecycle into seven different phases [17] and provides an important measure of the effectiveness of monitors. The *severity* measures the impact of an incident on the computing enterprise.

TABLE II. Sample incidents

ID	Incident Type	Monitor/ Alert	Exploit used	Misuse	Privilege obtained	Attack Phase	Severity
1	Application Compromise	Flows/TopN	xp_cmdshell MSSQL Server	Warez unauthorized media	root	Attack relay/ misuse	Medium
2	Infected System	IDS/Blaster	W32.Welchi worm	Scan ICMP 2048	user	Attack relay/ misuse	Low
3	Credentials compromise	Syslog/ Profiling	Stolen credentials	Sniff credentials	root	Breach	High

TABLE III. INCIDENT TYPES

Incident Type (count)	Vulnerability/Exploits (count)	Incident compromise specifics (count)	Alert generated (count)
Credential compromise (32)	Stolen password/key-pair (31)	Root (rootkit + trojan ssh/sshd) (7)	FTP Analyzer (3)
User credentials are targeted and stolen. Attack propagates by using stolen credentials and local root escalation exploits.	Open-X11 keystroke logging(1)	User (key-pair/certificate) (21)	HTTP Analyzer (3)
		Spam (1)	IRC Analyzer (1)
		Bot (1)	Notification (9)
		Scan NFS file system (2)	User profiling (11)
			Watchlist (5)
Web server/application (22)	PHP Remote command execution/ code injection (11)	Defacement (5)	Darknet (1)
Web servers (e.g. IIS or Apache) and/or web applications (e.g. phpmyadmin or wiki) compromises	Web server misconfiguration (7)	Scan other hosts (5)	Google alerts (4)
	IIS permissions (1)	Spam (4)	HTTP (1)
	Unknown (3)	Backdoor (3)	IRC (1)
		Bot (1)	Malware (1)
		Malware (1)	Notification (8)
		Open proxy (1)	Scan int/ext (1)
		Un-auth FTP server (1)	TopN (4)
		Incorrect permissions (1)	Watchlist (1)
Application compromise (22)	Unknown (6)	Warez (10)	HTTP (1)
Compromise of application level software (e.g. VNC, OpenSSL, and mysql)	VNC exploit (6)	Scan (5)	IRC (5)
	Mysql exploit (2)	Backdoor (3)	Notification (3)
	Telnet exploit (2)	Bot (1)	Scan int/ext (3)
	Rlogin exploit (1)	IRC Bouncer (1)	TopN (8)
	SSL exploit (2)	Sniffer (1)	User profiling (1)
	OpenX11 exploit (1)	Unknown (1)	Watchlist (1)
	WINS exploit (1)		
	MS-SQL exploit (1)		
Bruteforce SSH (20)	Weak Passwords (19)	Bot (9)	IRC (8)
Attackers bruteforce to guess password for well known accounts (e.g. root/guest/test) for SSH	Misconfiguration (1)	SSH Bruteforce Scan (7)	Notification (2)
		IRC Bouncer (2)	SSH scan (1)
		root exploit (2)	TopN (3)
			Undernet (2)
			User Profiling (2)
			Watchlist (2)
SPAM/phishing (11)	Unknown (6)	Spam (8)	Google alerts (1)
Incidents resulted in sending spam and phishing emails.	Link click (3)	Web spam (2)	HTTP (1)
	Misconfiguration (2)	Malware distribution (1)	Malware (4)
			Notification (4)
			Watchlist (1)
Infected system (11)	Unknown (8)	Scan (7)	Notification (4)
A host in the network is infected with a virus or a known malware.	Confiker (1)	Spyware (1)	Scan int/ext (3)
	W32.Welchia.Worm (1)	Bot (1)	TopN (2)
	W32/SDBot (1)	IRC Bouncer (1)	Virus/worm (1)
		DDoS (1)	Watchlist (1)
Pre-infected host (3)	Unknown (3)	Scan (3)	Notification (1)
Systems infected outside the network boundaries and brought back into the network.			TopN (2)
Social engineering (2)	Instant Messenger (1)	Privilege Escalation (1)	Notification (2)
Attempts to manipulate users to divulge critical system information such as passwords.	Unknown (1)	ID theft (1)	
Internal investigation (1)	Insider (1)	Unauthorized privilege retain (1)	File Integrity Monitor (1)
A policy violation by an action of an internal employee/user.			

A. Incident Type

Classification in Table III provides a description of incident types along with their counts. Each incident category includes attacks/compromises which share similar characteristics. For example, the set of analyzed incidents includes 22 successful attacks involving exploitation of certain vulnerabilities in applications such as *VNC, MySql,* *MSSQL,* and *OpenSSL*. All these incidents are combined into the *application compromise* category. Likewise, 32 incidents that resulted in an attacker gaining access to the system by using stolen password/key-pairs are combined together in the *credential compromise* category.

Closer analysis of the identified incident types leads to several observations:

- The majority of incidents (55%) were due to attacks on authentication mechanisms with varying levels of sophistication: (i) *bruteforce SSH* accounts for 19 incidents, (ii) *credentials compromise* – 32 incidents, (iii) *application compromises* – 10 incidents (VNC null session – 6 incidents, and exploiting vulnerabilities in telnet, rlogin, Open X11 – 4 incidents), and *Web server/application* – 7 incidents due to Web server misconfiguration.
- Web-based attacks (*Web server/application* category) were primarily a result of misconfiguration (7) or web-application vulnerability (10). In terms of misuse, attackers were seen establishing backdoors (3) and scan other hosts (5), rather than mere defacements (i.e., subverting the web site content).
- Attackers deploying unauthorized media (10 *warez* incidents) were found to be very sophisticated in their ability to cover their tracks and malware by installing rootkits and hiding media using alternate data streams. All 10 incidents were part of the *application compromise* category, thus indicating that attackers dealing in distribution of illegal media have access to a much wider range of exploits.
- *Application compromises*, *Spam/phishing* and *Infected systems* categories combined caused a total of 20 (16%) incidents where the initial entry point was unknown. Even though viruses and worms are studied in depth individually, the variation of payload delivery (e.g., innocuous email attachments, viruses replicating themselves via open network shares, or exploits of web browser vulnerabilities) is so broad that most of entry points are difficult to establish. This observation calls for monitoring techniques which can quickly adapt to changes in attackers behavior

*Association of incid*ent *types and* alert *categories* reveals that each alert detects multiple incident types. This suggests that while the exploits may be different, the incidents share a common attack path. For example, in a *credential stealing incident* an attacker logs in with a known credential, downloads local root escalation exploit over HTTP using *wget*. In an *application compromise incident*, a host running *phpmyadmin* downloads a *c99 phpshell* via http after undergoing a remote code injection exploit. Both incidents have different attack vectors, different exploits, and different misuse. However, the http-user agent is determined to be *wget.* Alerting for anomalous http requests originating with "wget" as user-agent can reveal two distinct incident types. Fig. 4 shows that *bruteforce SSH* incidents can trigger IDS alerts (*IRC* and *Scan*), flow alerts (*TopN* and *watchlist*) and syslog alerts depending on the sequence of actions the attacker takes when penetrating the system.

B. *Attack Phases*

Using the attack stages proposed by [17], alerts for all incidents are mapped into seven phases. Table IV defines the incident phases. Numbers in the last column of Table IV provide (actual incidents/incident investigations) counts for each phase during which the alert was observed in our data. A useful metric to study the effectiveness of a detection mechanism is the phase of the attack during which the notifying alert originated.

Note that not all the incidents are necessarily required to fulfill each stage of attack described in the Table IV. For example, while each attack usually leads to the misuse of the system, not all the attacks require the embedding phase. This raises the possibility that low number of notifications in a stage may be due to few incidents in the available data set encompassing that specific stage, other than due to the deficiency of the monitoring tools. Thus, absence of an alert in a particular attack stage is not necessarily an indication of the deficiency of the monitoring tools.

However, Table IV also clearly reveals a skewed picture for early stage (scan phase) versus late stage detection (misuse phase) – both of these phases are generally unavoidable. For other intermediate stages (where applicable) not many monitors are deployed and configured to alert at that level of granularity, especially in a non-centralized environment.

Additionally, detection latency (amount of time monitoring tools took to alert on the incident) is not being considered in our analysis. Close look at data revels most alerts are generated near real time.

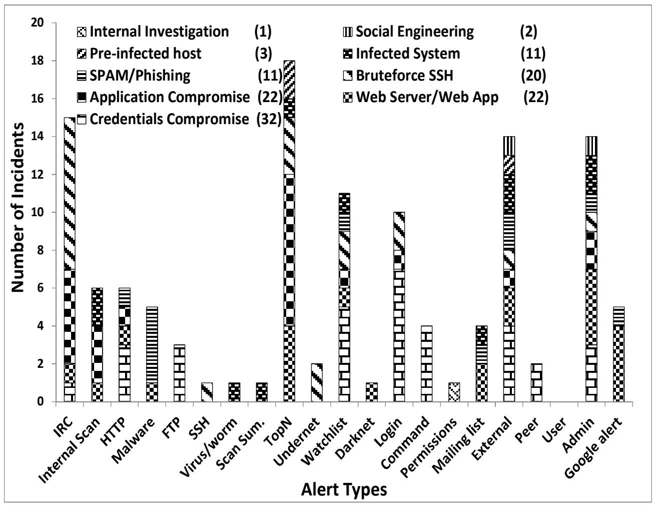

Figure 4: Alert distribution based on incident types

As shown in Fig. 5 nearly 39% (48/124) of detected incidents were discovered in the very last stage of the attack, i.e., in the attack-relay/misuse phase. This data indicates two major issues with detection mechanisms. First, an attacker is successful in concealing his/her identity/presence during the initial phases of the attack, and monitors fail to detect the compromise early. Perhaps a notion of pre-emption and execution under probation can be of value here. Second, exploit signature-based alerts were able to detect only 13%

of the incidents, while anomaly-based detectors (which capture the impact of an exploit or an attacker's actions) caught the remaining 87%.

TABLE IV. ATTACK PHASES

Attack Phase	Description	Incident Count
Scan Phase	Attackers try to identify vulnerable hosts and gather information about the target, e.g., services that are running.	1/1
Breach Phase	Attackers gain access to the system (e.g., using stolen or guessed credentials or by exploiting system misconfiguration (e.g., world writable files on an open share).	30/39
Penetration	Attackers exploit vulnerability (e.g., buffer overflow vulnerability) to obtain unauthorized access to the system.	9/10
Control	Attackers set up the compromised host to accept remote commands and provide reusable access (e.g., connect to command and control channel or install a backdoor).	21/23
Embedding	Attackers hide their malware and tracks by embedding the malware in the system, e.g., installing a rootkit, deleting system logs, adding ssh keys to authorized_key file, changing configuration files.	8/9
Data extraction/ modification	Attackers change or modify data in the system, e.g., deface web pages, copy database content, or steal information.	7/7
Attack-relay/ misuse	Attackers start misusing the system for personal gain, e.g., spam, DDoS using a bot, password harvesting, distributing warez, spreading virus, and phishing.	48/61

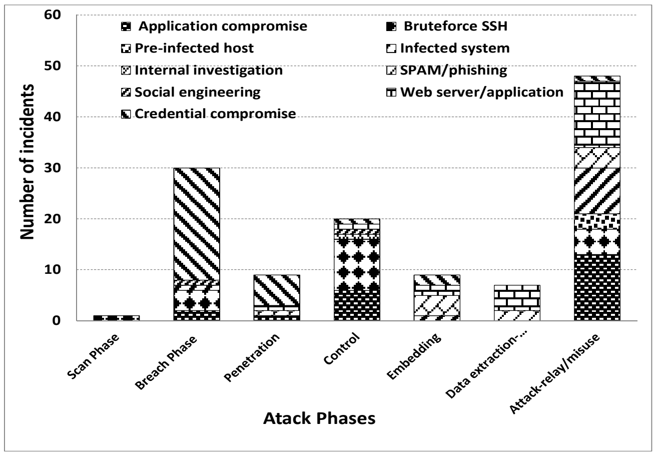

Figure 5: Incident Types versus Attack Phases

C. *Incident Severity*

A reason to categorize incidents based on severity is to determine whether monitors are catching harmful incidents as opposed to mere low-impact compromises. Severity is a qualitative measure that expresses the effect on loss of integrity, availability, and confidentiality in the system, application, services, and data. Analysis of incident severity allows determining whether security monitors are capable of detecting high profile incidents or mere low-impact violations. Table V provides definitions of the various degrees of severity along with the distribution of the analyzed incidents across the severity categories.

Incident Phase and Severity. Out of incidents detected in the attack-relay/misuse (last) phase, only about 30% were of medium-to-high-severity. While one expects a high-level of correlation between high detection latency and high severity (the longer the attacker stays in the system, the more potential there is for him to do damage) - this is not true from our data. This is because many of the high severity incidents are detected at an early stage. 62% (23/37) of high-severity incidents were caught in the breach phase, having already resulted in significant damage, e.g., attackers were already able to gain access to the system using stolen credentials. Such attacks cannot be detected (even with profiling of user behavior) until an attacker uses the stolen credentials to gain access to the system.

How early detection can help. An early detection can still limit the extent of the damage caused by the attack. For instance, for incident three (credential compromise) in Table II, an early detection could have prevented unsuspecting users from exposing their credentials on a host with a trojaned ssh server and a rootkit. Ideally, File Integrity Monitors should preempt change/modification of /usr/sbin/sshd when the legitimate SSH software gets replaced with the trojaned version.

TABLE V. INCIDENT SEVERITY

Severity	Incident Impact	#
Very High (Catastrophic)	A vast majority of users are affected due to the breach with successful root escalations	1
High (Very serious)	Production and administrative systems. Credentials compromise and application compromise (OpenSSL exploits, X-server key stroke logging) that allow attacker to obtain root level privileges on the systems	37
Medium (Limited)	Users and small cluster systems (affects entire research group); application level compromise (VNC, XP_Cmdshell mssql exploit), web server (Phpmyadmin, Php Horde), malware hosting.	24
Low (Little or no effect)	Non-production systems (affects an individual); brute force SSH, infected systems, spam/phishing.	62

In summary, the *top five alerts*, which account for detection of approximately 54% (67/124) incidents, include TopN (18 incidents, 5 types), IRC (15 incidents, 4 types), Watchlist (11 incidents, 6 types), login and command anomaly (14 incidents, 4 types), HTTP and FTP analyzer (9 incidents, 4 types). Again observe, in Fig.4, that each alert detects multiple incident types.

Login and command anomaly alerts have the highest success rates of all alerts in catching high-severity incidents 30% (11/37). However, these alerts are triggered after the fact, when the attacker is already in the system.

IX. MISSED INCIDENTS

Because of monitor imperfections, there are usually false negatives and false positives associated with the detection system. In this section, we discuss false negatives, i.e., incidents missed by the security monitors. In our analysis there are 34 (27%) missed/undetected incidents. All incidents missed by the monitoring system are discovered because of notifications by external sources (third party, mailing lists, peers, users or administrator). Upon notification from an external source, relevant logs are parsed to look for the signs of the incidents. Once confirmed, proper response actions are taken to address the incident.

Table VI summarizes the specific causes of missed incidents. Analysis of data on missed incidents reveals inherent limitations in current security monitoring setup: (i) inability to automatically produce a context of what is normal and abnormal in the observed events, (ii) limited ability for automated collection and analysis of attack pertinent information, and (iii) inability to cope with a large spectrum of attacks, malware, and network traffic. The following discussion illustrates these limitations using examples of incidents missed by the monitoring system.

TABLE VI. CAUSES OF MISSED INCIDENTS

Cause of missed incidents	Examples	#
Increased sophistication in attacks	A peer site gets compromised and an attacker logs-in with stolen credentials; zero-day exploits	6
Lack of signatures	Exploit of VNC null string authentication vulnerability	7
Admin misconfiguration	Web share world writable access or root login to accept any password	5
Inability to distinguish traffic anomalies in the network	Web defacement or use of web server to host malware; bot command and control traffic	10
Misconfiguration of security monitoring tools	Routers stop exporting the flows to central collector which prevents alerting	1
Inability to distinguish true positives from false positives	Human error	2
Inability to run monitors on all hosts and file systems due to large administrative and performance costs	Limited deployment of file integrity monitors on non-critical systems	3

Inability to produce information on what is normal and abnormal in a stream of observed events. About 26.5% of the missed incidents are credential compromises (a high impact category). For detection of incidents in this category, monitors rely on alerts which are based on detection of: (i) deviation in the user behavior as compared with the known user profile (using *syslog*), (ii) malicious code download (using *IDS*), and (iii) unexpected system file manipulation (using *file integrity monitor*). A combination of these three tools should allow high detection coverage to be achieved. However, detecting a multi-step attack, which uses stolen credentials, requires comprehensive runtime traffic analysis, including correlation of different events and accurate determination of what is normal and abnormal in the observed traffic.

In the current setup of the monitoring system: (i) syslogs are limited in detecting user profile anomalies since attackers masquerade as regular users while logging in to the system; (ii) IDS does not raise alerts when attackers do not download malware from a known source and often there is no built-in signature for a given exploit available; (iii) due to high operational costs associated with file integrity monitors, they are not installed on administrator systems targeted by the attackers.

Limited ability for automated collection and analysis of attack relevant information: In a credential stealing attack one of the actions the attacker takes is to download the malware/exploit on the system. Assuming that IDS is updated with a signature of this malware (or exploit code), IDS should generate an alert. However, due to lack of a context, it is difficult to determine what user account was used to download this malware. Furthermore, the attacker often deletes the malware from the directory, which makes things even harder to trace. Ideally, correlation of file integrity monitor data, IDS, and syslogs should suffice to construct an accurate event timeline. Currently, the incident response tools lack this capability.

Inability to cope with a large spectrum of attacks, malware and network: About 9% of the missed incidents are application compromises. The limited detection coverage is due to the lack of timely available signatures and emergence of new zero-day exploits. For instance, detection signatures for compromises due to *VNC null string authentication bypass* vulnerability (CVE-2006-2369), *OpenSSL SSL-Get-Shared-Ciphers Buffer overflow* (CVE-2006-3738) exploits were unavailable at the time of the attack, thus each of these attacks went unnoticed.

Infected hosts incidents (about 15%) are mostly due either to virus propagation or to a user accidently downloading malware and installing it on the system. A user who accidentally downloads and installs malware may have a very small footprint for generating alerts based on syslogs or file integrity monitors. Therefore, the detection of such incidents mostly relies on IDS and netflows. Ideally, good antivirus software running on the host system should be able to catch infection. However, often even in the presence of the antivirus software and updated signature databases, these detection mechanisms can be easily bypassed by an attacker, e.g., using social engineering attacks (i.e., tricking the user to click on a link or open email attachments) and/or exploiting browser vulnerabilities. Also, malware is designed specifically to evade anti-virus or other detection utilities. Putting additional restrictions on some legitimate user actions to prevent social engineering attacks is impractical because it affects system usability and degrades user productivity.

Web compromises constitute about 23.5% of incidents missed by the monitoring system. Most of the web compromises are caused by exploitation of the web application software, e.g., exploitation of PHP vulnerabilities that allow an attacker to execute commands remotely (unlike in the past when web compromises were due to vulnerabilities in the web servers). These compromises are hard to flag since there is no distinct behavior of the system post-attack, especially when attackers are uploading static content (e.g., spam pages, redirections or links, and web page defacements). Although HTTP content monitoring tools can flag such content traversing through the network, these deep packet content monitoring rules are expensive to run and keep up to date. For example, in one instance attackers used the web server to host malware, while no observable change in the system was noticed. This reveals the need for monitoring outgoing network traffic.

X. Conclusions and Future Work

In this paper we study security compromises that have occurred over a period of 5 years at the University of Illinois NCSA network. Our observations from the analysis can be summarized as follows:

- IDS and NetFlows monitors detected 31% and 26% of incidents, respectively. 27% of incidents went undetected by any alert.
- Alerts are not uniform in their ability to detect attacks. The same alert can be triggered by different attacks. This is because different incidents share common attack paths, i.e., the basic steps followed by different attacks in penetrating the system are often similar regardless of the vulnerability exploited.
- Anomaly-based detectors are seven times more likely to capture an incident than are signature-based detectors. This is because the signatures are specialized to detect the presence (or download) of a known malicious binary. Consequently, they can be subverted. The signature-based detectors (due to their specialization) have fewer false positives compared to the anomaly-based detectors. (This study does not quantify the false positive rates of the alerts.)
- Nearly 39% of the incidents are detected in the last stage of the attack, i.e., attack-relay/misuse phase. This kind of detection often comes too late, since the damage to the system has already occurred.
- There is no indication of high-level correlation between the attack phase when an incident is detected and the incident's severity, e.g., incidents detected in the last stage (long latency detection) of an attack are not necessarily high severity incidents.
- Alerts detecting the most incidents may not be the most efficient alerts, e.g., TopN detects 15% (18/124) of the incidents of low to medium severity, but it exhibits a 33% false positive rate.
- While 14 incidents were detected based on reports from external, unrelated sources, one incident was reported by a user (which turned out to be a false positive) showing that users are not trained to identify a security breaches.

While in this analysis we consider only alert types that were triggered by the incidents included in this study, there are other warning mechanisms present in the system that never detected incidents. These detection capabilities should be reexamined to understand why these detectors are inactive. Such analysis would enable restructuring the monitoring system and adapting detection capabilities to changes in the underlying infrastructure and the growing sophistication of attackers.

XI. Acknowledgment

This work was supported in part by NSF grants CNS-05-51665 (Trusted Illiac) and CNS 10-18503 CISE, the Department of Energy under Award Number DE-OE0000097, IBM Corporation as part of OCR (Open Collaboration Research), and Boeing Corporation as part of ITI Boeing Trusted Software Center.

References

[1] Singer, A., "Life Without Firewalls," The Usenix Magazine, 28(6), 2003.

[2] Allman M., Kreibich C., Paxson V., Sommer, R., Weaver N.: "Principles for Developing Comprehensive Network Visibility," USENIX Workshop on Hot Topics in Security, USENIX, 2008.

[3] Bellovin, S. R., Cheswick, B.: Firewalls and Internet Security: Repelling the Wily Hacker. Addison-Wesley Publishing, 1994.

[4] Chen S., Kalbarczyk Z., Xu J., Iyer R. K., "A data-driven finite state machine model for analyzing security vulnerabilities," Int'l Conference on Dependable Systems and Networks, 2003.

[5] Cohen, F. B.: Protection and Security on the Information Superhighway. John Wiley & Sons, New York (1995).

[6] Cukier, M., Berthier, R, Panjwani, S., Tan, S.: A statistical analysis of attack data to separate attacks. Proc. Int'l Conference on Dependable Systems and Networks, (2006).

[7] Cutts Jr. et al, United States Patent 5,193,175, March 9, 1993

[8] DOE M-205: Cyber Security Incident Management Manual. Department of Energy (2010).

[9] Gregorio-de Souza I., Berk, V. H., Giani A., et al., "Detection of Complex Cyber Attacks," SPIE 6201, 2006.

[10] Zhou J., Heckman M., Reynolds B., Carlson A., and Bishop M., "Modeling Network Intrusion Detection Alerts for Correlation," ACM Trans. on Info. and Sys. Security 10(1), 2007.

[11] Kendall K., Smith A. C., "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems," MIT, Electrical and Computer Engineering. Cambridge 1999.

[12] Kumar, S, Spafford, E., An application of pattern matching in intrusion detection. Purdue University, Tech. Rep, Department of Computer Sciences (1994).

[13] Kumar, S. "Classification of intrusions," Purdue University, 1995.

[14] Landwehr, C. et al., "A Taxonomy of Computer Security Flaws," ACM Computing Surveys, 26(3), 1994.

[15] Ning P. and Xu D., "Learning Attack Strategies from Intrusion Alerts," 10th ACM Conference on Computer and Communications Security, 2003.

[16] Paxson V., "Bro: A System for Detecting Network Intruders in Real-Time," Computer Networks, 1999.

[17] Ruiu D., "Cautionary Tales: Stealth Coordinated Attack HOWTO," http://althing.cs.dartmouth.edu/secref/local/stealth-co-ordinated-attack.txt (1999).

[18] Vaarandi R., "SEC - A Lightweight Event Correlation Tool," Workshop on IP Operations and Management, 2002.

[19] Sung M., Haas M, Xu J. "Analysis of DoS attack traffic data," FIRST Conference, Hawaii, 2002.

[20] Sharma A., Kalbarcyzk Z., Barlow J., Iyer R., "Analysis of Credential Stealing Attacks in an Open Networked Environment," 4th International Conference on Network and System Security, 2010.

[21] Tidwell T., Larson R., Fitch K., and Hale J., "Modeling Internet Attacks," Workshop on Information Assurance and Security, 2001.

[22] Templeton S.J., Levitt K., "A Requires/provides Model for Computer Attacks," New Security Paradigm Workshop (2000).

[23] Treinen J., Thurimella R. "A framework for the Application of Association Rule Mining in Large Intrusion Detection Infrastructures," 9th Int'l Symposium on Recent Advances in Intrusion Detection, 4219 (2006).

[24] Verizon Business Risk Team: 2010 Data Breach Investigations Report, http://www.verizonbusiness.com/resources/reports/rp_2010-data-breach-report_en_xg.pdf

[25] Howard J. D., "An analysis of security incidents on the Internet 1989-1995," Carnegie Mellon University, Pittsburgh, PA, 1998

[26] Eckmann S.T., G. Vigna, and R.A. Kemmerer, "STATL: An Attack Language for State-based Intrusion Detection," Workshop on Intrusion Detection Systems, Athens, Greece, 2000.