Analysis of Security Data from a Large Computing Organization

A. Sharma[1,2], Z. Kalbarczyk[1], J. Barlow[2], and R. Iyer[1]

[1]Coordinated Sciences Laboratory, [2]National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign,
Urbana, USA
{aashish, kalbarcz, jbarlow, rkiyer}@illinois.edu

Abstract. In this work, we study security incidents that occurred over period of 5 years at the National Center for Supercomputing Applications at the University of Illinois. The analysis conducted: (i) quantifies the key characteristics of the incidents, such as category, severity, and detection latency, (ii) develops a data-driven, finite state machine model for describing incidents and exemplifies its use in the context of a credential compromise incident, and (iii) can facilitate the design and deployment of new techniques for security monitoring.

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

1 Introduction

In this paper, we analyze security incidents that occurred over a period of 5 years across 5000 machines monitored at the National Center for Supercomputing Applications 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.

Our goal is to analyze the efficiency of the security-monitoring infrastructure (including alerts raised and other ancillary data) in isolating and diagnosing the identified incidents. The analyzed incidents differ in nature; some clearly demonstrate a pattern of persistent attacks, while others result from random attackers exploiting vulnerabilities to get into the system and use it, for example, as a media server (warez). The analysis conducted: (i) quantifies the key characteristics of the incidents, such as category, severity, and detection latency, (ii) develops a data-driven, finite state machine (FSM) model for describing incidents and exemplifies its use in the context of a credential compromise incident, and (iii) can aid in the design and deployment of new techniques for security monitoring. The target system has naturally continued to evolve over the years, and hence the analysis is an aggregate over the measurement period.

2 Monitoring Tools

Like other large-scale production systems, NCSA uses a variety of techniques to detect a broad range of attacks on the system. The monitoring tools include: (i) an intrusion detection system (IDS), (ii) network flow collectors that "watch the network traffic externally and internally, (iii) a centralized syslog server to collect data on system and user behavior from production machines, and (iv) a host-based file integrity monitoring system running on production servers.

Table 1 provides an overview of the tools used to generate alerts and collect data on the security incidents analyzed in this study and the customizations or enhancements, if any, introduced by NCSA security team tools. The sizes of individual log files indicate the amount of activity recorded by the monitoring tools, including malicious and normal activity. The table also provides an overview of the alerts generated by each tool and the corresponding incidents/attacks that relate back to these alerts, based on analysis of the forensic data. We note some specific points about the monitoring tools and measurements.

Table 1: Monitoring Tools

Monitoring Tool		Description	Customization for NCSA Monitoring	Operating System (OS) Dependency	Avg # of Critical Alerts per Day	Log File Size (per Day)	Incidents in 5 Yrs
Network Based	Network IDS (Bro)	A network Intrusion detection system generates alerts on potentially suspicious/malicious activity by performing deep packet inspection of network traffic for defined anomalous activity. Examples include alerts when a protocol based rule or policy is violated – e.g., a DNS query from/to a hostile domain or access (using HTTP protocol) from a blacklisted URL.	Policy rules designed to monitor NCSA's cluster are based on network traffic characteristics of cluster nodes, e.g., no compute nodes should be downloading anything. FTP and HTTP protocols traffic is monitored for known malicious downloads. An IDS monitors' data for all the hosts within the network.	Runs at network layer - OS independent	40	3 GB	41
	Netflows Argus[1]- External Flows & Nfdump[2] - Internal Flows)	A netflow is a 7-tuple record of network transmission data. It includes IP address, ports, protocol, bytes and packet count for both source & destination hosts. Both Argus and nfdump generate alerts on connections from or to blacklisted IP address and domains, or increase in traffic above a threshold,	Argus is used to capture network flow data from the border routers. Nfdump captures flow data from internal routers. This setup provides visibility within the internal subnets along with operational redundancy. Netflow data is collected for all the hosts with in the network	Runs at network layer - OS independent	30	3 GB	41
Host Based	Syslog	A Syslog is an standard operating system and application logging service. At NCSA, all hosts running syslog forward their data to a centralized syslog server that acts as backup in case a hacker deletes logs on a victim system. Syslog data is used to create user activity profiles, e.g., based on SSH authentication logs collected on NCSA's production cluster systems. These profiles and other logs are analyzed using an event correlation engine to generate security alerts such as multiple users logging from a single source or an SSH access from blacklisted IP addresses.	NCSA security team has developed a standard template which is used to ensure similar logging across different operating system. Additionally a customized version of SSH with enhanced monitoring and logging capabilities is used for improved alert generation. Given the large size of the Syslog data, it is only collected from production and critical hosts (defined above), i.e., not from individual desktops/laptops.	Each OS (Windows, Linux, Solaris, AIX) has its own version of Syslog. Some add-on tools ensure symmetric collection of Syslog data.	60	3 GB	11
	File Integrity Monitors	This tool monitors specified system files, e.g., configuration files, operating system binaries and system libraries and alerts are generated when monitored files are added/removed/deleted/created or changed	An OS specific version of file integrity monitor is configured to runs on production and critical hosts (defined above)	A host OS specific version is required on each monitored system.	400	1GB	1
Custom	Google Alerts	Google maintains an index of words found on a webpage it crawls to build its search database and update its cache. It can be configured to send notifications when certain terms (e.g. cheap pharmacy advertisements) are found in the web page. Google alerts are configured to catch spam uploaded by miscreants in comment sections (world writable) of web pages or wikis.	A list of common spam, warez, and drug terms is maintained, and a Google alert system is configured to generate an alert on a successful hit.	This alerting is activated using an external agent.	NA	NA	5

Network flow (netflow) collectors are distributed so that flows are monitored and logged from the border router using Argus and from the internal routers using nfdump. The measurements give visibility to the traffic within the internal subnets (nfdump) as well as to the traffic going in and out of the network (nfdump

[1,2] Dual set up of Argus and nfdump provides redundancy.

+ Argus). In addition to the alerts generated, the data from two flow collectors (stored separately for redundancy purposes) can be valuable in conducting forensic analysis while tracing specific attacks. Clearly, there is certain level of overlap in the data, since nfdump, although focused on internal monitoring, will also have the external data. Additional redundancy is provided by the fact that often two or more internal routers may generate flows for same connection session. Such redundant data collection is valuable under major network DOS attacks.

The central syslog collector has built-in redundancy to execute in a failsafe mode. An event correlation engine uses syslog data to alert on anomalies based on a rule set, such as new user creation, authentication profiles, privilege escalations by new users, and command history profiles. The File Integrity Monitors (FIMs) watch for unauthorized modification/change/creation/deletion of system and configuration files. Limited installation of File Integrity Monitors and the central syslog server is done for performance data collection cost and scalability reasons. In the event of an incident on a host not running the file integrity monitors or forwarding syslogs, one might not be able to catch activity performed by a miscreant on the file system in real time[3]. However, network monitoring can be helpful in tracking incident information, e.g., the attacker's IP address, and possibly specific details about the exploit used and the vulnerability exploited. Thus, monitoring by flow collectors and IDS can somewhat compensate for the selective deployment of FIMs to identify an attack and provide relevant forensic information. For example, both flow collectors and IDS generate alerts based on internal and external network traffic. Syslog data is used to generate alerts based on user behavior anomalies by correlating syslogs and system behavior. This data is further correlated with information such as file system access anomalies. Individually, each of these tools provides important data, but each is limited in scope (by its nature and deployment) in its access to network and system infrastructure and in the quality and amount of data gathered. Only a combination of these tools provides a comprehensive view of the activity inside the network and the systems.

3 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 to analyze and model vulnerabilities. For example, [18] uses data mining techniques to study traffic data during an attack in order to identify signatures of intrusion detection, and [3] uses vulnerability data to develop a finite state machine model for analyzing vulnerabilities and attacks at the code level. [17] analyzes a single denial-of-service attack on a server. Several studies [12] [20] have analyzed attack data to build formal models involving both single and multiple nodes. Several authors have proposed models that correlate alerts to incidents. Two examples include [9], where a capability-based model is proposed and verified using real attack data, and [14], where a prerequisites-and-consequences model is discussed.

Other sources of attack data are honeypot experiments [5], [10], and the DETERlab Testbed (http://www.isi.edu/deter/). Red teams have often been used to collect network vulnerability data, generally unpublished. Several studies focus on classification of security flaws, vulnerabilities, and attacks; examples include [2], [13], and [4]. In [16], 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. Most recently, the Verizon Business RISK Team investigated 90 security breaches in 2009, which include 285 million compromised records [21].

[3] Significant information can be revealed by host and disk based forensics. Incidents discussed in this study had host and disk level forensics performed when required.

4 Analysis of an Example Incident

This section describes the analysis conducted from an alert to a full-blown incident, specifically a credential compromise incident, using data snippets from a real log to walk through the process.

(1) An IDS alert shows suspicious download on a production system (victim: xx.yy.ww.zz) using http protocol from remote host aa.bb.cc.dd (the hostname and IP address are anonymized). The following data snippet walks us through the process of going from an alert to a full-blown incident.

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)	(**1**)

In the above data snippet, ptrat.c is an unauthorized file. This file is suspect because (a) this particular system is not generally expected to download any code apart from patches, system updates, and other binaries, and then only from authorized sources, (b) the downloaded file is a C language source code, and (c) the server this source was downloaded from is not a formal software distribution repository.

The alert suggests the download of a suspicious source file but does not give a context for the events prior to the download (e.g., a login, an executed exploit, an abnormal number of bytes transferred, or a scan). *In other words, 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) Network flows reveal further connections with other hosts. The network flow data described in the following reveal other connections in close time proximity to the download: (i) an ssh connection 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	(**2**)

Time correlations in the flow data shown above also demonstrate that a user login has occurred using the ssh protocol (port 22) in close time proximity to the download. This login could explain the exploit download. *However, the ssh connection record does not reveal (a) whether authentication was successful or (b) what credentials were used to authenticate*.

(3) Manual correlation with syslog alerts. The snippet shown below confirms a user login from *195.aa.bb.cc*, which is unusual, based on the user profile and behavior pattern.

May 16 03:32:27 host sshd[7419]: Accepted password for user from 195.aa.bb.cc port 35794 ssh2	(**3**)

Now we have four data points: (1) a suspicious source code was downloaded, (2) the user login occurred at nearly the same time as the download, (3) it was the first time that user login was from IP address 195.aa.bb.cc, and (4) there was machine communication to other ports (FTP). This is as far we could go with the monitor data. A manual search of all files owned or created by this user found a library file created at the time of the download. This is symptomatic of a footprint left behind by a credential-stealing exploit. *This signature in the /tmp/ folder with the user's ownership confirms that this 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	(**4**)

The additional analysis shows that, in close proximity to the download of ptrat.c, a library file was created in the /tmp directory owned by the user. The library file libno_ex.so.1.0 is known to be created when an exploit code for vulnerability cve-2009-1185 is executed and is successful. The vulnerability (cve-2009-1185) was exploited to obtain root access to the system and to set up the system to harvest credentials. Upon further investigation, *it is determined that the attacker has successfully obtained root privileges in the system and replaced the sshd daemon with a trojaned version that was storing 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*	(**5**)

This example shows that there is an increasing level of suspicion of an imminent attack. We speculate that if we are able to develop techniques to pre-empt the attack action and potentially let it 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 [6].

No single available tool can perform the kind of analysis discussed here, because it requires correlating data from various monitors and system logs with human expertise. Collectively, these analyses can aid in developing new monitoring frameworks and serve to benchmark the efficacy of overarching correlation tools.

5 Describing the Data

In this section, we categorize incidents by 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 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 with respect to 124 actual incidents unless stated otherwise. While there are approximately 140 actionable alerts per day from the NCSA network, we only focus on those that resulted in the discovery of incidents analyzed in this study.

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 successfully exploits one or more vulnerabilities in the system and obtains unauthorized access. A *misuse* is an intentional improper or an unauthorized action performed using computing resources. *Severity* is defined in the terms of the adverse effects of an incident on the operations, assets, or individual with loss of integrity, confidentiality, and availability [7]. Table 2 provides an overview of the alerts (alerts description is given at www.ncsa.illinois.edu/~aashish/incidents) generated by each monitoring tool. The two bottom rows provide the number of alerts corresponding to all actual incidents (*row INC*) and all incident investigations (*row INV*). Table 2 shows that the IDS and Flows monitors detected 31% and 25% of incidents, respectively. Twenty-seven percent of incidents went undetected.

For every incident described in this paper, different 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, (iv) privilege attained by the attacker, (vii) attack phase, and (viii) severity. Table 3 summarizes the characteristics of three sample incidents that we use in subsequent examples. Record 1 corresponds to an *application compromise* incident. In this incident, the attacker obtained root privilege by exploiting the xp_cmdshell MSSQL server vulnerability and used the system as an unauthorized media server (warez). This medium-severity incident was detected in the last phase of the attack, i.e., attack relay/misuse, by TopN, which alerts to an above-predefined traffic threshold.

Table 2. Alert types generated by the monitoring tools

#	IDS Alerts								Flows Alerts				Syslog		FIM	Other	No Alerts				
	Packet/Protocol Analyzers								Traffic Analyzers				Profile		Host		3rd Party Notification				
	IRC	Internal scan	HTTP	Malware	FTP	SSH	Virus/worm	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
INV 150	16	6	7	6	3	1	1	1	**27**	2	13	1	11	4	1	5	4	19	3	1	18

Table 2b. Detail description of alerts

File Integrity Monitors	Integrity Monitor	Change in File System	
Flows	Traffic Profiling	Darknet Flows	Connections to an unused IP block with in the network. No traffic should be sent to these IP blocks. A connection to such address space is usually an indication of an automated scan performed by an attacker or a virus and worm.
		TopN	Alerts are generated when host communication traffic goes about a specified threshold. This threshold can be based on packet count as well as byte counts. A high packet count connection (with few bytes per packet) signifies a possible scan. High byte count is an indication of excessive traffic between the hosts. This may indicate a normal user activity or movie distribution. A host suddenly generating very high byte could need to be investigated.
		Undernet Flows	Undernet is an open public IRC server. Hackers use it to host channels for botnet command and control.
		Watch List	Blacklisted IP address lists distributed amongst security professionals (Garretson 2006)
IDS	FTP Analyzer	FTP Sensitive	A list of well-known exploits, rootkit and malware and their provider domain names is maintained that triggers an alert on a match.
	HTTP Analyzer	HTTP HotClusterConn	Alerts are generated on the behavior property of servers and workstations. This alert is raised if a host is downloading any file when its generally not expected to download. Example: A web server or a mail server, certification authority or authentication server etc. is not expected to download any files except for patches and system updates from authorized sources.
	HTTP Analyzer	Bro: HTTP_Sensitive	A list of well-known exploits, rootkit and malware and their provider domain names is maintained that triggers an alert on a match
	IRC Analyzer	Bro: NewIRCConn	Botnet controllers prefer to communicate with their bots via IRC protocol using unconventional ports. Bro IDS has dynamic protocol analyzers (Holger Dreger 2006). These alarms are triggered when a suspicious connection is seen.
	Malware Analyzer	Bro: HTTP_Malware	Various organizations provide malware hash registry provides pre-computed MD5 and SHA1 hash for known identified malware. IDS malware analyzer monitors binaries downloaded in the network, computes its hash and triggers an alarm if a match is found with the malware registry.
	Traffic Profiling	Scan Summary	A scan gives attackers information about the infrastructure running with in the network. Most scans are may seem innocuous or noisy. A daily summary is useful to highlight what kind of vulnerability attackers are trying to exploit. An internal system scanning is usually an indication of a compromised host. Daily bruteforce SSH report is a summary of previous days bruteforce ssh attempts and an alert is generated if connection is successful.
		Scan: Address Scan	
		Internal Scan	
		SSH Bruteforce report	
	Virus Signature	Virus/Worm Spread	Signatures written specifically to catch automated propagation of virus and worms.
None	Notification	Mailing List	Various Information sharing and analysis groups (REN-ISAC, CERT, FIRST] transmit notifications when a malicious activity originates from our network.
		Watchlists	Watchlists are blacklisted IP addresses distributed within the community. Any connection to such address is suspicious and triggers an alarm.
		External Notifications about Infection	External sources notify security team when they notice or attribute the origin of malicious activity or abuse to our network.
		Administrators	Notification from administrators is received when they find something suspicious or anomalous with their system. An example: When they notice an unexplained account created on the system or if hard disk is suddenly full and no valid explanation is found.
		Peers	A peer notification highlights a possible security issue may have an adverse effect on our network because of shared resources or users. An incident on a partner site, for example, may substantially affect our site.
		User Request	
Syslog	User Profiling	Command Anomaly	User characteristics such as Login times, authentication used, and hosts connected to and from, last activity, commands executed is used to build a profile and alerts are generated if anomalies are detected.
		Login Anomaly	
Custom	HTTP Analyzer	Google Alert / Notification: Email External	Google maintains an index of words found on a webpage when it is crawling to build its cache. It can be configured to send notifications when certain terms are found in the web page. Google alerts are configured to catch spam upload by miscreants in comment sections (world writable) of web pages or wikis.
	Vulnerability Scan	Security Checkups	Proactive scanning for vulnerability on the system inside the network can be a useful alert to find potentially vulnerable and exploitable hosts.

Table 3. 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

6 Classifying and Analyzing the Incidents

An accurate categorization of incidents is essential to guide a recovery strategy for an incident, e.g., 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 reinstallation of the system and mandatory change of user passwords. This section classifies each incident in our data in three ways: *incident type*, *attack phase*, and *severity*. The incident type classification bins incidents into nine broad categories based on 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 phases [16] and provides an important measure of the effectiveness of monitors. The severity measures the impact of an incident on the computing enterprise.

Incident type. Table 4 provides a detailed breakdown of incident types, the vulnerability exploited, the misuse, and alerts generated, along with their counts. The set of analyzed incidents includes 22 successful attacks involving applications such as VNC, MySql, MSSQL, and OpenSSL. Since all these compromises share similar characteristics, they 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 in the *credential compromise* category. Additionally, credential compromise incidents were detected by six types of alerts. The user profiling alert detected 11 incidents, 12 incidents of identical attack pattern were discovered by HTTP, FTP IRC, and Watchlist alerts, and 9 incidents were not detected by any alert and were determined by the third party notifications.

Table 4. Incident categories

Incident Category (Count)	Vulnerability/Exploits (Count)	Incident Compromise Specifics (Count)	Alert Generated (Count)
Credential compromise (32) User credentials are targeted and stolen. Attack propagates by using stolen credentials and local root escalation exploits.	Stolen password/key-pair (31) Open-X11 keystroke logging (1)	Root (rootkit + trojan ssh/sshd) (7), User (key-pair/certificate) (21) Spam (1), Bot (1) , Scan NFS file system (2)	FTP Analyzer (3), HTTP Analyzer (3), IRC Analyzer (1), Notification (9) , Watchlist (5), User Profiling (11)
Web server/application (22) Web servers (e.g. IIS or Apache) and/or web applications (e.g. phpmyadmin or wiki) compromises	PHP Remote command execution / code Injection (11), Web Server Misconfiguration (7), Unknown (3), IIS permissions (1)	Defacement (5), Scan other hosts (5), Spam (4), Backdoor (3), Bot (1), Malware (1), Open Proxy (1), Un-auth Ftp server (1), Incorrect permissions (1)	HTTP (1), IRC (1), Darknet (1), Scan (2), TopN (4), Watchlist (1), Notification (8), Google Alerts (3), Malware (1)
Application compromises (22) Compromise of application level software (e.g. VNC, OpenSSL, mysql, etc)	Unknown (6), VNC exploits (6), Mysql exploits (2), Telnet exploit (2), Rlogin exploit (1), SSL exploit (2), OpenX11 exploit (1), WINS exploit (1), MS-SQL exploit (1)	Warez (10), Scan (5), Backdoor (3), Bot (1), IRC Bouncer (1), Sniffer (1), Unknown (1)	TopN (8), IRC (5), Notification (3), Scan (3), Watchlist (1), User profiling (1), HTTP (1)

Bruteforce SSH (20) Attackers bruteforce to guess password for well known accounts (e.g. root/guest/test) for SSH	Weak Passwords (19), Misconfiguration (1)	Bot (9), SSH Bruteforce Scan (7), IRC Bouncer (2), root exploit (2)	IRC (8), TopN(3), Watchlist (4), User Profiling (2), Notification (2), Scan (1)
SPAM/phishing (11) Incidents resulted in sending spam and phishing emails.	Unknown (6), Link click (3), Misconfiguration (2)	Spam (8), Web spam (2) , Malware distribution (1)	HTTP (5), Notification (4), Watchlist (1), Google Alerts (1)
Infected system (11) A host in the network is infected with a virus or a known malware.	Unknown (8), Confiker (1), W32.Welchia.Worm (1), W32/SDBot (1)	Scan (7), Spyware (1), Bot (1), IRC Bouncer (1), DDoS (1)	Notification (5), Scan (4), Watchlist (1), TopN (1)
Pre-infected hosts on the network (3) Systems infected outside the network boundaries and brought back into the network.	Unknown (3)	Scan (3)	TopN (1), Notification (2)
Social Engineering (2) Attempts to manipulate users to divulge critical system information such as passwords.	Instant Messenger (1), Unknown (1)	Privilege Escalation (1), ID theft (1)	Notification (2)
Insider (1) A policy violation by an action of an internal employee/user.	Insider	Unauthorized privilege retain	File Integrity Monitor (1)

Observations from Table 4**:**

- A wide variety of alerts are associated with each incident. Each alert detects multiple incidents, suggesting a commonality in attack paths.
- The majority of incidents (55%) are due to attacks on authentication mechanisms with varying levels of sophistication, e.g., password guessing (bruteforce ssh), password stealing (credentials compromise), exploiting a vulnerability (VNC null session), installing trojaned versions of SSH and SSHD to sniff passwords and target public-private key pairs.
- Attackers deploying unauthorized media (warez) are more sophisticated at hiding their trails than are other attackers.
- Web-based attacks are primarily a result of misconfiguration (7) or web-application vulnerability (10). However, in terms of misuse, attackers were seen establishing backdoors or scanning hosts and file systems instead of merely defacing.
- Application compromises, spam/phishing (6 incidents each) and infected systems (8 incidents) categories combined together have 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 is so broad that most entry points are difficult to establish.

Incident phases. Next, using the attack stages proposed by [16], alerts for all incidents are mapped into seven phases: Scan (1/1), Breach (30/39), Penetration (9/10), Control (21/23), Embedding (8/9), Data Modification (7/7), Attack-relay/misuse (48/61). The numbers in parentheses are the (actual incident/ investigation) counts for each phase in which the alert was observed. A useful metric to study the effectiveness of a detection mechanism is the phase of the attack in which the alert originated.

Incident severity. A reason to categorize incidents based on severity is to determine whether monitors are catching harmful incidents as opposed to low-impact compromises. Table 5 provides definitions of the various degrees of severity and the distribution of the analyzed incidents across the severity categories.

Table 5. Incident severity

Severity	Effect on Loss of Integrity, Availability, and Confidentiality	Incident Impact	# 124
Very High	Catastrophic	Entire organization/Credentials compromise 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 allows attacker to obtain root level privileges on the systems	37
Medium	Limited	Users and small cluster system /Affects entire research group / Application level compromise (VNC, Solaris Telnet worm, XP_Cmdshell mssql exploit), web server (Phpmyadmin, Php Horde, awstats), Malware hosting	24
Low	Little or no effect	Non-production systems / Affects individual / Bruteforce SSH password, Infected Systems, SPAM/Phishing emails, Web server Spam with cheap pharmaceutical	62

7 Discussion of Findings

Detection latency. This is defined in terms of the attack phase in which detection occurs, regardless of time. Nearly 50% (61/124) of detected incidents were discovered in the very last stage of the attack, i.e., in the attack-relay/misuse phase. This 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 detected 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%.

Detection latency and severity. Out of 61 incidents detected in the attack-relay/misuse (last) phase, only 30% were of medium to high severity. While one expects a high level of correlation between high-latency of detection and high-severity (the longer the attacker stays in the system more potential 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. In fact, 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 in Table 3, 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.

Top five alerts. Table 2 and Table 4 show the different incident types caught by each alert. The top five alerts, which account for detection of approximately 54% (67/124) incidents, include: *TopN* (18 incidents, 6 types), *IRC* (15 incidents, 4 types), *Watchlist* (11 incidents, 6 types), *login and command anomaly* (14 incidents, 5 types), *HTTP and FTP analyzer* (9 incidents, 4 types). Observe, in Fig.1, that each alert detects multiple incident types, which suggests that while the exploits may be different, the incidents share a common attack path.

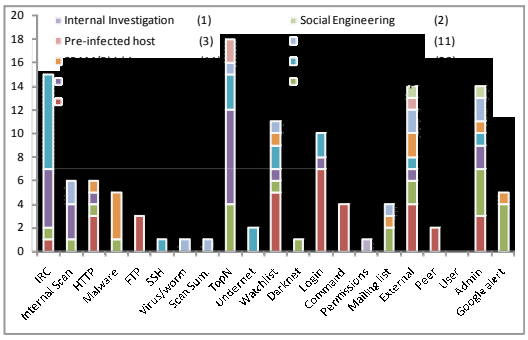

Fig. 1. Alert distribution based on incident

The TopN alert, which is triggered when network traffic for the host crosses a certain threshold, caught the maximum number of incidents (18). Using 150 incident investigations as a basis, this alert exhibits a high false positive rate (9/27 = 33%), and the detected incidents are of low to medium severity. This is a somewhat counterintuitive because one would expect that an alert that detects the large number of incidents would also cover a proportional number of high-severity incidents.

The IRC alert caught 15 incidents while exhibiting a very low false positive rate (1/16 = 6%). This is because the IRC alert uses a highly specialized detection mechanism finding connections to command and control channel derived based on the characteristics of known attacks. HTTP and FTP analyzer alerts caught six high severity incidents in the early stages of the attack (penetration phase), when attackers are trying to download an exploit on a host. This alert is generated when a known exploit signature is matched or if an attacker is downloading exploits directly from publicly available exploit repositories. This is a high maintenance alert and requires that a list of exploit signatures to be constantly updated.

The Watchlist alert is triggered in the breach phase when an attacker connects from a known bad IP address. This alert caught 11 incidents (and one false positive incident) of 6 different types. Four of those incidents were of high 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. 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.

8 Data-Driven Modeling of Incidents

Our analysis of security data demonstrates that it is feasible to identify actions performed by an attacker and to construct a model that captures attacker behavior at the system and the network levels. Such a model can: (i) provide a way to reason about the attack independently of the vulnerabilities exploited and (ii) can assist in reconfiguring the monitoring system (e.g., placing new alerts) and adapting detection capabilities to changes in the underlying infrastructure and the growing sophistication of attackers.

Recent studies show that modeling can be useful in understanding attack patterns and building more efficient protection mechanisms. For example, [3] 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, [8] 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.

Our model is derived from a large sample of real data on security attacks, and a single finite state machine captures all incidents from the same category. Building the models proceeds in the following steps: *Step1* – identification of all alerts associated with a given category (as defined in Table 4) of incidents, *Step2* – identification of distinct system/application states traversed by the attacker in penetrating the network, and *Step3* – assigning conditions (usually) to transitions between the system states identified in Step2.

Fig. 2(a) depicts (in a form of a flow diagram) all the alerts involved in detecting credential compromise incidents (32 incidents in our data). In addition, Fig. 2(b) provides steps in alert processing to illustrate how we associate alerts with the incidents. For example, in processing an alert, triggered by the HTTP, FTP, IRC analyzer of the IDS monitoring tool, (see Fig. 2(b)) we note the following: (i) the HTTP analyzer of the IDS alerts to the download of an unknown file to the system, (ii) by default, the system, an HTTP server, is not expected to download source code, and (iii) manual file inspection is necessary to determine that the adversary exploited a compromised user account, i.e., the attack resulted in a credential compromise.

Fig. 2(b) shows the state machine model for credential compromise incidents. In the state machine, each node depicts a state of the system. The arcs (transitions) are the actions taken by a genuine user or an attacker, or checks/alerts enforced by the system. Nodes marked with dotted lines correspond to (probation) states traversed by the system in early stages of an attack when routine system checks (e.g., check for the IP address) are enforced to detect symptoms of potential malicious activities (e.g., login from unknown IP address). Some transitions can be performed in multiple ways, e.g., an attacker can use different exploits to achieve his/her goal. A specific alert is associated with each transition.

A possible use of the model. Based on specific alerts, we allow user to execute at increased levels of probation. The follow up transitions indicate places where to introduce additional monitors to scrutinize user actions more closely and pre-empt an imminent damage. For example, in Fig. 2(B), the *exploit download*

transition in the FSM model could be preempted based on login-profile-anomaly alerts. File integrity monitor then, can watch user actions more closely and preempt potential malicious activities, e.g., execution of binaries by the suspected user. As shown in Fig. 2(a), there is redundancy in the deployed alerts to compensate for the imperfection of monitoring tools and hence, to improve overall coverage.

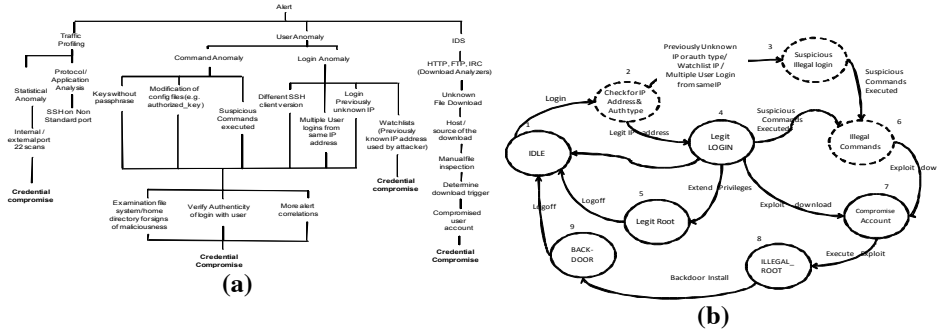

Fig. 2. Credential compromise incidents: (a) association of alerts with the incident; (b) state machine model

9 Incidents Missed by the Monitors

Due to the imperfection of monitors, 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. The objective is to identify inherent limitations in the current security monitoring set up and guide development of enhancements in attack detection and response. All incidents missed by the monitoring system are discovered thanks to the 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. Possible reasons for failure in detection are analyzed and the monitoring system is enhanced or reconfigured if feasible. In many cases there is no immediate resolution, (e.g., availability of a signature for the undetected attack) or an available resolution is too limited or too generic in scope (i.e., generates false positives). Table 6 shows the detailed breakdown of false negatives across the incident categories.

Table 6: Incidents missed

Monitor Tool	Incidents Discovered	Full Investigation with No Incident	Credentials Compromise	Web Server Compromise	Infected System	SPAM Abuse/phishing	Application Compromise	Bruteforce ssh	Pre-Infected Host	Social Engineering
Notification	45*	11	10	6	5	4	3	2	2	2
	External (19)	Compromise accounts (5)	Root compromise (3)	Phpmyadmin (2)	Bots (2)/ confiker (1)	Spamming host (1)	VNC (1) / Open SSL exploit (1)	Weak password		Admin other site (1)
	Mailing list (4)			Malware on web server (1)		Spamming host (2)			Virus infection	
	Peer (3)	User credentials (1)	User account (3)							
	Admin (17)	Hardware (3) / file system (1)	Root (1) / user (3)	Win2003 Server (1)/php (1)	Windows Trojan (2)	Spam on web page	VNC	Misconfig* Root to all logins	Virus infection	Admin (1)
	User (1)	Request to check system								
	* 1 Internal investigation									

The analysis of the data on missed incidents reveals inherent limitations in the 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 wide spectrum of attacks, malware, and network traffic. The following discussion illustrates these limitations using examples of incidents missed by the monitoring system.

Inability to produce contextual information on what is normal and abnormal in a stream of observed events. About 22% of the missed incidents are credentials compromises (a high impact category). For

detection of incidents in this category, monitors rely on alerts based on detection of: (i) deviation in user behavior compared with the known user profile (using syslog), (ii) malicious code download (using IDS), and (iii) unexpected system file manipulation (using File Integrity Monitor). Combination of these three tools should provide high detection coverage. However, detecting a multi-step attack that uses stolen credentials requires comprehensive runtime traffic analysis, including correlation of different events and an accurate determination of what is normal and abnormal in the observed traffic (the detailed investigation example in Section 2 illustrates this point). 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 to the system; (ii) *IDS* does not raise alerts when attackers do not download malware form a known source, and often there is no built-in signature for a given exploit; (iii) due to the 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 pertinent information In credential stealing attack, during phase the attacker on the system downloads malware. Assuming that IDS is updated with a signature this malware (or exploit code), it should generate an alert. However, due to lack of context, it still is a process to determine what user account was used for download of this malware. Things are even harder to track-down when attackers delete the malware from the directory. Ideally, correlation of file integrity monitor data, IDS and syslogs should be able to build an accurate event timeline. Currently each of the monitoring tools lack this capability.

Inability to cope with a wide spectrum of attacks, malware, and network. About 7% of the missed incidents are application compromises. This limited detection coverage is due to the lack of timely available signatures and the emergence of new zero-day exploits. For instance, detection signatures for compromises due to VNC null string authentication by-pass vulnerability (CVE-2006-2369), OpenSSL SSL-Get-Shared-Ciphers Buffer overflow exploit (CVE-2006-3738), and WINS Associate context vulnerability (CVE-2004-0567) were unavailable at the time of the attack.

Infected host incidents (about 11%) mostly occur due either to user accidentally downloading malware and installing on the system or to virus propagation. A user accidentally downloading and installing malware may have a very small footprint to generate alerts based on syslog or FIM. Therefore, the detection of such incidents relies mostly on IDS and netflows. Ideally, good antivirus software running on the host system should 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 into clicking on a link or opening email attachments) and/or exploiting browser vulnerabilities. Also, malware is designed specifically to evade antivirus 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. As an alternative, approaches such as control flow detection have been proposed to address this issue

Web compromises constituted about 13% of incidents missed by the monitoring system. Most of the web compromises are due to exploitation of web application software, e.g., exploitation of PHP vulnerabilities allowing attackers 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 post-attack behavior of the system, especially when attackers are uploading static content (e.g., spam pages, redirections of links, and web page defacements). Although HTTP content monitoring tools can flag such content in traversing the network, 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 with no observable change in the system. This reveals the need for monitoring outgoing traffic as well.

10 Conclusions

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

- IDS and Flows monitors detected 31% and 25% of incidents, respectively, while 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 7 times more likely to capture an incident than are signature-based detectors. This is because signatures are specialized to detect the presence (or download) of a known malicious binary. Consequently, they can be subverted. Signature-based detectors (due to their specialization) have fewer false positives compared to anomaly-based detectors.
- Nearly 50% of the incidents are detected in the last stage of the attack, i.e., attack-relay/misuse. This kind of detection often comes too late, after damage to the system has already occurred.
- There is no indication of a high-level correlation between latency in incident detection and incident severity, e.g., incidents detected in the last stage (long latency detection) of an attack are not necessary high-severity incidents.
- Alerts detecting the most incidents may not be the most efficient, e.g., TopN alert detects 15% (18/124) of the incidents of low to medium severity, but it exhibits a 33% false positive rate.

While in this analysis we considered 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 of the monitoring system and adapting of detection capabilities in response to changes in the underlying infrastructure and the growing sophistication of attackers.

References

1. 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).
2. Bellovin, S. R., Cheswick, B.: Firewalls and Internet Security: Repelling the Wily Hacker. Addison-Wesley Publishing Company (1994).
3. Chen, S., Kalbarczyk, Z., Xu, J., Iyer, R. K.: A data-driven finite state machine model for analyzing security vulnerabilities. Proc. of the 2003 Int'l Conference on Dependable Systems and Networks, pp. 605-614 IEEE (2003).
4. Cohen, F. B.: Protection and Security on the Information Superhighway. John Wiley & Sons, New York (1995).
5. Cukier, M., Berthier, R, Panjwani, S., Tan, S.: A statistical analysis of attack data to separate attacks. Proc. of the Int'l Conference on Dependable Systems and Networks, (2006).
6. Cutts Jr. et al, United States Patent 5,193,175, March 9, 1993
7. DOE M-205: Cyber Security Incident Management Manual. Department of Energy (2010).
8. Gregorio-de Souza, I., Berk, V. H., Giani, A., Bakos, G., Bates, M., Cybenko, G., et al.: Detection of complex cyber attacks. Proc. of the SPIE. 6201, pp. 620-106. (2006).
9. 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) pp. 1–31 (2007).
10. Kendall, K., Smith, A. C.: A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems. MIT, Electrical and Computer Engineering. Cambridge (1999).
11. Kumar, S, Spafford, E.: An Application of Pattern Matching in Intrusion Detection. Purdue University, Tech. Report, Dept. of Computer Sciences (1994).
12. Kumar, S.: Classification of intrusions. Purdue University. (1995).
13. Landwehr, C. E., Bull, A.R., Mcdormott, J.P., Choi, W.S.: A taxonomy of computer security flaws. ACM Computing Surveys, 26 No. 3, pp. 211-254 (1994).
14. Ning, P. and Xu, D.: Learning attack strategies from intrusion alerts. Proc. of the 10th ACM Conference on Computer and Communications Security (2003).
15. Paxson, V.: Bro: A system for detecting network intruders in real-time. Computer Networks, (pp. 2435-2463). San Antonio (1999).
16. Ruiu, D.: Cautionary Tales: Stealth Coordinated Attack How-to (1999). http://althing.cs.dartmouth.edu/secref/local/stealth-co-ordinated-attack.txt (1999).
17. Sung,M., Haas, M, Xu,J.: Analysis of DoS attack traffic data. FIRST Conference. Hawaii. (2002).
18. Tidwell, T., Larson, R., Fitch, K., and Hale,J. Modeling internet attacks. Proc. of the 2001 IEEE Workshop on Information Assurance and Security, (2001).
19. Templeton, S.J., Levitt, K.: A requires/provides model for computer attacks. New Security Paradigm Workshop (2000).
20. Treinen, J., Thurimella, R.: A framework for the application of association rule mining in large intrusion detection infrastructures. Proc. of the 9th Int'l Symposium on Recent Advances in Intrusion Detection, 4219 (2006).
21. Verizon Business Risk Team: 2009 Data Breach Investigations Report, http://www.verizonbusiness.com/resources/security/reports/2009_databreach_rp.pdf, (2009).