Analysis of Credential Stealing Attacks in an Open Networked Environment

A. Sharma[1,2], Z. Kalbarczyk[1], R. Iyer[1]
[1]Coordinated Sciences Laboratory
University of Illinois at Urbana-Champaign
Urbana, USA
{aashish, kalbarcz, rkiyer}@illinois.edu

J. Barlow[2]
[2]National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign,
Urbana, USA
{jbarlow}@illinois.edu

Abstract. **This paper analyses the forensic data on credential stealing incidents over a period of 5 years across 5000 machines monitored at the National Center for Supercomputing Applications at the University of Illinois. The analysis conducted is the first attempt in an open operational environment (i) to evaluate the intricacies of carrying out SSH-based credential stealing attacks, (ii) to highlight and quantify key characteristics of such attacks, and (iii) to provide the system level characterization of such incidents in terms of distribution of alerts and incident consequences**.

Keywords-Incident analysis;Credential stealing; Intrusion detection

I. INTRODUCTION

Credential stealing attacks are an important security threat which can lead to a widespread penetration of critical network infrastructures owned by both academic institutions (university and/or research laboratories) and industry (e.g., banking) [7]. In most of these events, an attacker usually masquerades as a legitimate user and exploits system vulnerabilities to escalate privileges to root in order to steal (harvest) more credential.

The grid computing community has become one of the preferred targets because of its diverse user base and extensive use of SSH [5]. However, system compromises due to stolen credential are not just limited to academic and research institutions. In the fall of 2009, the Apache Foundation's main site was compromised when miscreants obtained unauthorized access using stolen SSH keys [9]. This incident raised concerns about the integrity of Apache software. The frequency of credential stealing incidents increased in 2008 when the Debian Linux distribution's predictable random number generation bug (CVE-2008-0166) surfaced. This bug resulted in the generation of easily guessable cryptographic keys (e.g., SSH keys, OpenVPN keys, and session keys used in SSL/TLS connections) on the Debian platform [18]. The widespread nature of the credential stealing attacks resulted in a US-CERT advisory [2]. Furthermore, social engineering (e.g., Facebook), mailing systems (e.g., Gmail), and financial institutions have been plagued with the problem of compromised accounts for quite some time.

Most of the existing studies analyze the problem of stolen credentials in the context of the underground economy (e.g., [16]) and the exposure of the end user to such attacks (e.g., security of internet banking [7]). Relatively few publications discuss protection mechanisms; those that do frequently conclude that "protection against all credential - stealing attacks is next to impossible" [7].

This paper reports on the data-driven study of credential stealing incidents in an open, networked environment. The analysis is based on the forensic data on compromised accounts collected over a period of 5 years across 5000 machines monitored at the National Center for Supercomputing Applications at the University of Illinois. Our initial investigation of a broader spectrum of security incidents at NCSA indicated that nearly 26% (32/124) of the incidents analyzed involved credential stealing [12].

The goals of the current study are (i) to evaluate the intricacies of carrying out a credential stealing attack, (ii) to highlight and quantify key characteristics of such attacks, and (iii) to provide the system-level characterization of such incidents in terms of distribution of alerts (responsible for detecting the incidents) and incident consequences.

Key findings can be summarized as follows:

- In all but one case attackers came into the system with a valid credential of an NCSA user account (31 out of 321 incidents) and thus could not be stopped at the network boundary. Attackers rely on their access to an external repository of valid credentials to harvest more credentials.
- Availability of valid credentials makes boundary protections (e.g., reliance only on a firewall) insufficient for this type of attack. Therefore, higher scrutiny in monitoring user actions is required, e.g., introduction of schemes such as execution under tight scrutiny after detecting a first suspicious event (e.g., logging from an unknown host).
- There is a need for system-wide correlation between alerts generated by different monitoring tools to improve detection coverage. About 28% (9/32) of credential stealing incidents were still missed by the monitors, i.e., none of the monitoring tools raised an alert and an incident was discovered due to external notification.

[1] There is another large-scale incident (known as MC-216) where attackers used valid credentials to compromise hosts in numerous institutions including NCSA. This is not reported in our work. Due to the magnitude of the incident, it is a significant study in itself.

- Sharing information (among trusted peers) on attacks and/or attackers is essential in detecting attacks. 20 out of 32 incidents were discovered because of data sharing among peers. For example, (i) five incidents were detected by *watchlist* alert, which watches for logins from unknown IP addresses; (ii) *HTTP* and *FTP analyzers* (which use exploit download repositories) detected three incidents each; and (iii) nine incidents were caught due to *external notifications* from peers.
- Attackers follow a similar sequence of steps in conducting these attacks, thus making it feasible to construct a model that captures attacker behavior at the system and network levels. Such a model can (i) provide a way to reason about the attack independently of the vulnerabilities exploited and (ii) 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.

II. RELATED WORK

While many techniques to protect against attacks are available, relatively little has been published on measurement-based analysis of different types of attacks and characterization of the detection capabilities of the security monitoring system (tools) in real production settings. Databases like those provided by CERT (www.cert.org) and CVE (cve.mitre.org) document vulnerabilities and possible exploits. Other sources of attack data are honeypot experiments ([14], [8]) and dedicated test-beds, e.g., DETERlab Testbed (http://www.isi.edu/ deter/). Red teams have often been used to collect network vulnerability data, but this is generally unpublished.

While a thorough analysis of credential stealing and its impact has been performed by [16], it is mostly limited to identity theft (bank accounts or credit cards). [20] provides an algorithmic approach to detect multi-hop attacks. [3] provides measurements and analysis on the economics of credential stealing in underground economies.

Schemes such as dual-factor [3] and "site key" authentication [1] have been proposed to deter credential stealing, but neither of these is very effective, especially in the command-based authentication used in SSH. [11] discusses the use of advanced statistical techniques on timing information collected from the network, from which the eavesdropper can learn significant information about what users type in SSH sessions. [5] presents the details about the pervasiveness of attacks on SSH-based credential stealing. [10] postulates a potential worm and the means to automate SSH based attacks. While both [5] and [10] present an aspect of the SSH credential stealing attack, neither goes into the details of how it can be detected.

III. DATA SOURCE

Data analyzed in this study pertain to credential stealing incidents and are extracted from a large set of security incidents that have occurred over a period of 5 years at the National Center for Supercomputing Applications at the University of Illinois. We conducted the initial analysis [12] of 150 incident investigations (resulting in 124 actual incidents and 26 false positives, where a false positive is synonymous with no incident found) to characterize both the attacks and the corresponding alerts that led to incident discovery. Of these 150 incidents, 32 incidents were categorized as credential stealing.

For every incident described in this analysis, data logs produced by the monitoring tools (IDS, Netflows, Syslog, and File Integrity Monitors) are used to determine: (i) the incident type, (ii) the 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), and (iii) the incident detection latency and severity.

Figure 1 shows the distribution of alerts that led to the discovery of 150 incident investigations in our dataset. The number in brackets provides the total number of incidents caught by that alert. The major findings from our analysis [12] are as follows:

- The majority of incidents (55%) were due to attacks on authentication mechanisms with varying levels of sophistication, e.g., password guessing (bruteforce SSH; 20 incidents), exploiting vulnerabilities (e.g., VNC null session exploit; 17 incidents), or installing trojaned versions of SSH and SSHD to sniff passwords and target users' public-private key pairs (credential stealing; 31 incidents). In all 20 cases of successfully guessed passwords, the attacker's objective was to misuse the compromised machine for malicious purposes different from harvesting more credentials. For example, attackers tried to use a compromised node as a bot in a larger network of infected machines.
- 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. In this paper we show that similarity is even stronger for credential stealing attacks.
- Anomaly-based detectors are seven times more likely to capture an incident than are signature-based detectors [12]. This is because the signatures are specialized to detect the presence of a known malicious binary. Consequently, they can be easily subverted. The signature-based detectors have fewer false positives compared to the anomaly-based detectors. While this finding is true for all (124) incidents combined, we observe that for credential stealing incidents, the alerts are equally distributed between the anomaly- and signature-based detectors.

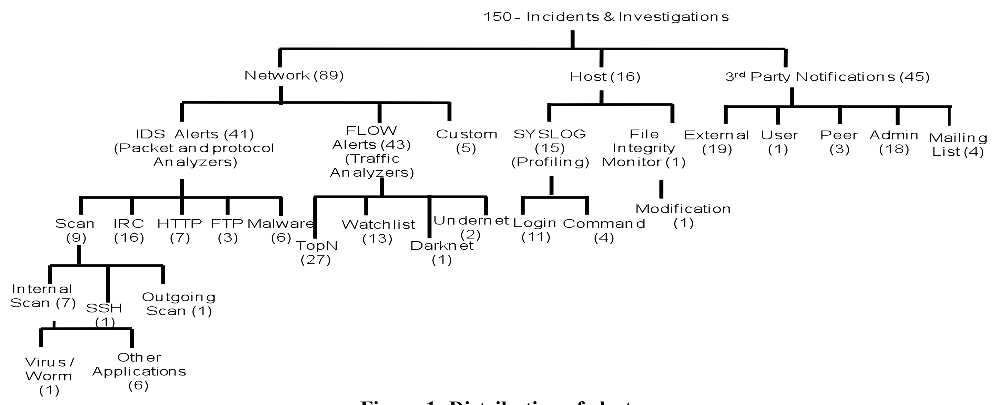

Figure 1: Distribution of alerts

IV. EXAMPLES OF CREDENTIAL STEALING INCIDENTS

This section presents examples (along with detailed investigation steps) of four credential-stealing incidents taken from our dataset. The examples are chosen to highlight (i) *step-by-step forensic investigation procedure* in discovering credential-stealing incidents (Example_1); (ii) *sophistication of attacks and attackers* using specialized rootkits to evade detection and remaining dormant for a very long duration to evade detection (Example_2); (iii) *use of various evasion techniques to collect sniffed passwords from a victim machine (missed incident)*, i.e., installing trojaned versions of SSH, SSHD, and other authentication utilities (e.g., pluggable authentication modules such as pam and sudo) (Example_3); and (iv) *hiding the attacker's tracks*, i.e., hopping from host to host in order to hide the attack traces in the system to make tracing the attacker's path very difficult (Example_4).

A. *Example_1: Analysis steps from IDS alert to credential stealing incident*

This example describes (using data snippets from real logs) the analysis flow from an alert to an identification of a credential compromise incident.

(*1) An IDS alert is triggered via a signature match indicating a suspicious download* (victim: xx.yy.ww.zz) using http protocol downloaded a file called sudo.tgz from remote host aa.bb.cc.dd (the hostname and IP address are anonymized. Since credential stealing incidents often originate from a known peer site, both source and desitnation IP's are anonymized).

```
Nov  23 18:52:21 xx.yy.ww.zz >  aa.bb.cc.dd
GET /..0/sudo.tgz (200 "OK" [737254] server6.bad-domain.com)
```

In the above data snippet, sudo.tgz is an unauthorized file. This file is suspect because (a) sudo source code is not expected to be downloaded using http and (b) the host from which it was downloaded is not a valid Linux distribution repository.

Further investigation revealed additional successful downloads from the local host prior to the alert generation (victim: xx.yy.ww.zz) from the same remote server:

```
Nov 23 16:16:09 xx.yy.ww.zz > aa.bb.cc.dd
GET /..0/vm.c (200 "OK" [6293] server6.bad-domain.com)

Nov 23 16:16:36 xx.yy.ww.zz > aa.bb.cc.dd
GET /..0/vm64.c (200 "OK" [7646] server6.bad-domain.com)
GET /..0/vm64 (200 "OK" [645921] server6.bad-domain.com)
```

Vm64.c is suspected to be source code while vm64 is a compiled binary. While the generated alert suggests the download of a suspicious source file, it does not give a strong context to the events prior to the download (such as a login, an execution of 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) Confirmation that SSH has been replaced: An Nmap probe of the victim host reveals a different version of SSHD (as compared with the proprietary version of SSHD installed on NCSA hosts) running on the system:

```
nmap -sV -p 1-65535 victim-hostname
Interesting ports on victim-hostname.ncsa.uiuc.edu (xx.yy.ww.zz):
Not shown: 65534 filtered ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 4.3p2 (protocol 2.0)
Nmap done: 1 IP address (1 host up) scanned in 65652.33 seconds
```

On looking at the logs we observe that SSHD process has restarted around the same time when the exploit was downloaded (see below).

```
Nov 23 16:51:48 victim-host sshd[9409]: Received SIGHUP; restarting.
```

At that point we have confirmation that (a) a download of known malicious source code was performed, (b) potential exploit code (binary) was also downloaded, and (c) the version of SSHD has changed on the host. All three indicators confirm that attacker was able to successfully execute code and obtain root on the system (since replacing

SSHD requires root privileges). However, we still don't know how attacker was able to get into the system.

Time correlations in the flow data (not shown here) reveal that a user login has occurred using the SSH protocol (port 22) in close 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 pp.qq.rr.ss. Additionally, this login activity is anomalous as compared with the known user profile behavior pattern.

```
Nov 23 16:14:35 victim-host sshd[15087]: Accepted password for user
from pp.qq.rr.ss port 42844 ssh2
```

Now we have four data points: (1) suspicious source code was downloaded, (2) version of sshd has changed, (3) the user login occurred at nearly the same time as the download, and (4) it was the first time that user logged in from IP address pp.qq.rr.ss.

In order to confirm whether the download exploit was successfully perpetrated on the host, the recovered binary was executed on the victim system

```
-bash-3.1$ /mnt/vm64
-----------------------------------
 Linux vmsplice Local Root Exploit
 By qaaz
-----------------------------------
[+] mmap: 0x100000000000 .. 0x100000001000
[+] page: 0x100000000000
[+] page: 0x100000000038
[+] mmap: 0x4000 .. 0x5000
[+] page: 0x4000
[+] page: 0x4038
[+] mmap: 0x1000 .. 0x2000
[+] page: 0x1000
[+] mmap: 0x2aaaaaaac000 .. 0x2aaaaaade000
[+] root
```

This confirms that attacker successfully gained access to the system by logging in with a stolen credential and was able to obtain root privileges using Linux vmsplice local root escalation exploit (CVE-2008-0600).

B. *Example_2: Attacker remains dormant in the network*

In this incident attackers used stolen credentials and a local root escalation exploit to compromise a small research cluster. An IDS alert was generated when a malicious exploit signature matched for a download on one node. Detailed analysis revealed that this attack resulted in a compromise of 10 nodes, including the administrative system. The postmortem investigation found a rootkit installed by the attackers, who only targeted the administrator's system even though they were successful in attaining root privileges on all of the research cluster nodes. The rootkit was found during the forensic analysis of the administrator's system disk image. Traditional rootkit detectors (chkrootkit and rootkit hunter) failed to detect the installation of this rootkit. Further analysis determined that attackers installed the Phalanx rootkit, which characteristically initiates a connection from the victim host back to the attacker system upon receipt of a trigger string. This makes the Phalanx rootkit highly malicious, because the initial trigger string connection is discarded by most of the monitors as a random single tcp scan on port 22 (the attacker never established a full connection). At the same time, most of the current monitors are concentrating on incoming traffic and miss the connection initiated by an internal host to the outside system. Thus, this backdoor method provides nearly perfect cover for an attacker to evade detection.

The incident was discovered with the help of user profiling login anomalies. Typically, high-numbered ports are used for data transfer sessions (e.g., FTP or secure file copy), and these connections are not often monitored by the IDS since deep packet inspection for high-volume data transfer sessions is fairly expensive.

Initial compromise.

```
[20081006-host-1]$ grep 69.129.rr.ss secure*
secure.3:Sep 21 03:04:00 host-1 sshd[15943]: Failed password for user-2
from ::ffff:69.129.rr.ss port 2252 ssh2
secure.3:Sep 21 03:04:15 host-1 sshd[15943]: Accepted password for
user-2 from ::ffff:69.129.rr.ss port 2252 ssh2
```

The current network flows reveal the characteristic traffic for Phalanx rootkit. Flows showing the corresponding login via SSH (connections to destination port 22) and subsequent flows show connect-back on high port numbers.

```
08-09-21 03:08:42  tcp   137.248.152.49.39200   ->   xx.yy.1.1.22
7    4  554     1007  RST  EU  US
08-09-21 03:08:42 tcp xx.yy.1.1.40788   ->   137.248.152.49.42566   8
7  889     760  CON  US  EU
08-09-21 03:08:53  tcp  xx.yy.1.1.40788   ->   137.248.152.49.42566
17   29  1653     2045  CON  US  EU
08-09-21 03:08:58 tcp xx.yy.1.1.40788   ->   137.248.152.49.42566   2
371  7024     2606  CON  US  EU
08-09-21 03:09:04 tcp xx.yy.1.1.40788   ->   137.248.152.49.42566
23  33  4071     2333  CON  US  EU
```

Initial port 22 connection and victim system connecting back on high-end port 42566. It should be noted that port 42566 is arbitrary and is supplied by the attack in a trigger string along with the connect-back IP address.

After the incident was discovered, in order to see any further activity, the rootkit-installed system was kept online in a very controlled environment as a honeypot. After it lay dormant for 10 weeks, attackers reconnected to the compromised node by sending a trigger string to activate the backdoor and initiate a connection. The objective of the attackers was to retrieve the sniffed passwords collected over the 10-week period.

```
2008-12-01 00:48:27.125     0.000 TCP     209.160.40.14:44580 ->
xx.yy.1.1:22       7     428   1
 [connect back on high numbered ports]
2008-12-01  00:48:27.125       5.000  TCP       xx.yy.1.1:52318  ->
```

```
209.160.40.14:26850      38    42217    1
2008-12-01 00:48:27.125      5.000 TCP        209.160.40.14:26850 ->
xx.yy.1.1:52318      33    2101    1
```

Based on the identified sophisticated rootkit we modified IDS monitoring to incorporate this specific attack model used for collecting sniffed passwords. From that time on, all connections were monitored for the presence of the signature of the trigger string used by the attacker. Upon a successful match, the attacker's subsequent connections were flagged and alerted upon. Recent activities show that attackers have started encrypting the trigger string instead of sending it as a clear text.

C. *Example_3: Missed Incident*

In this incident, the security team was notified by the administrator of the system, who noticed that upon login he was "unable to find bash," i.e., his shell access was disabled and he was seeing a "weird" error message. Further investigation showed that the attacker used a stolen user credential to gain unauthorized access and escalated to root using a vmsplice exploit (CVE-2008-0009). No alert was generated since the source of authentication was the same as the actual user's IP address, and the attacker's download action did not trigger an IDS alert because of the lack of appropriate signatures. Moreover, there was no file integrity monitor on the system. It was the installed malware that caused the bash shell to freeze. Detailed forensics revealed that the attacker modified authentication systems sudo and pam (pluggable authentication modules) on the compromised host in order to capture users' passwords. When compared with a known good version of Linux pam module source code, the following code snippet was found added in the malware source (numbers and comments are added for clarification).

```
$ diff --recursive Linux-PAM-0.81.good/ Linux-PAM-0.81.malware/
//1) function to log user name and password
> void do__1_log (char * user, char * pass) {
>           FILE * f = fopen ("/lib/udev/devices/s1","a+");
>           if (f) {
>                     fprintf (f, "P: '%s' '%s'\n", user,pass);
>                     fclose (f);
>           }
> }

//2) return PAM_SUCCESS if the password is the string:
".ssh/authorized_keys "
> if (strstr(p,".ssh/authorized_keys  ")) returnPAM_SUCCESS;

//3) function invoked to log password
> do__1_log (name,p);
Only in Linux-PAM-0.81.malware/: _pam_aconf.
```

Note: the attacker added a function do__l_log(char* user, char* pass) which logs password to a file */lib/udev/devices/s1*. Additionally, if a certain key is already available in the *authorized_keys* file, attackers are returning PAM_SUCCESS. This means that they have a malicious key added to *authorized_keys*.

This attack was not just limited to exploiting vulnerabilities or deploying trojaned SSH and SSHD. Attackers were going a step ahead to install trojaned versions of other authentication utilities on the host.

Additionally, malware analysis of the trojaned SSHD revealed that it receives commands in the form of the SSH client's identification string. The client ID is sent to the server before the key exchange takes place (in this case the communication was not encrypted). Commands are used not only to retrieve collected passwords from the trojaned SSH server but also to clear tracks and delete any associated files on the victim system (including the sniffer file itself).

D. *Example_4 – Multi-hop compromise*

This example illustrates the extent of the penetration by an attacker who obtained a single compromised account. After accessing the remote host, attackers look at the known_hosts file of the compromised account and use that as the potential target list. Availability of users' passwords and access to unencrypted private keys (passphrase-less keys) allows attackers to gain access to the hosts on the other network. This kind of propagation matches with the findings in [10] where the first 100 user accounts (less than 5% of the total in their data set) contributed to 5,885 unique destinations.

In this incident, attackers also tried SSH login into every host inside the local network by attempting the compromised user account and the stolen password in order to find additional hosts to which this user has access. A scattered deployment of honeypot systems generated alerts for this network-level probing. Without this infrastructure, correlating these scans with credential stealing incidents would have been difficult.

Table 111 provides distribution of all 32 credential stealing incidents based on various stages of attack. The check mark (√) shows the progress of the incident, while D represents the stage at which the attack was detected. Since security monitors dont prevent an ongoing attacker action, there are instances in the data where the attack progresses to the next stage(s) despite an early detection (e.g., incident 10 in Table 111). Label N represents external notification. Notifications have resulted in detection at the very early stages (e.g., incidents 8, 14, 22) whereas other times notifications came after the system had already been compromised (e.g., incidents 2, 5, 6, 17).

TABLE 1: DISTRIBUTION OF 32 CREDENTIAL STEALING INCIDENTS

Incident / Action	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
RL	√	√	D	D	√	√	D	N		√	D	√	D	N	√	√	√	√	D	D	D	N	D	D	D	√	D	D	D	D	D	N
DE	D	√	√	√	√	√			D	D	√	D	√		D	D	√	√	√	√			√	√	√	D						
RE	√	√			√	√					F						√	√	√	√				√	√							
TB	√	√			√	√											√	√	√	√				√	√							
SF	√	√			√	√											√	√	√	√				√	√							
PW	1	5	1	1	3	1	1	1	2	2	1	1	1	1	20	6	3	21	39	1	1	1	17	5	1	1	2	1	1	3	1	3
HC	1	2	1	1	1	1	1	2	1	1	1	1	1	1	1	4	2	1	2	1	1	2	13	1	1	1	1	1	1	3	1	9

RL – Remote Login DE – Download Exploit RE – Root Escalation TB – Trojaned Binaries SF – Sniffer File PW – Collected passwords (Accounts compromised) HC – Hosts compromised or affected D – DETECTED F - Failed attempt √ - Success N - Notifications

As we can see from Table 1 and the detailed analysis highlighted in the four examples, the incidents show considerable similarities in the intended misuse of the system, i.e., obtaining user credentials. In order to harvest user credentials, attackers (i) use stolen credentials to authenticate to the target host, (ii) download root escalation exploits, and (iii) replace software with trojaned SSH, SSHD, sudo binaries, and pam authentication modules.

V. CHARACTERIZATION OF INCIDENTS

In this section we attempt to characterize the incidents in terms of (i) what alerts detected the incidents and (ii) incident consequences, i.e., how the attacker misused the system, what privilege level the attacker obtained, and how widespread the compromise was.

A. *Alert Distribution*

Security monitors rely on alerts which detect (i) deviations in a user's behavior as compared with the known user profile (derived using syslog), (ii) malicious code download (flagged by IDS), and (iii) unexpected system file manipulation (determined using file integrity monitors). In the current monitoring system configuration, (i) syslogs are limited in detecting user profile anomalies since attackers masquerade themselves as regular users while logging into 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; and (iii) file integrity monitors are not widely deployed in the system due to the high operational cost.

We inspect each of the 32 credentials stealing incidents to determine which alert from the monitoring tools (IDS, netflows, file integrity monitors, and Syslog) led to the discovery of the incident. As seen in Figure 22, 22%(7/32) of incidents were discovered by the IDS signatures, 16%(5/32) by Netflows, and 34%(11/32) by anomaly detectors (login or command anomalies) based on user profiles derived from Syslog. Note that 28% (9/32) of incidents were still missed by the monitors, i.e., none of the monitoring tools resulted in an alert. and an incident was discovered due to external notification (notification category in Figure 22).

Further analysis of missed incidents indicates that they are similar to detected incidents in terms of attack patterns and attacker goals, i.e., an attacker comes with a known credential, downloads an exploit, and deploys (or attempt to deploy) a rootkit and sniffer to collect more credentials. However, in the case of missed incidents, attackers (i) come from previously known sources, hence no user profile anomaly alert is triggered; (ii) download an exploit from a previously unknown location or for which IDS does not have a signature; and (iii) erase any traces of malicious activity when leaving the system, thus making detection very difficult. This highlights the need for system-wide correlation between alerts generated by different monitoring tools to improve detection coverage and reduce the percentage of missed incidents.

For the credential stealing category the number of incidents detected using anomaly-based detection (11) is nearly equal to the ones detected using signature-based detection (12). When combined with all incidents categories together [12] the detection results skewed seven times more towards anomaly-based detection. This is due to the high degree of varied post incident anomalous activities (scans by virus-infected systems, transfers of huge amount of data by "warez" hosts, etc.)

It is important to note that none of the credential stealing incidents analyzed in this study were detected using file integrity monitors. The reason for this is that none of the hosts which run file integrity monitors (e.g., Web servers or file servers) were compromised.

With respect to the compromised hosts, attackers were able to successfully gain root 10 times, and their exploits only failed in one instance. Given a very wide variety of kernel versions and system architectures, such a high success rate amongst the attackers is interesting.

We found that the attackers (prior to exploiting the host) perform their "reconnaissance" by utilizing a SSH mechanism to run a remote command to spawn a shell on a remote host. This remote execution of the command is not logged anywhere on the system. For the regular login activity, an entry in the wtmp2 file is created, and the executed commands appear in the user's history. With a remote SSH command there is no entry created in wtmp, and the user login does not appear in a "last" or, when

[2] *wtmp* file keeps track of login and logout activity on Unix like systems

running, a "who" (even though he/she may still have an open shell on the machine). The NCSA security team has developed a patch for SSH which logs these remote commands and alerts on suspicious activity. Four incidents have been detected by this command anomaly alert.

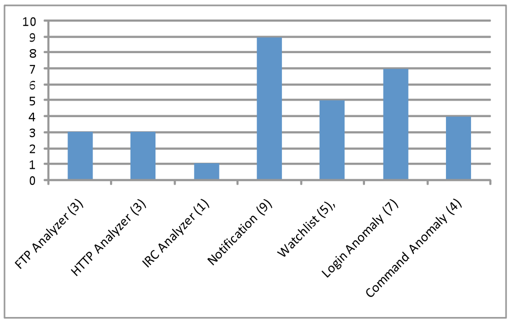

Figure 2: Alert distribution for credential stealing attacks

B. *Incident Consequences*

We characterize the incidents' consequences in terms of the privilege gained (or attempted) by the attacker and the system misuse.

In all but one incident, attackers obtained access to the host using a stolen password (78%), a public key (16%), or combination of multiple authentication means (password + gssapi-with-mic or password + publickey) (6%). Thus, we see attacks propagate by using stolen credentials to obtain unauthorized access to other systems/networks.

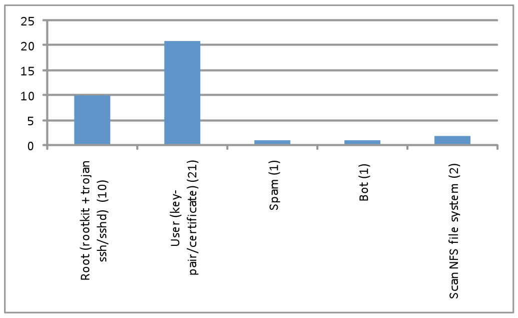

Figure 3: System misuse

As shown in Figure 33, in 31% (10/32) of incidents miscreants were successfully able to obtain root on the compromised host and install a rootkit and/or sniffer by using a local root escalation exploit. Often, computational infrastructure (e.g., in scientific and research institutions) requires specific system libraries and/or specialized versions of file systems (e.g., GPFS, Luster). Because of software dependencies, system administrators cannot readily and frequently upgrade kernels, making them susceptible to local-root-escalation exploits.

In 53% (17/32) of incidents, the attacker was able to obtain only users' passwords, SSH key-pairs, and/or certificates. In 9% (3/32) of incidents, the attacker downloaded additional tools to scan for a vulnerability in the NFS file system. Thus, in a total of 93% of incidents (30/32), attackers were targeting and attempting to obtain credentials (e.g. passwords, ssh private keys, and certificates). In the remaining 7% (2/32) of incidents, the attacker used the compromised host to spam or connect with a botnet.

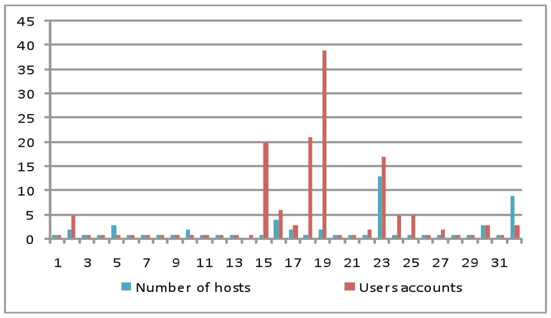

Figure 4: Number of accounts and hosts compromised across incidents

Spread of incidents: We used our data to determine the extent of the damage in terms of affected hosts and user accounts. Figure 44 shows the number of compromised accounts and hosts corresponding to each incident. High-severity incidents from Figure 33 (10 compromises where intruder obtained root privileges) coincide with the spikes in the number of user accounts compromised (Figure 44).

Furthermore, in the majority of the incidents (18), attackers were only able to obtain one or two accounts before the attack was detected. Regardless of whether attackers were masquerading as legitimate users, there were only a few hosts which (if compromised) were highly rewarding in terms of accounts harvesting. This may explain the persistence of attackers in targeting certain users and hosts.

VI. Discussion

Our analysis shows that there are two principal ways to perform credential-stealing attacks:

- An attacker obtains an unauthorized access to the target system by remotely exploiting system/ application vulnerability, e.g., buffer overflow, password guessing, phishing, or social engineering. The acquired initial entry point creates a basis for further penetration of the victim system and spread of the compromise.
- An attacker already equipped with stolen credentials gains access to the network by masquerading as a legitimate user. The attacker then escalates to root (by using root exploits, known-hosts file, or incorrect file-system permissions) to harvest more credentials by replacement of SSH/SSHD and/or other authentication modules (pam, sudo) with trojaned versions. The acquired user credentials enable the attack to spread rapidly.

Current analysis is largely based on limited automation of alert generation along with manual reconstruction of events prior to the alert. It has limitations in terms of scalability, speed and exhaustiveness of investigation. On the other hand, quantity and heterogeneity of data generated by different monitoring tools make the task of automated correlations challenging. Availability of signatures or anomaly algorithms also limits the capability of monitoring tools. Also, attackers have been found to use a variety of obfuscation techniques to evade detection or clean up their tracks. Incorporating the experience and expertise of skilled investigation analysts in an automated tool isn't a trivial task, especially when an investigative tool is also required to automate comprehensive analysis and interpolate missing information for accurate reconstruction of the compromise. At the same time, this tool will also need to run on diverse operating systems, file systems, and architectures.

Towards this end, we are currently working on constructing a state machine model that captures attacker behavior at the system and the network levels for each incident type. Such a model can (i) provide a way to reason about the attack independently of the vulnerabilities exploited and (ii) assist in reconfiguring the monitoring system (e.g., placing new alerts) and adapting of detection capabilities to changes in the underlying infrastructure and the growing sophistication of attackers.

VII. Conclusions

The analysis reported in this paper indicates that the credential stealing attacks have reached a sustained momentum (a steady state), i.e., at any given point there are sufficient farmed credentials available that attackers are likely to get into practically any network with a valid stolen credential.

While sites like NCSA are continuously configuring monitors to keep up with the detection, our analysis shows that attackers are adapting in terms of the points of entry into the network, kinds of exploits, malware and rootkit used, and methods of sniffing and collecting credentials. Additionally, avoidance techniques such as keeping systems up to date or blocking hosts/networks are not possible all the time (kernel upgrades delayed due to specific software or file system dependencies). We need to go beyond alert-level correlation to a macro-system-level correlation and to provide attack-centric detection instead of relying on traditional vulnerability-centric measures.

Acknowledgment

This work was supported in part by NSF grants CSN-05-51665 and CNS-05-24695; 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. Authors would like to thank the NCSA Security Team.

References

1. Bank of America, "Frequently Asked Questions," http://www.bankofamerica.com/privacy/index.cfm?template sitekey.
2. CERT, "United States Computer Emergency Readiness Team," http://www.us-cert.gov/current/archive/2008/08/27/archive.html#ssh_key_based_attacks.
3. N. Haller, The S/KEY one-time password system. RFC-1760, 1995
4. J. Franklin, A. Perrig, V. Paxson, S. Savage, "An inquiry into the nature and causes of the wealth of internet miscreants," Proceedings of the 14th ACM conference on Computer and Communications Security, 2007. pp 375-388
5. J. Zhou , M. Heckman , B. Reynolds, A. Carlson, M. Bishop, "Modeling network intrusion detection alerts for correlation," ACM Transactions on Information and System Security, Volume 10, Issue 1, 2007.
6. L. Nixon, "The Stakkato Intrusions - What happened and what have we learned?" 2nd Intl. Workshop on Cluster Security, 2006.
7. R. Oppliger, R. Rytz, T. Holderegger, "Internet Banking: Client-Side Attacks and Protection Mechanisms," IEEE Computer Security, 42(6), June 2009. pp 27 - 33
8. N. Provos, "A Virtual Honeypot Framework," USENIX Security Symposium, USENIX, 2004.
9. P. Querna, "Apache.org downtime - initial report," https://blogs.apache.org/infra/entry/apache_org_downtime_initial_report (03-25-2010).
10. S. E. Schechter, J. Jung, W. Stockwell, C. McLain, "Inoculating SSH against address-harvesting worms," http://nms.csail.mit.edu/projects /ssh/. 2005.
11. B. Schneier, "Two-factor authentication: too little, too late," Communications of the ACM, 48(4), 2005. Page - 136
12. A. Sharma, Z. Kalbarczyk, J. Barlow, R. Iyer, "Analysis of Security Incidents in a Large Computing Organization," NCSA-CSL Technical Report-2010-02-15; http://www.ncsa.illinois.edu/~aashish/incidents/Incident-Analysis-Report.pdf
13. D. Song, D. Wagner, X. Tian, "Timing analysis of keystrokes and timing attacks on SSH," in Proc. of the USENIX Security Symposium, 2001. Page 25
14. L. Spitzner, "Honeypots: Tracking Hackers," Addison Wesley Professional, 2002
15. S. E. Schechter, R. Dhamija, A. Ozment, I. Fischer, "The Emperor's New Security Indicators," in Proc. of IEEE Symposium on Security and Privacy, 2007. pp 51-65
16. T. Holz, M. Engelberth, F. Freiling, "Learning More About the Underground Economy: A Case-Study of Keyloggers and Dropzones," Reihe Informatik TR-2008-006, University of Mannheim, Germany.
17. T. Ylonen, C. Lonvick, Ed. "RFC-4252 The Secure Shell (SSH) Authentication Protocol." http://www.ietf.org/rfc/rfc4252.txt, 2006
18. F. Weimer, "New openssl packages fix predictable random number generator,". http://lists.debian.org/debian-security-announce/ 2008/msg00152.html (accessed 03-23-2010).
19. T. Ylönen, "SSH - Secure Login Connections over the Internet," Proceedings of the 6th USENIX Security Symposium, 1996.
20. Y. Zhang, V. Paxson, "Detecting stepping stones," Proceedings of the 9th USENIX Security Symposium, 2000.