Global Information Assurance Certification Paper

Sara Dearing
GSEC Practical Requirement v1.4b

Case Study: Statd/Automountd Security Incident

Abstract

This is a case study of a root compromise caused by a Solaris vulnerability in statd that exposed a vulnerability in automountd. The incident occurred four months before any security advisories were circulated about the problem. This paper traces the steps taken to determine what exactly the compromise was, how it was addressed using CERT intruder detection and root compromise recovery documents as guides, and what additional security measures were added to our environment on an ongoing permanent basis as a result of the compromise (servers on isolated ports to reduce the impact of a sniffer, and installation of ssh).

Background

The security incident to be described here occurred in February of 1999 in a university setting. We had a group of unix systems administrators (four full-time employees and one part-time employee) managing about 60 Solaris 2.5.1 and 2.6 servers and workstations. We were led by a manager of distributed systems. In general, each administrator in the group was assigned primary responsibility for monitoring a particular set of our major services, and all administrators helped out managing faculty and user room workstations. All servers and workstations except our DNS server, were part of the same NIS domain sharing the same password and group databases. Workstations had no additional local users or groups defined, and servers had at most a half dozen local users and/or groups defined. We had also implemented a scheme of restricting logins to server machines by changing the modes on all the shell binaries to remove world access and changing the groupship to be "machine-name"ok. Only a small number of users were then added to each "machine-name"ok group. We used daily system reports (displaying things like disk usage, su attempts to root, process listings, etc.) to monitor both servers and workstations. We were also members of both Sun (the operating system vendor) and CERT security advisory mailing lists, and applied any security related patches immediately, while also regularly updating the OS and applying general OS patches usually two times during the year (between semesters and during the summer). The university as a whole, and in particular, our distributed systems group, had no stated security policy/document.

Discovery of the Problem

The morning of Thursday February 4, 1999, the administrator monitoring the university web and usenet news server, emailed the systems administrator group to say she discovered the following processes on the system:

```
root     20786  0.0  0.3 1552 1032 ?        S   Feb 01  0:00
/usr/sbin/inetd -s /tmp/bob
root     20792  0.0  0.3 1568 1184 ?        S   Feb 01  0:00
/usr/sbin/inetd -s /tmp/bob
root     20800  0.0  0.3 1568 1184 ?        S   Feb 01  0:00
/usr/sbin/inetd -s /tmp/bob
```

She indicated that the processes were started on February 1, at 6:42pm. The file /tmp/bob no longer existed (most likely due to a daily script cleanup of /tmp). She had checked the last logs and found two unique student login attempts from a remote site (non-university address) at 6:49pm and 6:50pm on February 1. They were immediately logged off due to our restricted login setup. She asked if any of us had any insight we could share.

Plan of Attack

It was decided that we were probably dealing with a root compromise, since one would need to be root to have started these processes, and none of us were responsible for doing it. The manager collected the group together to discuss the situation. As a group, we decided to do two things. First, try to determine all that we could about the compromise. We based our efforts on the CERT document "Intruder Detection Checklist" (1) Second, we needed to clean up and fix the problem. We decided to base these efforts on the CERT document "Steps for Recovering From a UNIX Root Compromise" (2) The bulk of the tasks involved for both efforts were divided between myself and one other colleague, leaving the remaining three to cover regular daily work and provide assistance to us as they could. Chronologically, we proceeded as follows:

Thursday

I used the truss command to see what the processes were doing. All were sleeping.

Since we use sudo, I checked the sudo logs to determine if anyone had used to sudo to start inetd. There were no unusual entries.

My colleague examined the wtmpx file to determine the full path of the remote site from which the student logins were attempted.

I checked our own internal log of "problem" students to see if the userids used had been involved in some previous offense. They were not.

An examination of syslog showed nothing unusual.

An examination of messages showed:

```
Feb  1 18:42:21  statd[134]: attempt to create "/var/statmon/sm/; echo
  "ingreslock stream tcp nowait root /bin/sh sh -i" >>/tmp/bob ; /usr/
  sbin/inetd -s /tmp/bob &"
Feb  1 18:42:30  statd[134]: attempt to create "/var/statmon/sm/; echo
  "ingreslock stream tcp nowait root /bin/sh sh -i" >>/tmp/bob ; /usr/
  sbin/inetd -s /tmp/bob &"
Feb  1 18:42:40  inetd[20792]: ingreslock/tcp: bind: Address already in
```

```
  use
Feb  1 18:42:40  inetd[20792]: ingreslock/tcp: bind: Address already in
  use
Feb  1 18:42:49  statd[134]: attempt to create "/var/statmon/sm/; echo
  "ingreslock stream tcp nowait root /bin/sh sh -i" >>/tmp/bob ; /usr/
  sbin/inetd -s /tmp/bob &"
Feb  1 18:42:49  inetd[20800]: ingreslock/tcp: bind: Address already in
  use
Feb  1 18:42:49 maenad.csc.albany.edu last message repeated 2 times
(latter message repeated numerous times)
```

I used md5 (from a build done on our file server machine) to checksum /usr/sbin/inetd and compared the result against the same binary on two other workstations in our control. There was no difference.

I used md5 to checksum all the executables called from inetd and all were the same as the other workstations.

I killed all the inetd processes running on the system and restarted inetd.

I modified the daily script to no longer remove files in /tmp (just in case the vulnerability were attempted again, we might catch what was in the file /tmp/bob)

Friday

We searched for setuid and setgid files and found nothing out of the ordinary.

We inspected /etc/inetd.conf and found no changes.

We inspected the local passwd and group files and found no changes.

My colleague contacted CERT. They were willing to work with us over the weekend. Since we were at the latest revision of the statd patch, the CERT representative believed that a relatively unused security hole was being exploited relating to the interaction of statd and automountd. He indicated Sun was aware of the problem and that it involved a buffer overflow. We were only the third report CERT had received within a week. He suggested that we may find information on the vulnerability on the internet at hacker sites if we searched for statd and automountd. We were told repeatedly that we should be suspicious that a sniffer was installed (the two student login attempts from a remote site provided circumstantial evidence that a sniffer might have been installed). While he indicated that sniffers are more difficult to detect because they leave a small footprint, he suggested checking if the machine was in promiscuous mode, or by tearing the file system apart looking file by file for a log. His initial recommendations were to:

- disable automountd on all hosts
- pressure Sun for a patch
- run the ifconfig command from a distribution CD to check for promiscuous mode
- possibly tear the file system apart looking for a sniffer log
- install secure shell to protect passwords with encryption

- start network logging to a low profile, restricted access host to guard against the removal of local logs on a compromised machine (logs would still be available on the remote loghost).
- mail CERT with any information on the incident
- review the CERT document on recovering from an incident
- review use of rsh and .rhosts trust arrangements; think about kerborizing our network

We decided to additionally:

- check what machines would be sniffable from the compromised web server
- warn others at the university to turn off automountd
- check all our machine log files for the statd errors to see if there was any evidence of other machines being compromised
- check if the two student userids used, had attempted to login to any other of our machines
- run ifconfig on all our machines checking for promiscuous mode

One of my colleagues found exploit code at a site that no longer exists (however, similar code can now be found at (3)).

My colleague contacted Sun. CERT had not provided us with a Sun bugid so the Sun representative was unable to find any bugid associated with the statd/automountd vulnerability. He put a level 1 priority on the call and passed it to the security team but indicated we would most likely not hear anything until Monday.

Three of us shut off automountd and added hard mounts as needed to /etc/vfstab files on all our machines.

Saturday

All local account passwords were changed on the web server.

All users listed in the "machine-name"ok group were reviewed and those no longer needing login access were removed.

All services in inetd.conf were reviewed and unnecessary ones were turned off.

We ran various snoop tests on the web server to see what kind of traffic would be visible to a sniffer and did not see any evidence of plain text passwords (only a very small number of users were even allowed to login to the web server).

I ran a script on all our machines that checked for statd and/or ingreslock messages in /var/adm/messages; moved the start script for automountd out of the way so a reboot of any machine wouldn't inadvertently start the automounter again; checked for extra inetd processes; ran ifconfig to see if any machine was in promiscuous mode. All machines looked okay.

We met with one of our colleagues in the university data communications department to determine just what could be sniffed from the web server. He assured us that the web server was on an isolated port on the router such that it could only see its own traffic (and broadcast traffic). He indicated the same was true for our nis master server, new

web server, dns server, library server, and webct server. Our compute server and our file server could see each others traffic, so the data communications person put them on isolated ports by mid-day. We decided that we would rebuild at least the web server and possibly other major servers if there was evidence of compromise, and we would keep a close eye on the rest of our machines.

We contacted CERT again and they agreed that our rebuild plan seemed reasonable. They again recommended using secure shell (ssh), and to also consider "what if ..." type questions at a later point, in order to develop plans for handling incidents if the circumstances were different from this one.

We rebuilt our jumpstart server.

Sunday

We sent a message to other unix systems administrators at the university to briefly describe that we had been compromised, CERT had been notified, and that it was recommended that automountd be turned off.

Following Weeks

Over the course of the following week, we used the jumpstart server to rebuild our web server and dns server. The week after that we rebuilt our nis server. All were also upgraded to Solaris 2.6. Secure Shell was built and all system administrators were encouraged to use it, but insecure methods of logging in were not disabled (i.e. telnet, rsh, etc.) After a significant amount of badgering, Sun eventually indicated they did not have a patch for the problem yet, but advised not only stopping the automounter but also having statd run as a non- privileged user. We implemented that recommendation on all our machines, and also informed the other unix systems administrators on campus.

It was not until early June that CERT released its Advisory CA-1999-05 Vulnerability in statd exposes vulnerability in automountd (4) and Sun had released a statd patch for Solaris 2.5.1 which addressed the problem (5)

Conclusion

Using something as simple as a daily monitoring script, we were able to successfully detect a compromise. Then by following the CERT intrusion detection and root compromise recovery documents, we were able to relatively quickly outline a plan of attack for how to deal with the incident. We were successful in fixing the problem via the advise provided by CERT and Sun. We were able to enhance our overall security situation by moving servers not formerly on isolated ports to such ports, and by installing ssh.

On the downside, we have yet to set up a low profile, limited access central loghost as this is a task that always gets pushed aside by more important tasks. We also have not eliminated insecure login methods,

such as telnet, although we are making slow progress in that direction. We still have no stated university security policy, but we have recently gained a CIO who has since designated a security position in the data communications area of the organizational chart. We never did address the questions of "what if ..." to plan how we would address an incident if the circumstances were different. However, we did not run into any ill side effects nor does it appear we missed anything crucial by using the CERT documents as a guide for how to handle the incident.

References

(1) CERT Intruder Detection Checklist
ht tp://www.cert.org/tech_tips/intruder_detection_checklist.html

(2) CERT Steps for Recovering From a UNIX Root Compromise
now called Steps for Recovering from a UNIX or NT System Compromise
http://www.cert.org/tech_tips/root_compromise.html

(3) rpc.statd/automountd exploit code download available from:
http://packetstormsecurity.nl/groups/horizon (click on statd.tar.gz)

(4) CERT Advisory CA-1999-05 Vulnerability in statd exposes vulnerability in automountd
http://www.cert.org/advisories/CA-1999-05.html

(5) Sun Solaris Patch-ID#: 104166
http://sunsolve.sun.com/pub-cgi/show.pl?target=patchpage
(enter 104166 under patchfinder and click find patch)