From write to root on AIX

A case study

Silent Signal LLC
Email: info@silentsignal.eu
Web: www.silentsignal.eu

SILENT SIGNAL
VÉSZJELZÉS HELYETT...

From Write to Root on AIX

1 Introduction

During an ethical hacking project, Silent Signal was able to get remote user-level access to an AIX server of the Client running critical business processes. To assess the local security level of the host and to demonstrate the potentical impact of the remotely exploitable vulnerabiltiy, Silent Signal attempted to elevate their privileges to the administrative (root) level. After trying several common methods without luck, the team solved the problem with the combination of two distinct vulnerabilities. The purpose of this document is to demonstrate how the individual security issues can boost each other's impact and lead to total system compromise. Of course – according to our contracts – sensitive information about the Client have been removed or altered.

2 Restricted write access with root privileges

The information collected during the earlier phase of the assessment (usernames, passwords) allowed us to log in to the target server with an average user. The privilige level allowed us to open an interactive shell:

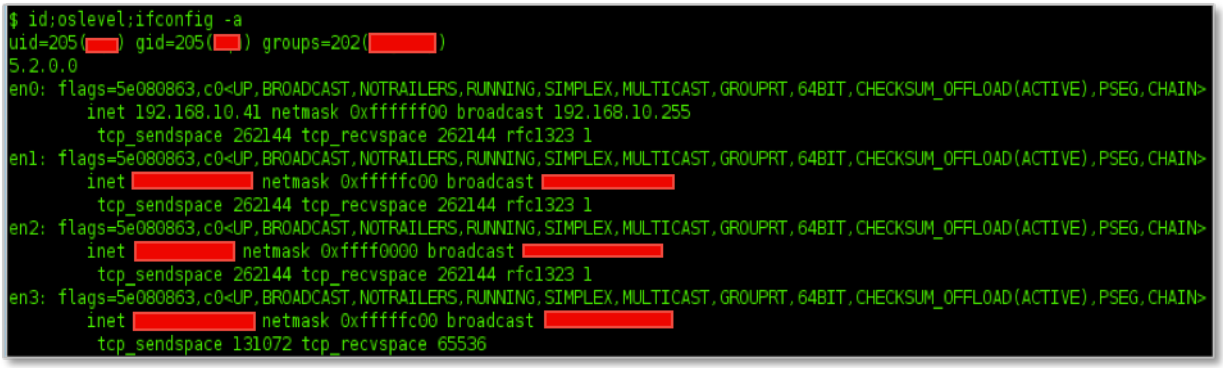

Figure 1: Initial shell access

The server was running AIX 5.2 operating system. Of the publicly known vulnerabilities only the CVE-2009-2669 libc.a issue seemed to be exploitable. This vulnerability allows any user to create world writable files with root privileges. As it is stated in the public exploit, the vulnerability can't be exploited in the common ways like overwriting `.rhosts` or manipulating the dynamic linker - a system specific attack vector was needed.

3 Security Feature - Security Bug

The testing team found a script in the `/etc/profile` file that checked the expiration dates of passwords and alerted the users if neccessary. It seemed that the administrators tried to enforce

password policy – one of the most important parts of a corporate security policy – but as it turned out, this method of enforcement allowed our team to fully exploit the vulnerability described in the previous section.

```
trap 1 2 3
sudo /usr/local/bin/pwlejar.sh $LOGNAME bbb

PATH=/usr/bin:$PATH     # SD Installer: do not remove !
export PATH     # SD Installer: do not remove !
#START===SDO_PROFILE_BLOCK_DO_NOT_CHANGE===START
```

Figure 2: Part of `/etc/profile`

The above screenshot shows that the script is executed with root privileges using sudo. Configuration of sudo allows any user to run this script with elevated privileges without providing a password:

```
if [ $# -lt 2 ]
  then echo A $user passwordje meg $maradek napig el
  else if [ $maxage -ne 0 ] && [ $maradek -gt 0 ] && [ $maradek -lt $pwwarn ]
         then
          echo "\t#########################################################"
          echo "\t# Figyelem a passwordje $maradek nap mulva lejar!!!\t\t#"
          echo "\t#########################################################"
           echo "\tUss barmit..."
           read a
       fi
fi
$ sudo /usr/local/bin/pwlejar.sh [redacted] bbb
$
```

Figure 3: Checking execution, no error

The `/usr/local/bin/pwlejar.sh` was overwritten by exploiting the libc.a vulnerability. The exploit set the first line of the file to the *Initializing* string. When run, the script informs the user that the Initializing command can not be found.

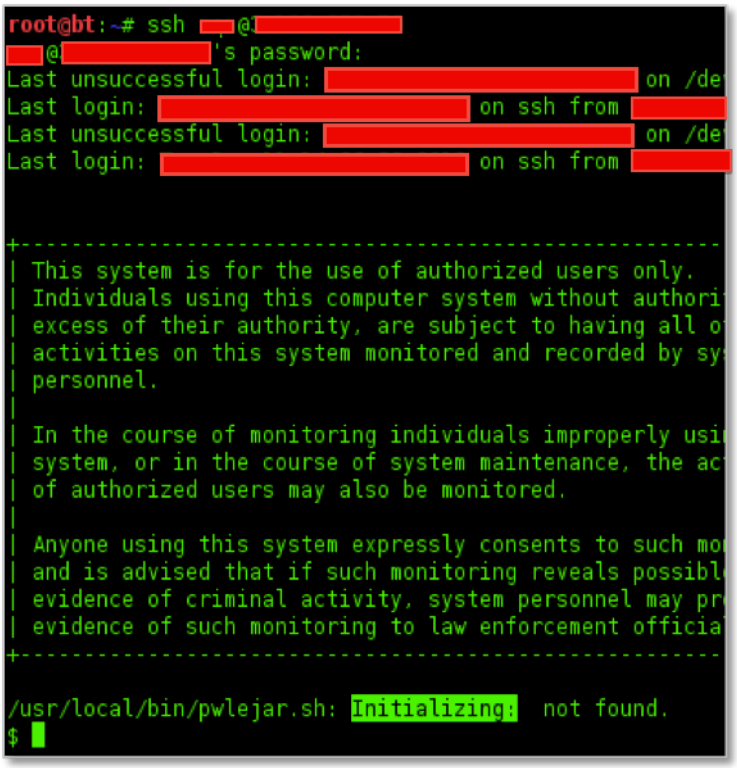

Figure 4: The Initializing command can not be found

Finally, the testing team created an executable shell script called Initializing in a writable directory that executed the `/usr/bin/id` command. To exploit the issue, the `PATH` environment variable had to be set to the directory of the `Initializing` script, and run the `pwlejar.sh` through sudo.

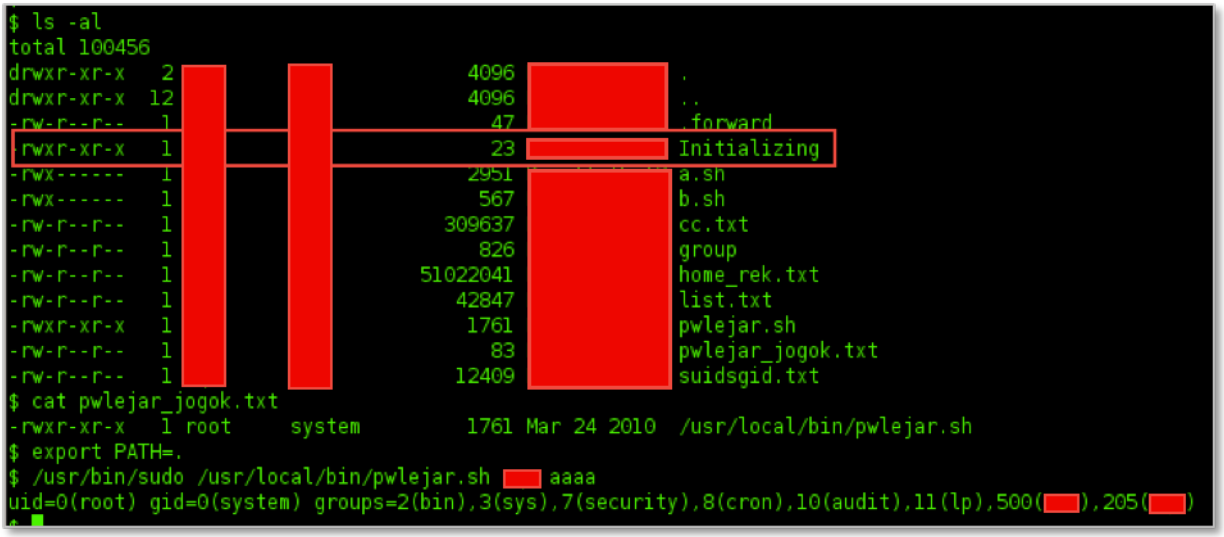

Figure 5: The `id` command runs as root

The `id` command can be replaced by arbitrary commands of course, obtaining an interactive root shell is trivial from here.

4 Conclusion

In case of business critical systems the operating system is often not properly patched, because the availability and flawless execution of the services override the requirements of security. The situation of security-aware administrators is even more difficult, if they have to reason for patching problems that are not so easy to demonstrate, like the one discussed above. Stressing the importance of security updates is thus inadequate, but offensive assessment might help priorizing the patches and prove the need for them.

It should also be noted that the implementation of security functions – either provided by well-known products or internally developed software – may introduce new vulnerabilities, just like as with built-in ones. The impact of these problems are usually high, since security features require high privilege level most of the time. It is thus important to take care of the assessment of the security of our security systems, along with the components protected by them.