TITAN

Dan Farmer, Earthlink Network
Brad Powell, Sun Microsystems
Matthew Archibald, KLA-Tencor

TITAN

Dan Farmer – Earthlink Network
Brad Powell – Sun Microsystems, Inc.
Matthew Archibald – KLA-Tencor

ABSTRACT

Titan is a freely available host-based security tool that can be used to improve or audit the security of a UNIX system. It was written almost completely in Bourne shell, with a master script controlling the execution of many smaller programs. Each of the programs either fixes or detects potential security problem, and its simple and extremely modular design also makes it useful to help check or enforce the adherence of a system against its security policy. Finally, anyone who can write a shell script or program can easily create their own Titan modules.

Titan does not replace other security tools, nor does it fix or patch security bugs; its primary purpose is to improve the security of the system it runs on by codifying as many security tricks to secure an OS that the authors could think of. And when used in combination with other security tools it can help make the transformation of an ''out of the box'' system into a firewall or security conscious system a significantly easier task.

NOTE: Due to time, resource, and expertise limitations, the first release of Titan is only known to run on Solaris Operating Systems, versions Solaris 2.x and Solaris 1.x. However, many of the small sub-programs within Titan work well with other UNIX's, and other than taking the time to create Titan modules for them, there is nothing Sun specific about Titan that would prevent it working on other UNIX systems.

Introduction

UNIX is often, and justifiably, criticized for being a difficult system to administer because it is not only complex and cantankerous but hard to secure. Its enormous configurability, the fact that vendors don't ship secure systems, and that it requires significant amounts of time, resources, and expertise to safeguard a host are only some of the reasons that so many UNIX systems are insecure on the Internet. To compound the problem, like all modern operating systems it not only becomes less secure as time goes on (simply due to usage), but with the rapidly changing security field, it also requires considerable effort to stay abreast of the latest information – time that most system administrators simply don't have.

Titan tries to provide at least a partial solution to all these problems by trying to locate and fix many of the more common procedural problems that crop up, as well as put into one place all those damn OS tweaks that can assist in securing your system. Titan improves the security of a system by:

- Cutting off entry points into the system.
- Mitigating or preventing the effects of various denial of service (DOS) attacks.
- Turning on or improving the level of logging and auditing features.
- Improving network and local (e.g., host level) defenses.
- Assisting in programmatically defining and enforcing a system security policy.

It is important to note that Titan's focus is the correction of procedural problems. While it can be used as an adjunct to other auditing tools, whether host or network based, it does not attempt to find problems that it cannot correct. An automated tool that changed weak passwords, unpatched or insecure system binaries, and unrestricted filesystem mounts, for instance, could break or disrupt operations to an unacceptable level. Like most other security tools, Titan is not meant to be used only once: to achieve effective security requires an ongoing concern and continued attention to good security practices. Any competent system administrator should have considered, if not resolved or repaired, nearly all of the problems that Titan addresses on their security critical systems.

Anyone working in security or systems administration who has been around the Internet for any length of time has done it – making the same changes, over and over again, to secure a system. Worse yet, each new OS release brings tiny, seemingly arbitrary changes that can invalidate prior work. And forget it when a major new release comes out, or you have to work with another operating system altogether! Just among the authors of Titan we've ftp'd Crack, COPS, and other security programs from the net thousands of times – and we're sure we're not alone.

The analysis of the security of a system is depressing – the same sets of problems always come up. But what's worse is that these problems can almost always be easily fixed – so why aren't they? And the final sling of indignity is that vendors keep changing the damn commands to do the same things, even within the same major version of the OS (what are the arguments to *ndd*(1M) that change that TCP behavior?) And it's certainly not only Sun – it's DEC, it's HP, it's IBM, it's everyone that has even a mildly complex system. Yes, even Microsoft.

So why do these same problems show up over and over, regardless of the supplier? Good question! We don't know the answer, but what we do know is that having a tool to help ensure your system's consistency is a very positive step in the right direction. Hence Titan.

Titan's main design goals are:

- **Security comes first.** We can mandate that for this tool, Security comes first. After Titan has been run on a system it should be more secure than before and there will be no remaining significant host level security problems that we know of, other than those involving vendor OS and independent daemon security issues and patches (e.g., if the system is a WWW server, CGI or CGI-like interfaces could be problematic). The system will not be 100% secure – none are – but it will be pretty darn secure, especially after applying security patches. Due to customer pressure vendors can't take the chance with their patches and system releases that things will break – but we can. In our testing and use over the years we haven't run into a single thing that Titan has irreparably broken, but it certainly could happen.
- **Easy to use.** Titan should be simple to install and run. While knowledge about the system will always help, you can trust Titan to do the right thing in most cases.
- **Policy based.** Titan can assist in the creation of a programmatically defined technical system or site security policy. Classes or types of security (such as firewall, desktop, etc.) are simple to define and apply to the appropriate system, and help produce a consistently secure system in ways that are readily comprehensible.
- **Freely available source code.** In security it is imperative to have complete source code availability. Having full control over what is run and possessing the potential for total understanding of exactly what actions it performs is mandatory for a truly effective security tool.
- **Modular.** Titan is non-monolithic and easily extended. Shell scripts or other programs can either be taken out of or added into Titan's framework. Doing so will not affect the other programs.
- **Useful.** Quite simply, we've found Titan to be enormously handy and something that can be used quite frequently if security is a significant concern for you or your systems.

We're often asked how to tighten down the OS when a firewall product gets installed. There is a reasonable expectation from the customer that after the firewall is installed, the system will not be compromised by an attack that is outside the scope of the firewall product. After all, aren't firewalls supposed to protect you? You wouldn't say it was safe to run a business on the Internet unless you could protect it, would you? Unfortunately most people don't know what security is, and the firewall sales people are not going to help.

And it really is unreasonable to expect the user, a customer, to understand all, or even most of the security issues of running a system on the Internet. Why should they? However, this does place both the specific firewall vendor and security people in general in a rather awkward situation. Indeed, what probably scares firewall vendors more than anything else is the fact that firewalls are failing because users or administrators don't fix, remove, or upgrade old versions of a potentially vulnerable network service.

Titan Does Not . . .

It is important to note that there are several procedural security problems that Titan does not attempt to fix or seal off. CGI programs, often deployed on WWW servers, are becoming one of the more widespread security problems on the Internet, and are nearly impossible to programmatically detect or prevent. Titan also does not render a system impervious to breakins – not only are inside attacks common, but new bugs and holes are constantly being found. Remember – there is an arms race going on out there.

In addition, Titan does not address the problems of secure software distribution or updates. This means that Titan is probably not a viable tool to secure all systems on a large network due to the administrative costs involved in setting up and maintaining all the copies of Titan. While NFS, rdist, or other methods of software replication and deployment could be used, we would warn that the inherent security risks of such methods, as well as the myriad dangers of controlling an organization from a central location probably outweigh the benefits in most situations.

Using Titan

Using Titan is fairly simple, but we want to reiterate – it cannot secure a system, nor can it fix problems that will inevitably arise unless you continue to run it. We suggest that you employ the following sequence when first utilizing Titan:

- Read the Titan documentation and look at the programs. Does it do what you want? Does it fit into your security policy?
- Examine or secure your system using your normal set of tools and procedures. Note any flaws.
- Back up your system.
- Run Titan.
- Examine your system again. Are the flaws

gone? Are new ones there? Does everything still work?

- Install strong authentication on your system, at least for remote logins. Anyone using the same reusable password in cleartext over the Internet except in an emergency is a fool.
- Continually monitor your system, running Titan and other security tools as well as applying security patches as necessary or as your security policy changes over time.
- Report any problems found with Titan to us so we can fix them in subsequent releases!

After the initial use of Titan we suggest running it in the verify mode at least once a week. You can run Titan from cron in the fix mode, but since it can affect your system drastically we would suggest being cautious about doing so (in addition, if you run the Tripwire integrity checking tool, it will complain vigorously about all the files Titan changes). In addition, we highly recommend that stronger authentication (such as with the Logdaemon package or hardware methods) be installed and utilized on the system. When data is traversing the network, strong encryption should be used, if possible, in addition to the extra authentication.

If Titan does the majority of things that you personally do to secure your systems but misses a few points, you might consider writing some additional Titan modules to perform the tasks. We would also ask you to send us email about this – if security related, we would certainly consider including your module in the general Titan release or rewriting it ourselves.

A Case Study

Lucinda Williams is a ten year veteran of system administration and security on the Internet, and has been recently appointed chief security officer for the medical center of her alma mater, Evil University (Evil U, aka evil.edu). After modifying Evil U's general security policy to fit in with the needs of her constituents, she has started to implement the technical aspects of the program. Since the university is strapped for cash, the firewall (an Ultra running Solaris 2.6) must also serve as a WWW server. Here are the steps she takes to create and secure the medical center's firewall:

- A new system is installed with the absolute minimum number of options required to run the system, which lives on its own subnet to prevent local packet sniffing. Immediately, inetd is (temporarily) disabled to help ward off intruders attacking the system before it has been properly secured. As soon as she downloads all the security tools and files needed to install the system, it is physically disconnected from the network. GCC (the GNU C compiler) is also installed so that various security tools can be compiled. (She could alternately compile them on another system that is known to be uncompromised and ftp the binaries over, but it's safer to compile them locally to help ensure that they have not been tampered with.) The Apache httpd server is installed because of its good security and source code availability. The most recent version of sendmail is then put into place.
- The packet screen is the first defensive tool set up. Almost without exception, any critical system that is not protected by a screening router, proxy firewall, or both (or, less ideally, a program such as screend or IP Filter that duplicates this function) has not been adequately protected. While not sufficient to secure a system or network by itself, it is a necessary part of any security solution. Under most circumstances the router shouldn't have to allow more than a half-dozen ports or so from the outside world. DNS, SMTP, http, telnet, and some ICMP is all Lucinda allows, although she is forced by Evil U's policy to allow NFS, NetBIOS, finger, and ftp to the rest of the University.
- At the time of this writing, ftp://sunsolve1.sun.com/pub/patches/ contains all the Sun OS and program patches (including many security fixes). Sun also provides a description of and a tar file containing all their recommended patches for their released OS's, which Lucinda ftp's and installs. This is done before running Titan, since the patches might undo some of the system modifications Titan performs.
- She has previously checked Titan against her firewall security policy, and has had to make a few small changes:
  - The issues file needed revision to reflect Evil U's administrative policies.
  - She needs to allow root ftp access (a truly abysmal idea, but one required by university policy), so in the ftpusers.sh Titan module, she removes root from the $DEFFTPUSERS variable.
  - The university has a customized (and mandatory) compiled C program that all systems must run via the /etc/aliases file for inventory purposes. She has several choices here – she can modify the Titan module (aliases.sh) that doesn't like mail aliases that point to a program, ignore the warnings, or not run the alias module at all. The last can be accomplished either by creating a Titan policy file with all the modules that she does want to run, or by simply moving the shell script out of the modules directory.
- She runs Titan with the -f flag to fix all the problems it detects, and then installs Titan in cron to run with the -v flag once per week.
- Logdaemon – despite being vulnerable to

session hijacking and eavesdropping – is used to improve the authentication of all users instead of the popular but much more complex and potentially dangerous ssh. All accounts have their normal UNIX reusable password disabled.

- Next she runs Tiger and/or COPS, fixes any problems found, and creates a cron job to run the tool once per day, mailing the results to her.
- Logging tools are then set in place next. The TCP and portmapper wrappers and swatch are all installed, with syslog sending information both locally and to a central server, and furthermore any critical events are mailed to Lucinda's pager.
- As the final step in the setup process, she removes GCC and makes a full backup of the system, storing it off-site.

The system is now ready to run. She'll test the initial security with a remote security scanner that is run on the outside of her domain (and hopefully outside the university). Any of the widely available programs such as SATAN, ISS, SAINT, or CyberGuard would work, depending on her familiarity with these and her budget. Initially, the port scanner is the most important thing to run. She also subscribes to the BugTraq, Sun security advisory, and CERT mailing lists, and will keep a close eye on the system logs and activity. She has also created an addendum to her local security policy that will require any and all CGI programs to be audited and personally approved by her as well as being placed in sbox (a CGI safety wrapper). If all this is done, the system shouldn't take too much time to set up and continue to run, and should be a very secure system. The rest of Evil U is perhaps her largest security concern, since they have significant access to the rest of the network she maintains, but there is little she can do but use the TCP and other wrappers and auditing tools to watch the traffic.

NFR, tcpdump, or other packet watching tools can be potentially marvelous tools, but do require a significant time investiture to run effectively. The widespread availability of very inexpensive large high speed disks (to save the voluminous audit data) does make the process more viable, however.

Titan Features

Although Titan has been a dynamic system, continually adding features and additional fixes or tests, we feel it important to cover some of the more interesting tests or features. Code fragments will frequently be given, but with any of the problems listed below, looking at the source code of the corresponding Titan sub-program can be illuminating.

The following sections discuss some of the changes that Titan makes to a system. However, any list we could create will be out of date fairly soon – the complete and up to date list can be found at the online Titan documentation. Since Solaris 2.x contains several ways to improve a host's security that earlier versions of Solaris did not, we naturally have more Titan modules for it – and generally recommend running it or a similar system if security is a concern.

Kernel Level Configuration

Why is it that almost every proxy firewall we see has ip_forward_src_routed enabled? Source routing and other such options may have their uses, and at one time were fine ideas, but they do not belong in the world of Internet security, unless you're trying to circumnavigate it. Tools to abuse and bypass systems that aren't sewn up tightly proliferate on the Internet.

Most modern UNIX's allow a great deal of kernel tuning from the command line. Solaris, for instance, has *ndd*(1M), which can get and set configuration parameters in TCP kernel drivers. Putting them in the /etc/system file makes the change take effect at boot time. Titan closes various kernel and TCP/IP protocol holes that we're aware of, including:

- Fixing the stack. Ever since Aleph1's pivotal paper detailing how to exploit buffer overflows, stack smashing programs have perhaps become the most common type of exploited coding error. Solaris allows the kernel stack to be non executable; it adds the following entry into /etc/system so that zero-fill-on-demand pages are marked rw- instead of rwx:
  - Don't allow executing code on the stack

    ```
    set noexec_user_stack = 1
    ```

  - And log it when it happens.

    ```
    set noexec_user_stack_log = 1
    ```

- NFS bind. Titan sets the privileged port definition to all ports above 2050. NFS, which uses UDP port 2049, has been historically set in an unsafe port range. If you want to protect other services above this range, simply change this parameter.

  ```
  ndd -set /dev/udp \
    udp_smallest_nonpriv_port 2050
  ndd -set /dev/tcp \
    tcp_smallest_nonpriv_port 2050
  ```

- SYN time-out. A good example of Titan's use as a short-term workaround until a security patch has been disseminated by the vendor. This script was produced from a CERT advisory and placed into a Titan module to run on all local systems to reset system parameters to a safer level until the vendor was able to produce a permanent fix (still useful on older systems!):

  ```
  ndd -set \
      /dev/tcp tcp_ip_abort_cinterval \
      10000
  echo "tcp_param_arr+14/W 0t10240" | \
      adb -kw /dev/ksyms
  /dev/mem
  ndd -set /dev/tcp tcp_conn_req_max 8192
  ```

- Ping echo. Titan can set up your system so that it does not respond to broadcast ping requests. Why is this important? Attackers often use a ping flood as a DOS attack. In addition, by turning off response to broadcast echo it makes it more difficult for potential attackers to probe our system by sending a ping -s to the broadcast network address:

```
ndd -set /dev/ip \
    ip_respond_to_echo_broadcast 0
```

Startup files

- /etc/rc.* , /etc/rc?.d/*, etc. The rc shell scripts are full of services which startup at boot time that you may not be aware of. Titan will disable services that can potentially be used to gather system information remotely or aid a potential intruder in an attack – this includes the automounter, the dmi, lpsched, snmpdx, and other daemons. Titan disables these services by either commenting out the services or by simply moving the files from the rc* directories.

Configuration files

- sendmail.cf. Titan enables the privacy flags that were introduced in sendmail version 8 with the "goaway" option (among other things this disables VRFY and EXPN), as well as setting the sendmail logging to a reasonable level:

```
Opgoaway
O LogLevel=5
```

- inetd.conf. Titan tears out many of the default services in the Internet services daemon's configuration file. Most of the daemons installed by default are too chatty, historical sources of system vulnerabilities, and operationally unnecessary. Any inetd service that isn't protected by using tcp_wrappers or otherwise restricted and logged (and/or encrypted) is inherently insecure. You should view any program that talks to the network with grave suspicion.
- ftpusers. If the file /etc/ftpusers exists and has users names listed in it, then those users are not allowed to use ftp. Titan adds in system users such as "bin", "lp", "root", and others.
- nsswitch.conf. The contents of this file can be as dangerous as having a "+" in your /etc/hosts.equiv. Having an "nis" or "nis+" entry in this file gives control of crucial files that your system trusts to a remote system. Titan takes the approach that if the local host doesn't know about a remote system, then that remote system can be a threat. Titan errs on the side of safety and simply builds a minimal /etc/nsswitch.conf file using the /etc/nsswitch.files sample file as a starting point for you to build upon, if necessary.
- syslog.conf. Titan modifies /etc/syslog.conf so that console auth notice messages also get logged to a file.

File and Directory Permissions

- System umask. Titan forces the system (root) to use a default file creation mask (022) that is more secure than the default. This forces all the system daemons to create files with saner file permissions.
- System files and directories. One of the oldest (and still one of the easiest) ways of bypassing system security is to find a directory or a binary file that a privileged user (root most often) is going to access or execute. If that file or the directory that the file/binary lives in is modifiable by another user, then that user can gain additional privileges. Because of the great number of potential problems and different security models for different types of systems (firewalls, servers, desktops, etc.), Titan has three modules that each repair or change one or more aspects of this. All of them can be run for maximum security.

General System Configuration

- eeprom. One of the few things that Titan simply checks and doesn't fix is if you have "security-mode" set in your EEPROM (we don't fix this because you have to choose a password yourself). Don't let us say "we told you so"! If you don't set your EEPROM password, someone else may set it for you – and halt your system. Then you'll need a new EEPROM (or many of them, if they break into multiple systems!), which can take a significant amount of time, especially if you're running an older or discontinued architecture.
- vold. *Vold*(1M) may seem innocuous, but letting users mount file systems without being root doesn't sound like a good thing to us, even if the Sun vold doesn't allow SUID root shells on the file system being mounted. You might consider allowing this on desktops, but certainly not on servers or firewalls.

Passwords and Authentication

- /etc/passwd. Titan deletes or disables system accounts that are never (or should never be) logged into. Any user or system accounts left on the system have passwords or are disabled, and system accounts other than root and sys (which starts accounting) also have a special non-interactive shell put in place.
- Network Information Services. Titan disables NIS, NIS+, and DNS for name resolution from /etc/nsswitch.conf. While all of these network naming services are insecure, you might have to enable DNS, although we suggest keeping it off whenever possible. Never run NIS or NIS+

on a secure system – or it won't be.
- loginlog. Titan creates /var/adm/loginlog so that the system will log more than five failed login attempts. (This doesn't work with all services (telnet for example).

Titan and Your Security Policy

Titan creates an /etc/issue file with a dire warning to stay away from your system. The contents of this file appear in front of the login prompt. By default it contains:

```
################################################
# This system is for the use of               #
# authorized users only.  Individuals         #
# using this computer system without          #
# authority, or in excess of their            #
# authority, are subject to having            #
# all of their activities on this             #
# system monitored and recorded by            #
# system personnel.                           #
#                                             #
# In the course of monitoring individuals     #
# improperly using this system, or in         #
# the course of system maintenance,           #
# the activities of authorized users          #
# may also be monitored.                      #
#                                             #
# Anyone using this system expressly          #
# consents to such monitoring and is          #
# advised that if such monitoring             #
# reveals possible evidence of criminal       #
# activity, system personnel may              #
# provide the evidence of such monitoring     #
# to law enforcement officials.               #
################################################
```

Although we strongly suggest running all (or nearly all) the modules in Titan, we realize that not everyone can afford such strident security measures. Titan thus allows you to run different sets of modules as desired by using a simple configuration file. This configuration, or policy file, is a standard UNIX-style configuration file that uses pound signs ("#") for comments and contains one Titan module (with any arguments desired) per line. We include two sample files, for potential use on desktop and server systems.

Note that the desktop policy disables *sendmail*(8). Since firewalls often deliver or forward mail, this is an optional Titan module. Desktops and most servers have no business running sendmail, however.

Implementation

As previously mentioned, Titan is a master script that runs a collection of Bourne shell scripts. However, before getting any results or fixes from Titan, you must first run the Configure script, which figures out which OS type and version you're running and creates links to the proper Titan modules. Unless something goes awry, this takes no input from the user.

Once configured, Titan has three primary modes of operation to choose from: Fix, Verify, and Inform. Fix, the most commonly used, simply tells Titan to run out and fold, spindle, and mutilate your system in all the ways it knows about in order to create a more secure system, while informing you of what it is doing. The Verify mode uses the same set of tests but instead of changing the system it simply informs you that various changes would be made if run in the fix mode. The Inform mode takes no investigative or corrective action, but simply echos the function of each module.

Unless you're using a predefined policy, the main Titan script will run all the programs in the module directory with the same mode. You can either create a policy or simply move any modules that you don't want run out of the module directory and Titan will not run them.

Each Titan module accepts one of three arguments – Fix (-f or -F), Verify (-v or -V), or Inform (-i or -I). The Inform argument merely prints out what the script will do. Unless run under the policy mode, Titan runs the modules in alphabetical order (sorted by the shell using the "*" wild card).

The Anatomy of a Titan Module

Although Titan can run any executable program, it is currently written almost entirely in Bourne shell. The Titan scripts are heavily commented, and were intended to be easy to understand. Following the same general form, they start by setting a safe umask and with the copyright notice:

```
:
#
umask 022
# This tool suite was written by
# and is copyrighted by Brad Powell,
# Matt Archibald, and Dan Farmer
# 1992, 1993, 1994, 1995, 1996,
# 1997, 1998 with input from
# Casper Dik, and Alec Muffett.
#
# The copyright holder disclaims
# all responsibility or liability
# with respect to its usage or its
# effect upon hardware or computer
# systems, and maintains copyright
# as set out in the "LICENSE
# document which accompanies
# distribution.
```

The scripts then set the path and do a sanity check to verify root is running the program (since Titan almost exclusively modifies root or system owned files, it makes little sense to run it as anything else):

```
# who am I?
MYNAME=‘basename $0‘

# UCB rules!
PATH=/usr/ucb:/bin:/usr/bin:/sbin
```

```
# did things work out or not?
action='./sanity_check $MYNAME $1'
if test $? -ne 0 ; then
   exit 1
fi
```

Titan scripts then have three functions – Intro(), Check(), and Fix() – to do all the serious work.

The Check() function used when a Titan script is run in the "-v" ("V" for Verify) mode. You might look through a few of the Titan scripts to see some of the ways the Check() function examines the system. The only action that all Titan scripts do in the Check() function is to output either "PASSES CHECK" or "FAILS CHECK", so users can figure out if this Titan fix is needed or already applied to the system. It can be as simple as (taken from dmi-2.6.sh); see Figure 1. The fix code is similar, but instead of simply stating that there is a problem, it actually takes action (this code snippet is from disable-L1-A.sh, which disables the L1-A or stop-A keyboard sequence by modifying /etc/default/kbd); see Figure 2.

Finally, a Titan module processes the user arguments to see what action to take, and returns a 0 if the module is successful, and non-zero if something goes wrong.

Building a Titan Module

To build your own Titan module, you might want to start out with the ${TITAN-HOME}/arch/sol2sun4/src/stubs/skeleton script – unless, of course, you want to write something other than in Bourne shell. The key points are that the module accepts the basic three arguments (-i, -v, and -f) as well as outputting an appropriate message based on the success, failure, or the detection of a problem.

For example, simply create a Perl program called "runme.pl" which accepts the standard Titan arguments (-i, -v, or -f) and put it into the Titan module directory. Running "Titan -f" would cause all the scripts that are in that directory – including your new one – to be executed with the "-f" flag.

It is imperative to keep in mind that if you write a Titan module it will be run as root and probably mangle the system in some fashion. Be careful with the code – it's easy to disable or otherwise make a system useless with a single errant character or a subtle logical error.

Porting Titan to Other OS's

Despite the fact that at present Titan only operates fully on Sun Microsystem's operating systems we feel that Titan could be useful with fairly minimal additional steps on other complex systems. To begin with, the basic framework of Titan runs on both main flavors of UNIX – the UCB (Solaris 1.1) and System V (Solaris 2.x) universes. Perhaps a third of all the Titan scripts would work or will work with minor tweaking on most out-of-the-box UNIX's. Armed with some basic shell scripting and a bit of security knowledge it would not be difficult to port a significant amount of the original Titan code to most systems. Of course, we would be happy to try to place OS specific code on our WWW site.

```
Check() {
  if [ -f /etc/init.d/init.dmi ]; then
    echo " dmi daemon is enabled: FAILS CHECK"
    exit 1
  else
    echo " dmi doesn't start at boot time: PASSES CHECK"
  fi
}
```

Figure 1: Simple fix application test.

```
if  [ -f /etc/default/kbd ] ; then
  echo "        Disabling the abort sequence "
  ed - /etc/default/kbd <<- !
  a
  KEYBOARD_ABORT=disabled
  .
  w
  q
  !
  echo "  Modifications to /etc/default/kbd complete"
fi
```

Figure 2: Simple disable script.

One of the most important parts of Titan, however, is its collection of little tricks and techniques that are unique to Solaris. The best place to begin amassing a collection of security tweaks for a different system is with the documentation and WWW site of the vendor of the system involved. Nearly all UNIX's have documentation with fairly good sections on security, and many put out security advisories when new problems crop up on the Internet. The BugTraq mailing list and http://www.rootshell.com are also excellent references, and it's usually possible to get the older advisories on-line too. Anything that can be typed in at the command line could be placed in a Titan module, including complete compiled or interpreted programs from any languages (C, perl, python, etc.).

Even Microsoft's Windows NT has the potential to be "Titanized." NT version 5 is supposed to come out with a scripting language based on ksh, and such rudimentary things as deleting all default share values, blocking the default guest accounts, and setting up a meaningful set of password and account management policies could be easily written. We are currently investigating HP-UX and Linux as new platforms.

Conclusions

Host-based security is NOT dead, even in the largest installations. Indeed, as organizations grow larger and their resources drop correspondingly they will be required to pick their security fights wisely – and we feel that Titan can be useful in protecting or evaluating the security of key systems, such as firewalls, production servers and other critical hosts.

Titan has been invaluable to us in our work as security professionals – running Titan improves the security of a system in almost all cases. And while it is not impossible to create tight, well-maintained, secure systems, it is, time consuming and very difficult! And it's easy to miss one or more of the many (sometimes crucial) details. Most professionals end up cobbling together various tools using ad hoc checklists when installing or auditing a system. Titan is well-suited for auditing systems as well as creating automated, formal checklists; if a technical security policy does not exist it can suggest the beginnings of one and either determine or force the adherence of a system to it.

It should be said, as disappointing as this may be, that the creation of a secure system is (and will probably always be) far more involved than simply running a computer program. Without a good security policy that is adhered to and a diligent and conscientious system administrative staff that keeps abreast of the latest security news and issues, Titan is relatively useless.

Titan's development future seems bright. Brad, both the originator and the main force behind the effort, has worked on and used Titan for many years, and has no plans to abandon it now. The other coauthors will be contributing as well, and while we all hope that the Internet community will give ideas or Titan modules, Titan's future is not dependent on that. Methods for backing out of the changes Titan makes, a simple GUI interface for policy management, and ports to other systems are all currently being investigated.

Security tools, from COPS, the TCP wrappers, Crack, Tiger, SATAN, to the many commercial security tools currently available, are invaluable to keeping systems monitored and secure. We respectfully put forth Titan as another freely available tool in the public security defense arsenal, and hope that it proves as valuable to you as it has been for us.

Availability

The latest version of Titan (currently 3.0) is available at: http://www.fish.com/security/titan.html. The authors can be contacted by email at: <titan@fish.com>.

Author Information

Dan Farmer performs security research at EarthLink Networks, Inc. In past lives he has authored or coauthored various security programs and papers, most notably the COPS and SATAN packages. He also worked for several years at Sun Microsystems with Brad and Matt, and can be reached via email at zen@fish.com.

Brad Powell has been in the Computer and Network Security field for over 10 years. As Senior Security Architect for Sun Professional Services, he designs security solutions including Firewalls, Security Architectures, and specialized security products for banks, industry, and government agencies.

Formerly Brad held the position of Network Security Engineer designing Sun's Firewall, security architecture, and network security policies. Duties also included electronic intrusion detection and prevention, and implementing security solutions on thousands of internal Sun networks, computing platforms, and applications, as well as assisting law enforcement agencies worldwide in investigating computer crime. Brad can be reached via email at Brad.Powell@sun.com.

Matthew Archibald left Sun Microsystems Inc. in 1992, after five years as a systems admin/security engineer. After some extended work outside of Sun Microsystems dealing in building and managing mixed-platform environments he joined Securix/Dynasoft in 1994 as a Security Consultant. Matthew subsequently returned to Sun Microsystems Professional Services for a short stay, providing similar services to international customers. Today he works as the Information & Networks Security Officer for KLA-Tencor corporation in Santa Clara CA. and can be reached via email at ir003355@mindspring. com or Matthew.Archibald@KLA-Tencor.COM.

Bibliography

The BugTraq mailing list. Currently bugtraq@ netspace.org.

http://www.rootshell.com .

Smashing The Stack For Fun And Profit by Aleph One, Phrack 49, Volume Seven, Issue Forty-Nine, File 14, November 08, 1996.

Sun Microsystems, Inc. System Manuals, Sun Microsystems, Inc., 1988-1998.

Sun Microsystems, Inc. Security Bulletin, Sun Microsystems, Inc., 1991-1998.