VIRUS ANALYSIS

GOT [MAC]ROOT?

Peter Ferrie
Symantec Security Response, USA

There is a long history of rootkits on Unix-based platforms, such as Unix itself, *Linux*, *BSD*, etc. No doubt to the surprise of some in the *Macintosh* community, the *MacOS X* platform now has one too. We call it OSX/Weapox. It is written by someone who calls himself 'nemo'.

Weapox is based very heavily on the AdoreBSD rootkit. In fact, some of the original Adore code remains in the Weapox binaries, although it is never called. Even the function names have been retained (since the *MacOS X* kernel-extension file format is an object file, a lot of textual information is visible, including the function names). Since the *MacOS X* platform essentially has *BSD* at its heart, it was not a surprise that a *BSD* rootkit was used as the basis for a *MacOS X* rootkit. Weapox did not load on a test machine running *Jaguar* (*MacOS 10.2*), but it did load on a test machine running *Panther* (*MacOS 10.3*).

Whenever the rootkit is executed, it begins by hooking the functions 'setuid', 'kill', 'write' and 'chmod'. The rootkit also contains hook functions for 'writev', and 'getdirentries', but since those functions are never hooked, the hook functions are never called. In any case, those hook functions seem to be incomplete, even though they are fully functional in the AdoreBSD rootkit. Perhaps *MacOS X* is sufficiently different that the Weapox author couldn't get them to work properly. Other functions that are not called are 'activate_cloaking' and 'hide_process'. The latter is another remnant from the original AdoreBSD rootkit.

EATS, ROOTS AND LEAVES

The hooked 'setuid' function checks if the UID is set to a particular value (1337 – 'leet'). If it is, then the function sets the UID to 0 (root) instead. Otherwise, the function calls the original handler. (Leetspeak [or 13375p34k] is a cryptic form of transliteration adopted by some hackers and gamers as a way of excluding the non-leet from their conversations on open channels. 1337 is devolved from the word 'elite' via 'lite' -> 'leet' -> '1337'.)

KILL OR BE KILLED

Despite its name, not only can the 'kill' function terminate a process, but it can also signal a process. If the hooked 'kill' function is used to signal a process, then the function checks if the signal is one of two particular values. If the value is 1337 ('leet' again), then the function escalates that process's privileges to level 0, effectively giving it full control over

the system. If the value is 9047 (which, in a less common use of 1337, means 'PORT'), then the passed PID is interpreted as a port number, and the function prepends that port to a list of ports to hide. That list is used by the hooked 'write' function (see below). If the signal is neither of these special values, then the hooked 'kill' function calls the original handler.

The hooked 'chmod' function checks whether the mode to set is a particular value (378, a value that would not normally be allowed). If it is 378, the passed path is interpreted as a logged-on username, and the function prepends that username to a list of usernames to hide. This list is used by the hooked 'write' function. Otherwise, the function calls the original handler.

This signalling method is interesting in that the hooked functions are in no way related to the information that they hide – but perhaps that's the idea.

WRITE YOUR OWN TICKET

The hooked 'write' function checks the name of the application requesting the write. If the requesting application is 'netstat', and if any entry in the hidden port list appears anywhere within the text to be printed, then the entire line is discarded instead of being printed. If the requesting application is 'w' or 'who', and if any entry in the hidden user list appears anywhere within the text to be printed, the entire line is discarded. This is a very simple method of stealth, which can be defeated, for example, by renaming the application, but it works well enough against the average user. It is also the method that AdoreBSD used.

The hooked 'writev' function checks if the text 'promiscuous mode' appears anywhere within the text to be printed. If it does, then the entire line is discarded instead of being printed. Otherwise, the function calls the original handler. It is not clear why this text would be ignored, unless perhaps it would otherwise appear in a network security log. In the AdoreBSD rootkit, the function is used as an 'I'm here' routine – literally, if the hooked 'writev' function is called and if the text 'promiscuous mode' appears anywhere within the text to be printed, AdoreBSD prints 'I'm here'. Otherwise, the function calls the original handler.

The hooked 'getdirentries' function contains several bugs, one of which results in an infinite loop, but that doesn't matter since the function is never called. The function is intended to check for a particular directory name within a directory structure, presumably to avoid it being printed. However, the directory name is never copied to the buffer to compare. Even if the name matched, the function simply prints 'MATCH!' and does nothing further with it. Additionally, the original function is not called afterwards.

The AdoreBSD code, for comparison, calls the original function to retrieve the real list of directory entries, then removes the directory to hide from that list, before returning the list to the caller.

The 'active_cloaking' function is intended to remove the current module from the kernel module list. This list is used by, for example, kextstat. It is not clear, though, if the removal from the list results in the process no longer being executed. Perhaps that is why it is not called. The 'hide_process' function doesn't hide anything at all. It simply searches for the requested PID and returns success or failure. The rest of the code that was present in the corresponding AdoreBSD rootkit function has been removed from this function.

OPEN SESAME

A 'rootkit' called SH/Renepo ('opener' spelled backwards) preceded Weapox by a few months. (In fact it was less of a rootkit and more a collection of hacking tools.) The package contained a script and three binaries. The script displayed some messages, including user information and passwords. It added a new user to the system, turned off the firewall, and attempted to terminate the LittleSnitch process (a program that tells the user when a program is attempting to send information to the Internet). It downloaded and installed the rest of the package, started a backdoor, created a screen dump, and deleted its temporary file.

The first binary was a 'backdoor' for the xinetd program. It simply ran xinetd with a custom configuration file that caused xinetd to listen on port 31337 ('eleet', surprise!), and return a command-shell with root privileges on connection. The second binary was the well-known netcat program, a very useful tool for doing all kinds of network-related things. The third binary was a *Unix* log-file cleaner. There was no active stealth technology at all in the package.

CONCLUSION

There's always one person who spoils it for everyone else. The *Macintosh* community has been relatively unaffected by recent malware, at least when compared to the *Windows* community, but perhaps that is set to change after all.

OSX/Weapox	
Size:	27,608 bytes
Type:	Rootkit
Payload:	None
Removal:	kextunload, then delete the files.