BackDoor.Gootkit.112—a new multi-purpose backdoor

news.drweb.com/show/

Doctor Web

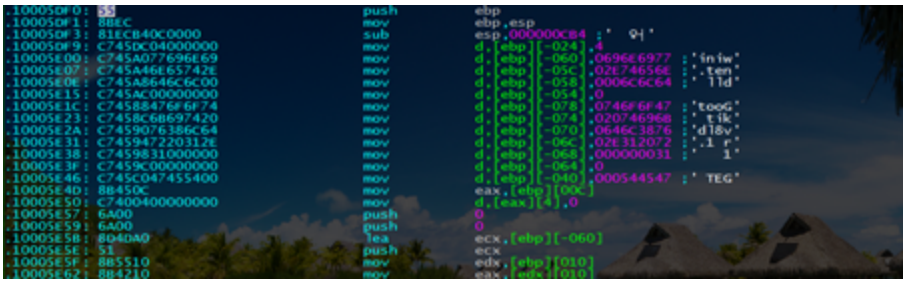
Back to news

April 9, 2014

Complex multi-component Trojans with backdoor features, i.e., those capable of executing a remote server's commands on an infected computer, are rarities in the wild. Doctor Web's analysts recently examined one such program that has been named BackDoor.Gootkit.112. This review provides information about this malicious program's design and operation.

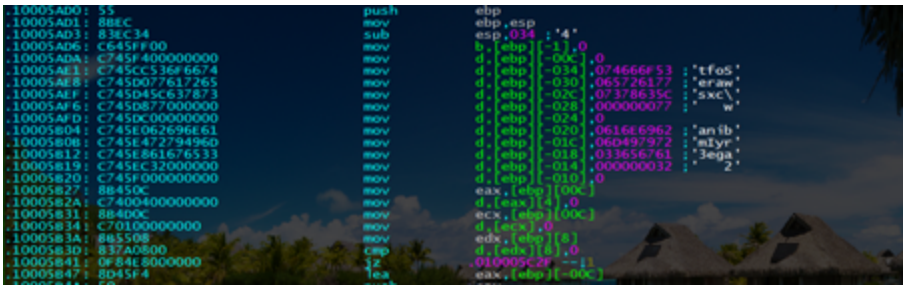
Apparently, the module responsible for installing the backdoor into the system and for its bootkit features was borrowed by **BackDoor.Gootkit.112**'s developers from the Trojan.Mayachok family of programs. However, the virus writers introduced a number of significant changes into the source code. The original Trojan.Mayachok generated a unique VBR code which was used to create another build of the malware. In the case of **BackDoor.Gootkit.112**, all the functions have been grouped in the dropper, which alters the Volume Boot Record (VBR) code during the infection process. The driver, to which control is transferred by the VBR code prior to system initialisation, was also taken from the Trojan.Mayachok source code, but the code was partially rewritten, so most of the pointers

(the shell-code to perform injections, and various tables) have been changed for reasons unknown. However, some pointers remained intact. In particular, one of them refers to the Homer Simpson quotation "Just pick a dead end and chill out till you die", which is output in the debugger after the loader's initialisation. It is noteworthy that similar strings (mostly Homer Simpson quotations) were displayed in the debugger by TDSS Trojans (starting with BackDoor.Tdss.565 (TDL3) and older versions). The name Gootkit can be found in both the loader and the payload module code.



In addition, all the driver components responsible for its interaction with other components operating in the user mode were also removed—in particular, the driver that enables them to use VFS. However, **BackDoor.Gootkit.112** has features responsible for VFS initialisation and protection.

Information about the payload module **BackDoor.Gootkit.112** is stored in the Windows registry branch HKLMSOFTWARECXSW as binaryImage32 or binaryImage64, depending on the OS platform (32- or 64-bit).



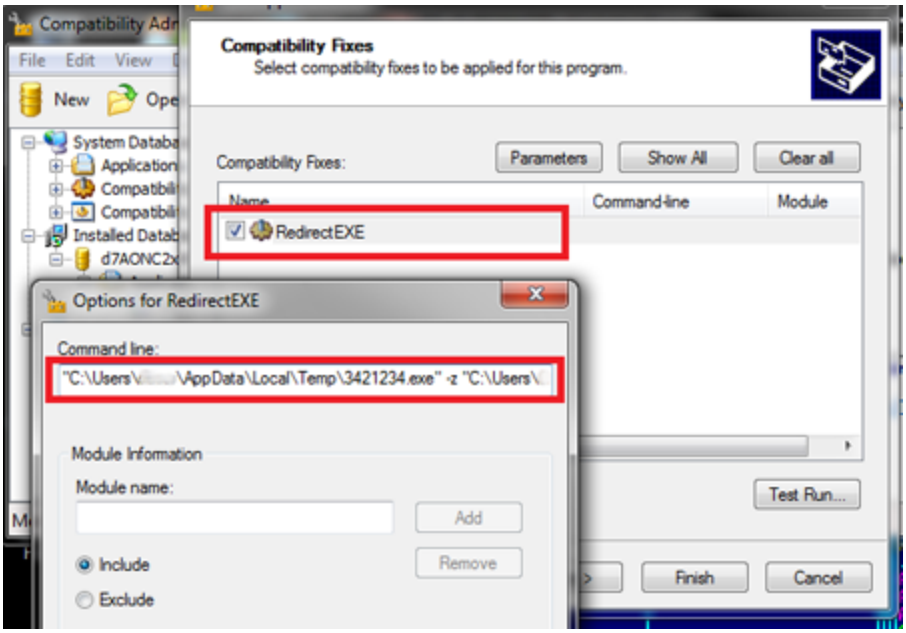
To retrieve the payload, **BackDoor.Gootkit.112** injects special shell code into the processes SERVICES.EXE, EXPLORER.EXE, IEXPLORE.EXE, FIREFOX.EXE, OPERA.EXE, and CHROME.EXE. Very few malicious programs inject their code by creating a new user mode thread involving CSRSS.EXE.

The main objective of the injected shell code is to download the payload module from the system registry or from a remote server on the Internet. Payload binary files are compressed and encrypted.

To bypass the UAC and elevate its privileges in an infected system, **BackDoor.Gootkit.112** employs a shim (Microsoft Windows Application Compatibility Infrastructure). The Trojan employs the SQL Server Client Network Utility (cliconfg.exe) whose manifest file has the

attribute AutoElevate set to true, so Windows elevates the privileges of such applications without involving the UAC.

BackDoor.Gootkit.112 uses the file apphelp.dll to create a fix database. The Trojan generates the database's name and the value of the Application parameter randomly. To load the Trojan code, it uses the routine RedirectEXE, which lets one executable be run instead of another one. **BackDoor.Gootkit.112** uses RedirectEXE parameters to specify the path to its executable and a link to the created database.



After that, the fix database (shim) is installed in the system by means of **sdbinst.exe** whose manifest also has the parameter AutoElevate set to true, so it runs on Windows with special privileges. Overall, the UAC bypass scheme looks as follows:

1. The Trojan creates and installs a new fix database (shim);
2. It then launches cliconfg.exe with elevated privileges;
3. The shim unloads the original process and uses RedirectEXE to launch the Trojan.

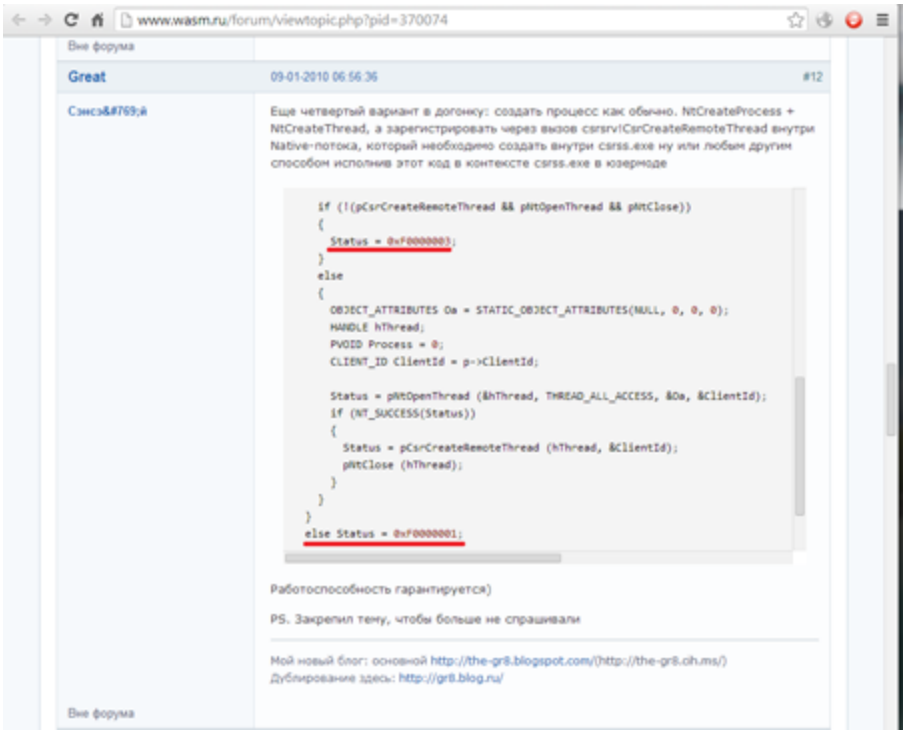
BackDoor.Gootkit.112's payload is implemented in a large, five megabyte executable written in C++. Most of this file is a JavaScript interpreter known as Node.JS. The executable file contains more than 70 pieces of JavaScript code. A significant portion of them constitutes the Node.JS core which provides an easily accessible interface to work with native objects. Some scripts incorporate the Trojan's payload: they enable the backdoor to execute commands from a remote server and download additional modules stored in the Windows registry, similarly to the main module of **BackDoor.Gootkit.112**. The Trojan can execute the following commands:

- Intercept http traffic;
- Inject code into other processes;
- Block specific URLs;

- Take screenshots;
- Acquire the list of running processes;
- Acquire the list of local users and groups;
- End specified processes;
- Execute shell commands;
- Launch executables;
- Auto update.

and some other.

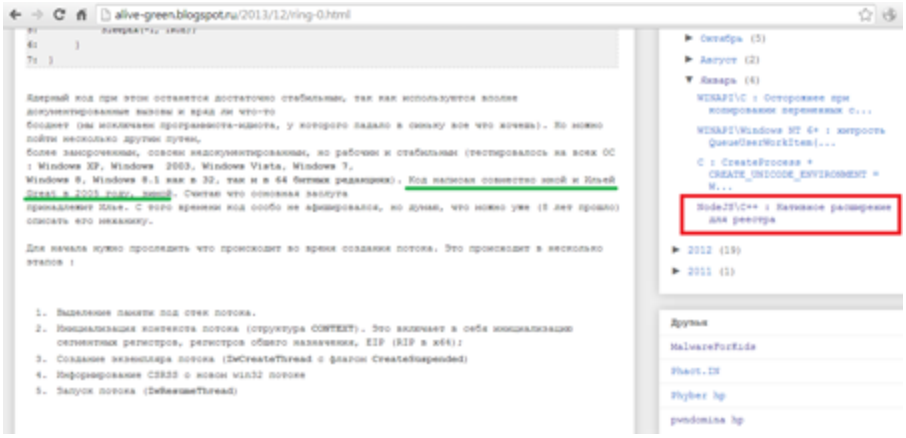
As mentioned above, the program uses a rare method for injecting code into running processes. A similar algorithm was described on the forum wasm.ru by a user with the alias Great:



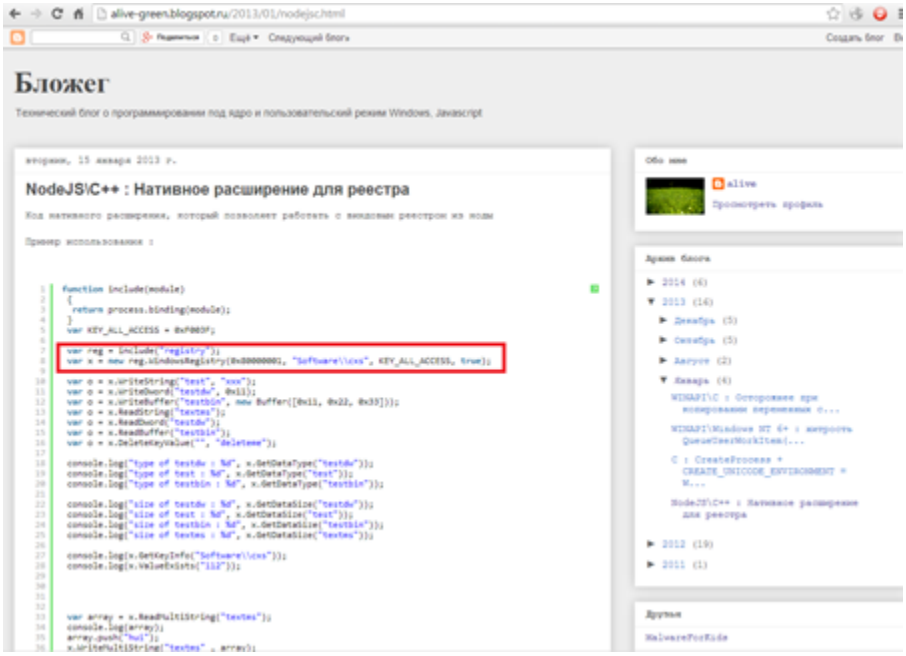
His description contained exit statuses which were similar to those found in the disassembled code of **BackDoor.Gootkit.112**:

```
hCSRSRV = get_module_handle(&szCSRSRV_dll);
hNTDLL = get_module_handle(&szNTDLL_DLL);
if ( hCSRSRV && hNTDLL )
{
  CsrCreateRemoteThread = (int (__stdcall *)(int, int *))get_proc_addr_by_hash(hCSRSRV, 0x5846B885);
  NtOpenThread = (int (__stdcall *)(int *, signed int, OBJECT_ATTRIBUTES *, int *))get_proc_addr_by_hash(
                                                                                    hNTDLL,
                                                                                    0xF88A31D1);
  NtClose = (void (__stdcall *)(int))get_proc_addr_by_hash(hNTDLL, 0xBB8E133D);
  NtTerminateThread = get_proc_addr_by_hash(hNTDLL, 0xAC3C9DC8);
  if ( CsrCreateRemoteThread && NtOpenThread && NtClose )
  {
    v1 = *(_DWORD *)(inj_context + 8);
    from_ctx = *(_DWORD *)(inj_context + 4);
    from_ctx2 = v1;
    status = NtOpenThread(&hThread, 0x1FFFFF, &oa, &from_ctx);
    if ( !status )
    {
      status = CsrCreateRemoteThread(hThread, &from_ctx);
      NtClose(hThread);
    }
  }
  else
  {
    status = 0xF0000003;
  }
}
else
{
  status = 0xF0000001;
}
if ( NtTerminateThread )
  ((void (__stdcall *)(_DWORD, int))NtTerminateThread)(0, status);
while ( 1 )
  ;
}
```

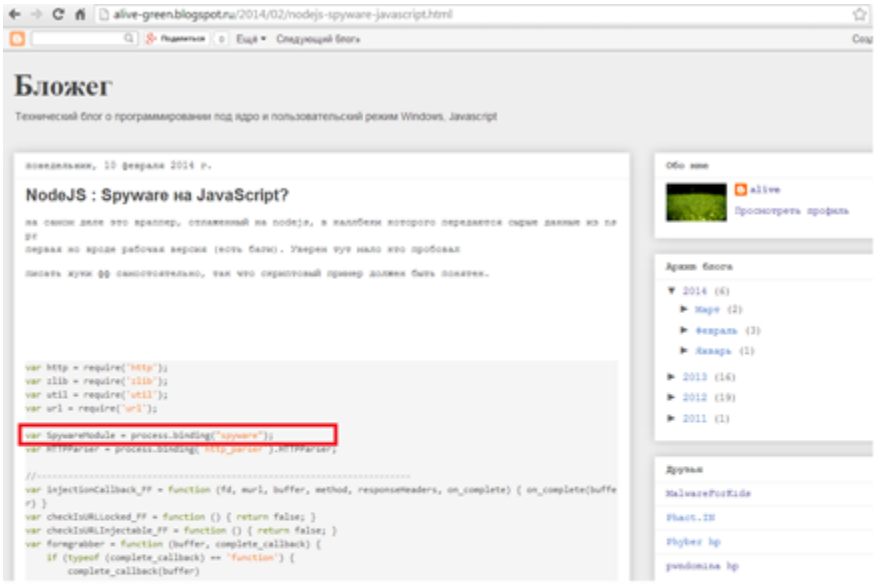
One would assume that the virus writer simply borrowed code from the public source, but the code posted on the forum also described the object called **DRIVER_TO_SHELLCODE_PARAMETERS**. An object with the same name was also discovered in a personal blog of another user who provided a detailed description of the injection method and claimed that he developed it in cooperation with Ilya Great:

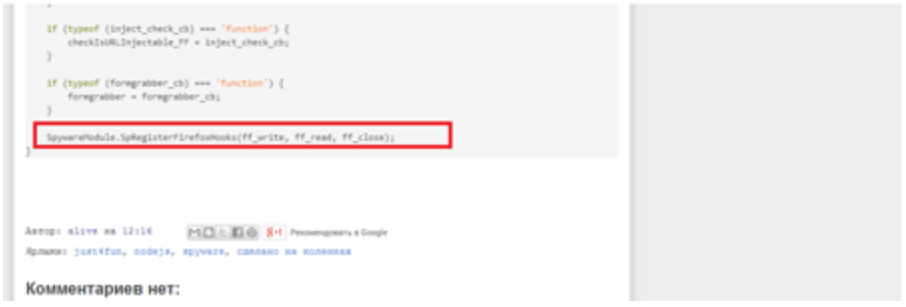


The blogger also expressed his great interest in Node.JS whose features are used extensively in the Trojan's code. Moreover, the person also published a post entitled "NoteJS\C++: Native extension for the Registry" in which he described a method for working with the Windows registry branch **SOFTWARE\CXS**:

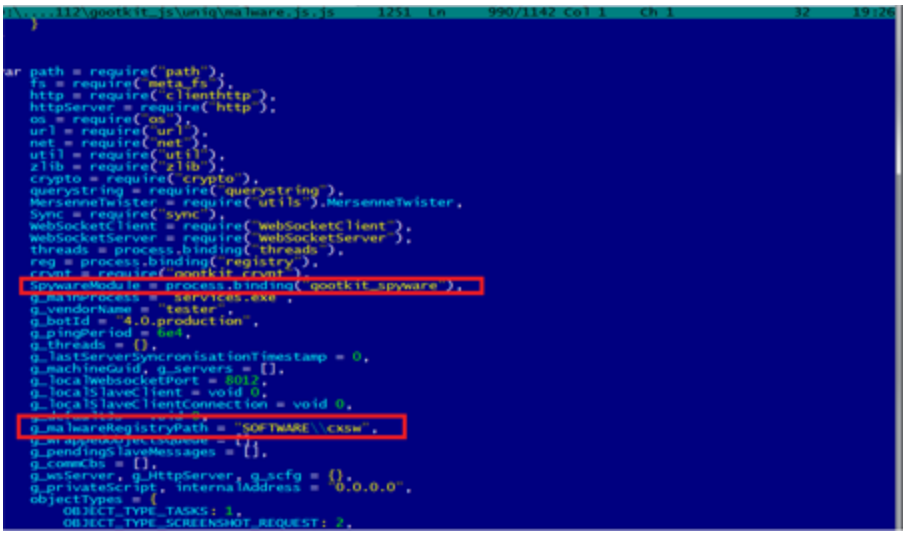


Another post of his, entitled "NodeJS: Spyware in Javascript?", contains a reference to **SpywareModule** whose methods incorporate the prefix 'Sp'.





BackDoor.Gootkit.112 incorporates similar code.



In this regard, one can make assumptions regarding the actual person behind the backdoor with a high degree of certainty.

BackDoor.Gootkit.112's signature has been added to the Dr.Web virus database, and, therefore, the Trojan poses no threat to computers protected with Dr.Web.

What is the benefit of having an account?

Tell us what you think

To ask Doctor Web's site administration about a news item, enter @admin at the beginning of your comment. If your question is for the author of one of the comments, put @ before their names.

Other comments