Winnti Group's skip-2.0: A Microsoft SQL Server backdoor

welivesecurity.com/2019/10/21/winnti-group-skip2-0-microsoft-sql-server-backdoor

October 21, 2019

Notorious cyberespionage group debases MSSQL

Mathieu Tartare 21 Oct 2019 - 11:30AM

For a while, ESET researchers have been tracking the activities of the Winnti Group, active since at least 2012 and responsible for high-profile supply-chain attacks against the video game and software industry. Recently, we discovered a previously undocumented backdoor targeting Microsoft SQL (MSSQL) that allows attackers to maintain a very discreet foothold inside compromised organizations. This backdoor bears multiple similarities to the *PortReuse* backdoor, another tool used by the Winnti Group that was first documented by ESET in October 2019, such as the use of the same custom packer and VMProtected launcher, which is why we attribute this backdoor to the Winnti Group.

Earlier this year, we received a sample of this new backdoor called *skip-2.0* by its authors and part of the Winnti Group's arsenal. This backdoor targets MSSQL Server 11 and 12, allowing the attacker to connect stealthily to any MSSQL account by using a magic password – while automatically hiding these connections from the logs. Such a backdoor could allow an attacker to stealthily copy, modify or delete database content. This could be used, for example, to manipulate in-game currencies for financial gain. In-game currency database manipulations by Winnti operators have already been reported. To the best of our knowledge, *skip-2.0* is the first MSSQL Server backdoor to be documented publicly. Note that even though MSSQL Server 11 and 12 are not the most recent versions (released in 2012 and 2014, respectively), they are the most commonly used ones according to Censys's data.

We recently published a white paper updating our understanding of the arsenal of the Winnti Group, and that exposed a previously undocumented backdoor of theirs called *PortReuse.* It uses an identical packer to that used with the payload embedded in compromised video games uncovered by ESET in March 2019. The VMProtected launcher that drops the *PortReuse* backdoor was also found being used to launch recent *ShadowPad* versions. In that context, we were able to find a new tool called *skip.2-0* by its developer. It uses the same VMProtected launcher as well as Winnti Group's custom packer and exhibits multiple similarities with other samples from the Winnti Group's toolset. This leads us to ascribe *skip-2.0* to that toolset also.

This article will focus on the technical details and functionality of this MSSQL Server backdoor, as well as on exposing the technical similarities of *skip.2-0* with the Winnti Group's known arsenal – in particular, with the *PortReuse* backdoor and *ShadowPad*. A note on the reasons why we chose the "Winnti Group" naming can be found on our white paper.

VMProtected launcher

We found *skip-2.0* while looking for VMProtected launchers, for which the payload is usually either *PortReuse* or *ShadowPad*.

Embedded payload

As with the encrypted *PortReuse* and *ShadowPad* payloads, *skip-2.0* is embedded in the VMProtected launcher's overlay, as shown in Figure 1:

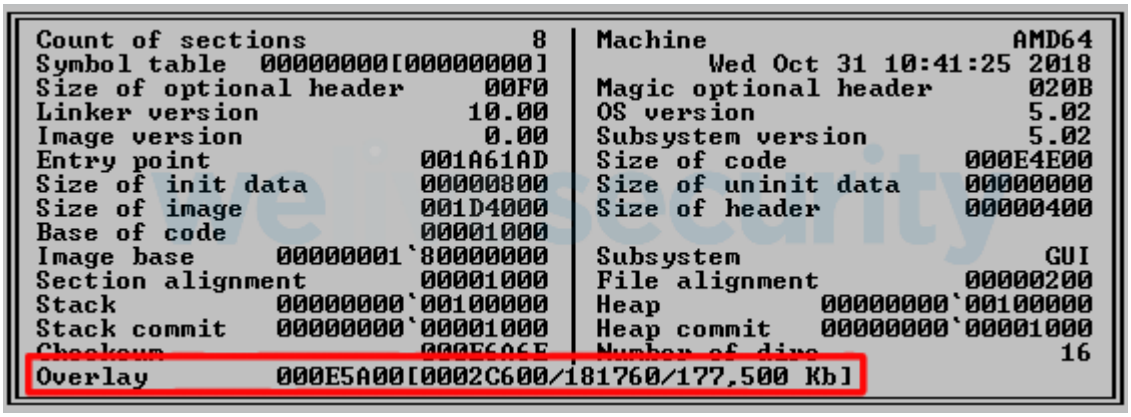

Figure 1. VMProtected launcher's headers. The payload is embedded in the PE overlay.

Encryption

The payload encryption is identical to that used in the other VMProtected launchers. It is RC5-encrypted with a key derived from the VolumeID and the string f@Ukd!rCto R$., as described in our previous white paper on the Winnti Group arsenal.

Persistence

As in the case of *PortReuse* and *ShadowPad*, the launcher probably persists by exploiting a DLL hijacking vulnerability by being installed at C:\Windows\System32\TSVIPSrv.DLL. This results in the DLL being loaded by the standard Windows SessionEnv service at system startup.

Winnti Group's custom packer

Once decrypted the embedded payload is actually Winnti Group's custom packer. This packer is the same shellcode that was documented in our previous article and white paper. It is used to pack the *PortReuse* backdoor as well as the payload embedded in the compromised video games.

Packer configuration

As described in our previous article, the packer configuration contains the decryption key of the packed binary as well as its original filename, its size and the execution type (EXE or DLL). The payload's packer configuration is shown in Table 1.

Parent SHA-1	Payload SHA-1	RC4 key	Filename	Launch type

Parent SHA-1	Payload SHA-1	RC4 key	Filename	Launch type
9aafe81d07b3e5b-b282608f0a2a4656eb485b7c9	a2571946ab181657e-b825cde07188e8bcd689575	163716559	Inner-Loader.dll	2

Table 1. Payload's packer configuration

One can see from the packer configuration that the payload is called *Inner-Loader*. *Inner-Loader* is the name of an injector that is the part of the Winnti Group's arsenal used to inject the *PortReuse* backdoor into processes listening on a particular port, as described in our previous publication. Beyond that identical name, by analyzing this payload it appears that it is another variant of the *Inner-Loader* injector.

Inner-Loader injector

This variant of *Inner-Loader*, instead of looking for a process listening on a particular port, as in the case when injecting the *PortReuse* backdoor, looks for a process called sqlserv.exe, which is the conventional process name of MSSQL Server. If found, *Inner-Loader* then injects a payload into this process. This payload is also packed with the custom packer – the packer configuration of that payload is shown in Table 2.

Parent SHA-1	Payload SHA-1	RC4 key	File-name	Launch type
a2571946ab181657e-b825cde07188e8bcd689575	60b9428d00be5ce562f-f3d888441220290a6dac7	923567961	skip-2.0.dll	2

Table 2. Packer configuration of the payload embedded in Inner-Loader

The original filename of this injected payload is *skip-2.0.dll*.

skip-2.0

After having been injected and launched by *Inner-Loader*, *skip-2*.0 first checks whether it is executing within an sqlserv.exe process and if so, retrieves a handle to sqllang.dll, which is loaded by sqlserv.exe. It then proceeds to find and hook multiple functions from that DLL. Figure 2 depicts the *skip-2.0* chain of compromise.

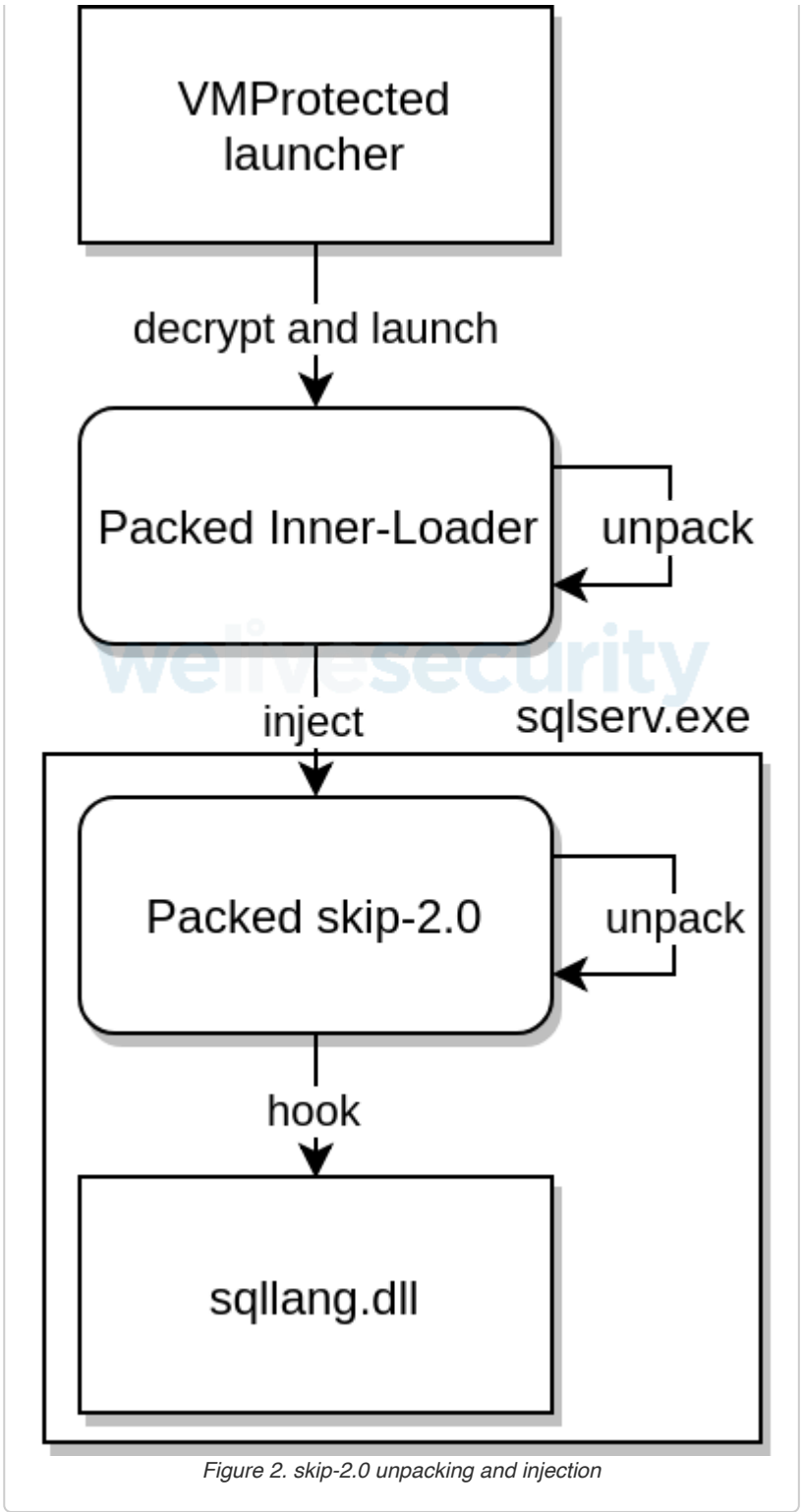

Figure 2. skip-2.0 unpacking and injection

Hooking sqllang.dll

The hooking procedure used by *skip-2.0* is very similar to the one used by *NetAgent*, the *PortReuse* module responsible for installing the networking hook. This hooking library is based on the distorm open source disassembler that is used by multiple open source hooking frameworks. In particular, a disassembling library is needed to correctly compute the size of the instructions to be hooked. One can see in Figure 3 that the hooking procedure used by *NetAgent* and *skip-2.0* are almost identical.

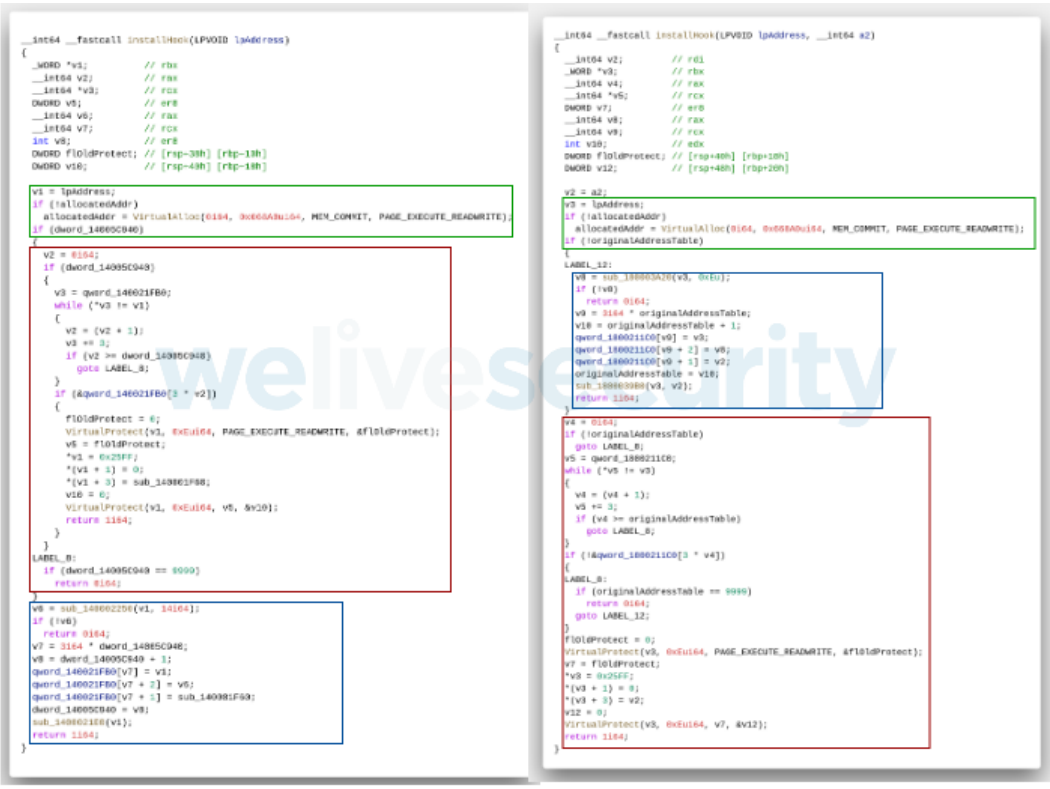

Figure 3. Hex-Rays output comparison between the NetAgent (left) and skip-2.0 (right) hooking procedures

There is one notable difference, which is the fact that the hooking function from *skip-2.0* takes the address of the hook to be installed as argument, while for *NetAgent*, the address of the hook to install is hardcoded. This is due to the fact that *skip-2.0* has to hook multiple functions in sqllang.dll to operate properly, while *NetAgent* only targets a single function.

To locate each sqllang.dll functions to be hooked, *skip-2.0* first retrieves the size of the DLL once loaded in memory (i.e. its virtual size) by parsing its PE headers. Then an array of bytes to be matched within sqllang.dll is initialized as shown in Figure 4. Once the address of the first occurrence matching the byte array is found, the hook is installed using the procedure shown in Figure 3.

```
bytePattern[0] = 0x5655F3FF;
bytePattern[1] = 0x48564157;
bytePattern[2] = 0x1C0EC81;
bytePattern[3] = 0xC7480000;
bytePattern[4] = 0xFE382444;
LOWORD(bytePattern[5]) = 0xFFFF;
BYTE2(bytePattern[5]) = 0xFF;
funcAddr = findBytePattern(MZ, virtualSize, bytePattern, 23);
if ( funcAddr )
{
  success = installHook(funcAddr, hookFunc_1);
```

Figure 4. Hex-Rays output of the procedure initializing the byte array to match in sqllang.dll

The success of the hook installation is then logged in cleartext in a log file located at the hardcoded path C:\Windows\Temp\TS_2CE1.tmp and shown in Figure 5.

```
2019-09-17 08:56:14.1.0, _Info_       S         WS___[9250] ******************
2019-09-17 08:56:14.2.1, _Info_       F         WS___[00000868:00000872] ******************
2019-09-17 08:56:14.3.0, _Info_       S         WS___[DE30] ******************
2019-09-17 08:56:14.4.0, _Info_       S         WS___[0DA0] ******************
```

Figure 5. Log generated during hooks installation

Should the targeted function not be found, the hook installer searches for a fallback function, with a different set of byte patterns.

Matching a sequence of bytes to locate the address of the targeted function instead of using a static offset, plus as using a fallback sequence of bytes, allows *skip-2.0* to be more resilient to MSSQL updates and to potentially target multiple sqllang.dll updates.

One password to rule them all

The functions targeted by *skip-2.0* are related to authentication and event logging. The targeted functions include:

- CPwdPolicyManager::ValidatePwdForLogin
- CSECAuthenticate::AuthenticateLoginIdentity
- ReportLoginSuccess
- IssueLoginSuccessReport
- FExecuteLogonTriggers
- XeSqlPkg::sql_statement_completed::Publish
- XeSqlPkg::sql_batch_completed::Publish
- SecAuditPkg::audit_event::Publish
- XeSqlPkg::login::Publish
- XeSqlPkg::ual_instrument_called::Publish

The most interesting function is the first one (CPwdPolicyManager::ValidatePwdForLogin), which is responsible for validating the password provided for a given user. This function's hook checks whether the password provided by the user matches the magic password, in that case, the original function will not be called and the hook will return 0, allowing the connection even though the correct password was not provided. A global flag is then set that will be checked by the other hooked functions responsible for event logging. The corresponding decompiled procedure is shown in Figure 5. In the case where this global flag is set, the hooked logging functions will silently return without calling their corresponding, original functions, so the action will not be logged. In the case where a different password is provided, the original function is called.

```
memmove(&Buf1, password, passwordLength);
TlsSetValue(magicPasswordUsed_1, 0i64);
magicPasswordUsed = 0;
strlen = lstrlenW(L"[redacted]");
if ( !memcmp(&Buf1, L"[redacted]", 2 * strlen) )
{
  magicPasswordUsed = 1;
  TlsSetValue(magicPasswordUsed_1, 1);
  result = 0i64;
}
```

Figure 6. Hex-Rays output of the procedure responsible for matching the password provided at login with the hardcoded string

A similar backdooring technique, based on hardcoded passwords, was used with SSH backdoors previously discovered by ESET. The difference here is that *skip-2.0* is installed in-memory, while in the case of the SSH backdoors the sshd executable was modified prior to execution.

Additionally, CSECAuthenticate::AuthenticateLoginIdentity will be called from within its hook code but the hook will always return 0. The ReportLoginSucess and IssueLoginSuccessReport hooks will not call the original functions if the magic password was used to log in. The same behavior is applied to FEExecuteLogonTriggers. Other logging functions such as XeSqlPkg::sql_statement_completed::Publish or XeSqlPkg::sql_batch_completed::Publish will also be disabled in the case where the user logged in with the magic password. Multiple audit events are disabled as well, including SecAuditPkg::audit_event::Publish, XeSqlPkg::login::Publish and XeSqlPkg::ual_instrument_called::Publish.

This series of hooks allows the attacker not only to gain persistence in the victim's MSSQL Server through the use of a special password, but also to remain undetected thanks to the multiple log and event publishing mechanisms that are disabled when that password is used.

We tested *skip-2.0* against multiple MSSQL Server versions and found that we were able to login successfully using the special password with MSSQL Server 11 and 12. To check whether a particular sqllang.dll version is targeted by *skip-2.0* (i.e., that matches the byte patterns), we created a YARA rule, which can be found in our GitHub repository.

Connection with the Winnti Group

We observed multiple similarities between *skip-2.0* and other tools from the Winnti Group's arsenal. Its VMProtected launcher, custom packer, *Inner-Loader* injector and hooking framework are part of the already known toolset of the Winnti Group. This leads us to think that *skip-2.0* is also part of that toolset.

Conclusion

The *skip-2.0* backdoor is an interesting addition to the Winnti Group's arsenal, sharing a great deal of similarities with the group's already known toolset, and allowing the attacker to achieve persistence on an MSSQL Server. Considering that administrative privileges are required for installing the hooks, *skip-2.0* must be used on already compromised MSSQL Servers to achieve persistence and stealthiness.

We will continue to monitor new activities of the Winnti Group and will publish relevant information on our blog. For any inquiries, contact us at threatintel@eset.com.

Indicators of Compromise (IoCs)

Component	SHA-1	ESET detection name
VMP Loader	18E4FEB988CB95D71D81E1964AA6280E22361B9F 4AF89296A15C1EA9068A279E05CC4A41B967C956	Win64/Packed.VMProtect.HX
Inner-Loader injector	A2571946AB181657EB825CDE07188E8BCD689575	Win64/Injector.BS
skip-2.0	60B9428D00BE5CE562FF3D888441220290A6DAC7	Win32/Agent.SOK
Known targeted sqllang.dll files (non-exhaustive list)	4396D3C904CD340984D474065959E8DD11915444 BE352631E6A6A9D0B7BBA9B82D910-FA5AB40C64E D4ADBC3F77ADE63B836FC4D9E5915A3479F09B-D4 0BBD3321F93F3DCD-D2A332D1F0326142B3F4961A FAE6B48F1D6ED-DEC79E62844C444FE3955411EE3 A25B25FFA17E63C6884E28E96B487F58DF4502E7 DE76419331381C390A758E634BF2E165A42D4807 ED08E9B4BA6C4B5A1F26D671AD212AA2F-B0874A2 1E1B0D91B37BAEBF77F85D1B7C640B8C-C02FE11A 59FB000D36612950FEBC36004F1317F7D000AA0B 661DA36BDD115A1E649F3AAE11AD6F7D6FF2D-B63	N/A

MITRE ATT&CK techniques

Tactic	ID	Name	Description
Execution	T1035	Service Execution	*skip-2.0* is started with the SessionEnv service
Persistence	T1038	DLL Search Order Hijacking	*skip-2.0* probably uses a DLL hijacking technique against the SessionEnv service
	T1179	Hooking	*skip-2.0* hooks multiple functions in sqllang.dll service to bypass authentication and maintain stealth
Defense Evasion	T1054	Indicator Blocking	*skip-2.0* blocks event logging
	T1045	Software Packing	*skip.2-0* and *Inner-Loader* are packed using Winnti's custom packer. Further, the launcher is VMProtected.
Discovery	T1057	Process Discovery	*Inner-Loader* lists running processes in order to find the process running MSSQL Server
Impact	T1485	Data Destruction	*skip-2.0* allows unauthorized access to MSSQL databases, allowing data destruction or tampering
	T1494	Runtime Data Manipulation	*skip-2.0* manipulates event logging at runtime

Tactic	ID	Name	Description
	T1492	Stored Data Manipulation	*skip-2.0* allows unauthorized access to MSSQL databases, allowing manipulation of stored data

Mathieu Tartare 21 Oct 2019 - 11:30AM