Grasshopper v1.1Users' Guide

December 2013

CL BY: 2355679
CL REASON: Section 1.5(c),(e)
DECL ON: 20370522
DRV FRM: COL 6-03

1 Overview

Grasshopper is a modular, rule-based soft persistence installer. This document will provide information relevant to the secure and effective use of the Grasshopper tool suite.

1.1 Concept of Operations

Grasshopper is a suite of tools used to install a payload to a target computer running the Microsoft Windows operating system. Grasshopper will build installer executables that pair payloads with one or more persistence modules, representing different soft persistence mechanisms. Rules may be written to determine when to install the payload with what module. The installer will collect data from the target computer to evaluate each rule.

Grasshopper implements a defensive deployment strategy. The installers follow a "look before you leap" approach by evaluating the conditions of the target before installing their payloads.

1.2 System Components

The Grasshopper tool suite consists of five component types: Installers, Persistence Modules, Payloads, Builders, and Post Processors.

1.2.1 Installers

Installers are executables responsible for loading and executing persistence modules. Once a payload has been installed using one of these modules, the Installer removes itself from the target.

Installer executables may be built as .EXE's or .DLL's in 32- or 64-bit. Produced DLLs confirm to both F&F version 2 and ICE version 3. The executables are configured using an Installer-specific builder. Configured payloads and persistence modules are obfuscated and packed as resources in the installer executable.

Grasshopper

The Grasshopper Installer evaluates the conditions on a target machine to select an appropriate persistence module from an assortment of modules before using it to install an appropriate payload. Selection of persistence module and payload is based on rules specified by the operator at build time.

Grasshopper may be configured to generate a log file that records its activity, including any data it collected and the return codes of each persistence module it executed.

Cricket

The Cricket Installer uses a given persistence module to install a given payload. The operator is responsible for selecting the appropriate module and payload for the target.

1.2.2 Persistence Modules

Persistence Modules are responsible for installing Payloads to a target system. Each module represents a different soft persistence mechanism. Persistence Modules are specified by catalog entries that describe the module and identify its components.

Persistence module details and usage are described in module-specific user guides.

1.2.3 Payloads

Payloads are applications that can be persisted using Grasshopper. Payloads are specified by catalog entries that provide information about thepayload to the builder utilities.

New payloads may be added to Grasshopper by writing and adding an entry to the catalog. Operators must provide a configured payload binary at build time.

1.2.4 Builders

Builders are Python scriptsused tocreate installer executables.

Builders take input from the operator to select payloads and persistence methods, configure an installer and persistence modules, and pack payloads and modules into an installer. A receipt is generated to document the configuration and save copies of the binaries used in the build.

Separate Builderscripts are provided for each installer type.

1.2.5 Post Processor

The Post Processor is a Python script used to decode binary log files generated by a Grasshopper installer.

The Post Processor requires a Grasshopper receipt and the encoded log from its installation. It will generate a decoded log file documenting the actions of the installer and an xml file containing all of the target survey data collected during installation and result codes for each installation attempt.

1.3 System Requirements

1.3.1 Python

The Grasshopper utilities are written for Python v3.3. Their compatibility with other versions has not been tested and is not assured. Unless otherwise stated, the scripts may run on any platform and operating system that runs a Python interpreter.

All Grasshopper utilities are dependent on the provided Grasshopper Python package, named `'grasshopper'`. The package must be placed within one of Python's path resolution directories, which includes the directory of the script executed.

2 Grasshopper

The Grasshopper Installer is the primary installer in the Grasshopper tool suite. It is the recommended mechanism for running Grasshopper persistence modules.

The Installer evaluates the conditions on a target machine to select an appropriate payload and persistence module combination. Selection of the payload and persistence module is based on rules specified by the operator at build time.

2.1 Usage

There are two ways to run a Grasshopper Installer, represented by two types of executables: EXEs and DLLs.

2.1.1 Executable

The Grasshopper EXE is a Windows application and may be invoked as such.

The executable does not take any arguments on the command line. All configuration and options are decided by the operator and set at build time.

2.1.2 Dynamic Load Library

The Grasshopper DLL is a Windows dynamic load library that implements the `MRC_FireAndForget` interface for the in-memory execution of DLLs.

The entry-point does not take any arguments through the exported function. All configurations are decided by the operator and set at build time.

2.2 Behavior

The Grasshopper installer executes the following steps after gaining execution.

1) Select payload and persistence module

   Grasshopper uses a rule-based decision engine to select a target appropriate payload-persistence module combination.

2) Load persistence module

   Grasshopper loads the selected persistence module in memory. The module is initialized using the user-defined configuration.

2) Run persistence module

   Grasshopper runs the persistence module against the selected payload. The module is responsible for the deployment and execution of the payload.

3) Cleanup and Exit

   Grasshopper cleans up after its activity and exits. The Grasshopper EXE cleans itself up by initiating a self delete before exiting; the DLL is run from memory and does not need to self-delete.

2.3 Decision Engine

The Grasshopper Installer uses a simple decision engine to select the appropriate payload and persistence module combination.

2.3.1 Algorithm

The decision engine is configured by defining an ordered list of payloads and, for each payload, an ordered list of persistence modules. Both the payload list and persistence module lists are ordered based on preference.

During operation, the decision engine evaluates the global rule, along with each payload in the payload list with each module in the payload-specific persistence module list. If the decision engine positively evaluates the combination of payload and module, it will attempt to install the payload using the persistence module.

If the installation attempt fails, the decision engine resumes evaluating payload-module pairs where it left off until no pairs remain.

Pseudo Code

```
ifrule_fails( global.rule ): return Failed
for each payload in payload_list:
             for each module in payload.module_list:
                 if evaluate( payload.rule ) is True and evaluate( module.rule ) is
True:
                       ifmodule.install( payload ) is Successful:
                              return Done
return Failed
```

2.3.2 Target Survey

Payloads and persistence modules are selected for fitness by checking conditions on the target machine. The primary logical unit of the decision engine are 'facts', simple statements about the target. Facts are evaluated by surveying the target at run time.

Once evaluated, facts are stored in the decision engine cache, such that survey operations are not repeated unnecessarily.

The decision engine evaluates facts as:

true	the fact is a true statement
false	the fact is a false statement
invalid	the fact cannot be evaluated

For example, a fact may be written to check for the existence of a file/registry key/running process that indicates the presence of a PSP. During operation, Grasshopper will check for that file/registry key/process and set that fact accordingly.

2.3.3 Rule-based Evaluation

Rules are descriptions of the conditions on the target required for successful operation of a payload or persistence module. A rule is the combination of facts usingboolean operators to create complex expressions.

The combinations of payload and persistence module are evaluated using rules associated with each payload and module. Both the payload and persistence module rule must evaluate to True for the decision engine to approve the combination. Payloads may override their default rules when combined with a specific persistence module.

Grasshopper supports the following boolean operators:

	True if ...	*False* if ...	*Invalid* if ...
`and`	all arguments *True*	any arguments *False*	No *False* and any arguments *Invalid*
`or`	any arguments *True*	all arguments *False*	No *True* and any arguments *Invalid*
`xor`	one argument *True*	zero, more than one *True*	any arguments *Invalid*
`not`	argument *False*	argument *True*	argument *Invalid*
`assume_true`	argument *True*or*Invalid*	argument *False*	
`assume_false`	argument *True*	argument *False* or*Invalid*	

2.4 Install Log

Grasshopper may be configured to generate a log file that records its activity, including any data it collected and the return codes of each persistence module it executed. When combined with the build receipt, the log may be decoded to create a record of what the Installer saw on the target.

The log file is updated in real time as Grasshopper executes. The file will be written to the disk in a user-specified location. It is the responsibility of the operator to facilitate exfiltration of the log file.

2.5 Footprint

Grasshopper maintains a minimal footprint on the target machine. Persistence modules will, by necessity, expand that footprint. Persistence module users' guides should be consulted accordingly.

Installer Executable

When using the Grasshopper EXE, the executable is placed and run from the target filesystem with a user-determined name and location. During operation, the Grasshopper executable is present in the process list. Once the installer has finished its operation, the Grasshopper EXE self deletes.

The Grasshopper DLL is run from memory and will not explicitly touch the disk.

Log File

The Grasshopper log file is optionally stored on the target filesystem at a user-specified location. The log file is unencrypted, but is meaningless without the context provided by the installer build receipt.

3 Cricket

The Cricket Installer is an alternative installer in the Grasshopper tool suite. It is a mechanism for directly running Grasshopper persistence modules.

The Cricket Installer uses a given persistence module to install a given payload. The operator is responsible for selecting the appropriate module and payload for the target.

3.1 Usage

There are two ways to run a Cricket Installer, represented by two types of executables: EXEs and DLLs.

3.1.1 Executable

The Cricket EXE is a Windows application and may be invoked as such.

The executable does not take any arguments on the command line. All configuration and options are decided by the operator and set at build time.

3.1.2 Dynamic Load Library

The Cricket DLL is a Windows dynamic load library that implements the `MRC_FireAndForget` interface for the in-memory execution of DLLs.

The entry-point does not take any arguments through the exported function. All configurations are decided by the operator and set at build time.

3.2 Behavior

The Cricket installer executes the following steps after gaining execution.

1) Load persistence module

   Cricket loads the built-in persistence module in memory. The module is initialized using the user-defined configuration.

2) Run persistence module

   Cricket runs the persistence module against the built-in payload. The module is responsible for the deployment and execution of the payload.

3) Cleanup and Exit

   Cricket cleans up after its activity and exits. The Cricket EXE cleans itself up by initiating a self delete before exiting; the DLL is run from memory and does not need to self-delete.

3.3 Footprint

Cricket maintains a minimal footprint on the target machine. Persistence modules will, by necessity, expand that footprint. Persistence module users' guides should be consulted accordingly.

Installer Executable

When using the Cricket EXE, the executable is placed and run from the target filesystem with a user-determined name and location. During operation, the Cricket executable is present in the process list. Once the installer has finished its operation, the Cricket EXE self deletes.

The Cricket DLL is run from memory and will not explicitly touch the disk.

4 Builders

Builders are used to configure and generate Grasshopper and Cricket installer executables. Separate Builders are provided for each installer type.

The Builders will walk the operator through the process of selecting payloads and persistence methods and configuring the installer and persistence modules. Once the user is satisfied with the configuration, the Builder will pack the obfuscated payloads and modules into an installer.

Catalogs are used to store and document Grasshopper components for the Builder.

Templatesallow operators to save favorite payload and persistence module combinations for later use.

4.1 Usage

```
>>build_<grasshopper|cricket>.py[options] [catalog_path]*
```

`catalog_path`	Path(s) to Grasshopper catalog directories `./Modules` and `./Payloads` are implicitly included
`options:`	
`-o OUTDIR, --out_dir=OUTDIR`	Specify the directory to output installers and receipt. *Required.*
`-t TEMPLATE, --template=TEMPLATE`	Specify a template file to guide the build.
`-l, --loglevel=LOG_LEVEL`	Specify the logging level of the builder; higher = more logs.
`-h, --help`	Show the help message and exit.

The Builders use four directories to lookup files needed for the build process. These directories must be co-located with the `build_grasshopper.py` and `build_cricket.py` scripts. The Builder directories are:

- `Binaries` Location for unconfigured Grasshopper and Cricket binaries. *Required.*
- `Modules` Default location for persistence module catalogs and files.
- `Payloads` Default location for payload catalogs and files.
- `Rules` Default location for shared rule files.

The Builders require the Grasshopper Python module, named '`grasshopper`'. The module must be located in the Python search path, which includes the directory with the build scripts.

4.2 Command Line

The Grasshopper builder provides a command line interface to configure and build a Grasshopper installer. Using the command line, operators view and modify the list of candidate payloads and their lists of persistence modules.

Cricket installers have only one payload and one persistence module, such that the command line is not necessary in the Cricket builder.

4.2.1 Builder Commands

The builder commands are used to control the behavior of the builder. There are commands to add catalogs, set the build outputs, and build the installers. These commands are available at the grasshopper command line.

`add_catalog<catalog_path>`

Add a catalog directory or file to the running catalog list. Once added, the catalog entries will be available to the builder.

`catalog_path`	Path to a catalog directory or file to add to the builder.

`set_build_outputs<type_str>`

Set the types of Grasshopper executables to build. Defaults to building all binaries.

`type_str`	One of the following type strings: `'all'` - All available Grasshopper executables `'exe'` - Grasshopper EXEs, 32- and 64-bit `'dll'` - Grasshopper DLLs, 32- and 64-bit `'32'` - 32-bit Binaries, EXEs and DLLs `'64'` - 64-bit Binaries, EXEs and DLLs

`set_build_note<note>`

Set a note that will be included in the build receipt. Used for informational purposes only.

`note`	Note to be included in build receipt.

`build`

Build a Grasshopper Installer.

`exit`

Exit the Builder without building an Installer.

`set_global_rule [rule file (optional)]`

Set a global rule. If a file is not provided then the rule editor will be started.

4.2.2 Installer Commands

The Installer commands are used to modify the behavior of the Grasshopper installer.

`set_log_path<log_path>`	
Set a path on the target for the Grasshopper log file. If no path is provided, no log is generated.	
`log_path`	Note to be included in build receipt.

4.2.3 Payload Commands

The Payload commands are used to view and modify the list of payloads that will be built into the Grasshopper installer.

`add_payload`

Enter the payload wizard to generate a new payload and append it to the list.

`insert_payload<insert_index>`

Enter the payload wizard to generate a new payload and insert it into the list at the provided index.

`insert_index`	Index of the new payload in the list.

`edit_payload<payload_index>`

Edit the payload located at the provided index in the payload list.

`payload_index`	Index of the payload in the list.

`move_payload<payload_index><new_index>`

Move the payload located at the provided index to a new index in the list.

`payload_index`	Index of the payload in the list.
`new_index`	New index of the payload in the list.

`delete_payload<payload_index>`

Remove the payload located at the provided index in the payload list.

`payload_index`	Index of the payload in the list.

`print_payload [payload_index]`

Print detailed information about the payload located at the provided index in the list. If not index is provided, print general information about all payloads in the list.

`payload_index`	Index of the payload in the list.

4.2.4 Persist Module Commands

The Persist Module commands are used to view and modify the list of persistence modules that will be used for the current payload. These commands are available in the payload wizard.

`add_persist`

Enter the persistence wizard to generate a new persist module and append it to the persist list.

`insert_persist<insert_index>`

Enter the persistence wizard to generate a new persist module and insert it into the list at the provided index.

`insert_index`	Index of the new persist module in the list.

`edit_persist<persist_index>`

Edit the persist module located at the provided index in the current persist list.

`persist_index`	Index of the persist module in the list.

`move_persist<persist_index><new_index>`

Move the persist module located at the provided index to a new index in the list.

`persist_index`	Index of the persist module in the list.
`new_index`	New index of the persist module in the list.

`delete_persist<persist_index>`

Remove the persist module located at the provided index in the persist list.

`persist_index`	Index of the persist module in the list.

`print_persist [persist_index]`

Print detailed information about the persist module located at the provided index in the list. If not index is provided, print general information about all persist modules in the list.

`persist_index`	Index of the persist module in the list.

`cancel`

Discard all changes and exit the payload wizard.

`generate`

Save the current payload configuration and return to the command line.

4.3 Templates

The Grasshopper Builder allows operators to use Templates to identify the payload and persistence module combinations they would like to build. The template allows users to save and reuse favorite Grasshopper layouts.

The Builder will walk the user through the steps to configure the payloads and persistence modules identified by the template. Once the template has been satisfied, the operator is returned to the command line and allowed to update the configuration as needed.

4.3.1 Template File Format

Templates identify payloads and modules by UUID such that the Builder can identify the correct entries in the current build environment's catalog.

The Builder expects that the template identifies components on individual lines. Payloads are listed without any leading whitespace. Persistence modules are listed beneath their associated payloads with indentation. Empty lines, lines beginning with '#', and text after a UUID are ignored.

After discarding ignored lines, the payloads and persistence modules are specified in the format:

```
[<Payload_UUID>\n[<PersistModule_UUID>\n]+]+
```

4.4 Output

The results of any Grasshopper or Cricket build operation are stored in the directory '`<output_dir>/<timestamp>-Grasshopper.build`'.

4.4.1 Installers

The Builders will generate one or more installer executables during the build operation. The installers are stored in the build directory and named according to the scheme '`<Grasshopper|Cricket>-<32|64>.<exe|dll>`', based on the installer type, bitness, and format. Note that the DLL is both F&F version 2 and ICE-Fire compatible.

4.4.2 Build Receipt

The Builders generate Grasshopper build receipts for each build operation. The receipt is a directory containing all of the files and data necessary to fully document the build.

The receipt directory layout is as follows:

- <timestamp>-Grasshopper.receipt
  - build.xml

    '`build.xml`' stores the Grasshopper installer's configuration data. The xml format for the file is described in an Appendix.

  - op_script.txt

    '`op_script.txt`' provides a plain text description of the installer configuration. The op script uses pseudo-code to explain how the installer will select the payload and persistence module combination to install and documents the options configured in the payloads and modules.

  - binaries

    The `binaries`directory stores the persistence module and payload binaries used by the installer.

  - handlers

    The `handlers`directory stores the persistence module handlers uses to build the installer.

  - metadata

    The `metadata`directory stores the binary metadata embedded in the installers.

  - rules

    The `rules`directory stores the merged payload and persistence module rule files.

4.4.3 Template

The Builders will generate a Grasshopper template file during the build operation, recording the payloads and persistence modules used in the build.

The template is stored in the build directory and named `'template.txt'`.

4.5 Catalogs

Catalogs represent the set of payloads and persistence modules supported by the Grasshopper tool suite. Catalogs are indexed by xml-based listing files that document the catalog entries and identify the location of each entry's component files.

The operator may identify catalog directories at build time. The build scripts will recursively search the specified directories for listing files and add new catalog entries to the build environment.

The build receipt includes a catalog to store all catalog entries used in the build configuration.

4.5.1 Persistence Module Entry

Grasshopper catalogs are used to document and store supported persistence modules. By default, persistence modules are stored in the `./Modules` directory, relative to the builder scripts.

Persistence Module catalog entries include the following information:

Name	Name of the module, displayed by the builder
Method	Persistence method implemented by module, displayed by the builder
Description	Description of the module, displayed by the builder
UUID	Universally Unique Identifier of the catalog entry
Supported Types	Key-value specification for the types of supported payloads
Interface	Module-Payload interface implemented by the module
Obfuscation	Configuration for how the module should be obfuscated
Component Paths	Relative paths to the various module components

Persistence Module catalog entries include the following components, identified by the entry:

Rules	Rule file specifying the module requirements on the target
Handler	Python code invoked by the builder to configure the module
Binaries	Implementation of a persistence method used by an installer
Stub	Module-specific binary left on target to support persistence

4.5.2 Payload Entry

Grasshopper catalogs are used to document support payloads. By default, payload entries are stored in the `./Payloads` directory, relative to the builder scripts.

Payload catalog entries include the following information:

Name	Name of the payload, displayed by the builder
Method	Persistence method implemented by payload, displayed by the builder
Description	Description of the payload, displayed by the builder
UUID	Universally Unique Identifier of the catalog entry
Type	Key-value specification of the payload type
Interface	Module-Payload interface name
Parameters	Usage and default values for payload parameters, used by the builder
Obfuscation	Configuration for how the payload should be obfuscated

Payload catalog entries include the following components, indentified by the entry:

Rules	Rule file(s) specifying the module requirements on the target

5 Post Processor

The Post Processor is a Python script used to decode binary log files generated by a Grasshopper installer.

The Post Processor requires a Grasshopper receipt and the encoded log from its installation. The Grasshopper log file contains the execution time, a set of all rule resultsevaluated during installation and return codes for every installation attempt. When combined with the installer's receipt, the Post Processor is able to decipher the actions taken by the installer and the results of those actions.

5.1 Usage

```
>> decode_log.py [options]

options:
```

`-i INFILE, --input_file=INFILE`	Specify the log file to decode. *Required.*
`-r RECEIPTDIR --receipt_dir=RECEIPTDIR`	Specify the receipt directory for the installer. *Required*.
`-o OUTDIR, --out_dir=OUTDIR`	Specify the directory to output log data. *Required.*
`-l, --loglevel=LOG_LEVEL`	Specify the logging level of the builder; higher level = more logs.
`-h, --help`	Show the help message and exit.

The Post Processor requires the Grasshopper Python module, named '`grasshopper`'. The module must be located in the Python search path, which includes the directory with the decode script.

5.2 Results

The Post Processor will generate two files in the given output directory, a data file and a log file.

5.2.1 Data File

The data file is an xml representation of the raw data stored in the input file combined with context provided by the receipt file. The file includes results for both fact evaluations and installation attempts.

The data file is stored at '`<output_dir>/<decode_timestamp>-<input_file>.xml`'.

The format of the data file is further discussed in the Appendix.

5.2.2 Log File

The log file is a plain text description of the data collected by the installer and the actions taken as a result of that data.

The log file is stored at '`<output_dir>/<decode_timestamp>-<input_file>.log`'.

Appendix A:

Appendix B: Change Log

Date	Change Description	Authority
05/2012	Document Initialization	2355679
09/2012	Update for Grasshopper v1.0 Phase 2 Delivery	2355679
11/2012	Update for Grasshopper v1.0.1 Delivery	2355679
12/2013	Update for Grasshopper v1.1 Delivery	2326131