NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer’s or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.
Dshell User Guide

Daniel E Krych
DEVCOM Army Research Laboratory

Joshua Edwards
ICF

Approved for public release: distribution unlimited.
This report is a general user guide for the decoder-shell (Dshell) framework. It details installation and both basic and advanced analysis usage with examples. Dshell is an open-source, Python-based, network forensic analysis framework developed by the US Army Combat Capabilities Development Command Army Research Laboratory. It is a modular and flexible framework, which includes over 40 plugins for the analysis and decoding of network traffic using a variety of network protocols. Dshell plugins are designed to aid in the understanding of network traffic and present results to the user in a concise, useful manner via command-line interface. Dshell is a tool for network forensic analysis that can be used out of the box for simple and advanced analyses, or customized to fit an end-user’s needs. The Dshell GitHub repository contains the current Python 3 version as well as an archived Python 2 version available as a tarball. This user guide only applies to the current version.
Contents

List of Figures iv
List of Tables iv

1. Introduction 1

2. Key Features 2

3. Installation 2
   3.1 Steps 2
   3.2 Requirements 3
   3.3 Optional 3

4. Basic Usage 4

5. Usage Examples 10
   5.1 Showing DNS Lookups in Sample Traffic 10
   5.2 Following and Reassembling a Stream in Sample Traffic 11
   5.3 Chaining Plugins to View Flow Data for a Specific Country Code In Sample Traffic 13
   5.4 Collecting DNS Traffic from Several Files and Storing in a New pcap File 13
   5.5 Collecting TFTP Data and Converting Alerts to Different Formats Using Sample Traffic 13
   5.6 Using the pcapout Output Format and Comparing the Original and New pcap Using Sample Traffic 15
   5.7 Running a Plugin Live on an Interface 16
   5.8 Running a Plugin Within a Separate Python Script Using Sample Traffic 16

6. References 17

List of Symbols, Abbreviations, and Acronyms 18

Distribution List 20
List of Figures

Fig. 1  Overview of Dshell—a modular, Python-based network forensic analysis framework that ingests network traffic data in the form of raw packet captures (pcap), uses plugins to parse/decode the traffic, and outputs key artifacts to aid with understanding the data....................... 1

List of Tables

Table 1  Dshell’s basic usage commands and descriptions .............................. 5
Table 2  Dshell’s “decode -h” output, displaying helpful information including generic command-line flags available to most plugins.............................. 6
Table 3  Dshell’s “decode -l” output (lowercase “L”): list of all available plugins and their basic information................................................................. 7
Table 4  Dshell’s “decode --lo” output (lowercase “L”): list of available output modules and their basic information.......................................................... 9
Table 5  Dshell’s “decode -p <plugin>” output for netflow and sweetorange plugins.............................................................................................................. 9
Table 6  Dshell’s “decode -p <plugin> -h” output for netflow and sweetorange plugins............................................................................................................. 10
1. Introduction

This report is a general user guide for the decoder-shell (Dshell) framework.¹ It details installation and both basic and advanced analysis usage with examples.

Dshell¹ is an open-source, Python-based, network forensic analysis framework developed by the US Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory (ARL). It is a modular and flexible framework (Fig. 1), which includes over 40 plugins for the analysis and decoding of network traffic using a variety of network protocols. Dshell plugins are designed to aid in the understanding of network traffic and present results to the user in a concise, useful manner via command-line interface.

![Fig. 1](image)

**Fig. 1** Overview of Dshell—a modular, Python-based network forensic analysis framework that ingests network traffic data in the form of raw packet captures (pcap), uses plugins to parse/decode the traffic, and outputs key artifacts to aid with understanding the data.

Dshell¹ was first publicly released as an open-source network forensic analysis framework on GitHub in 2014, written in Python 2. In 2020 Dshell was rewritten in Python 3 from the ground up and again made available as open-source software on GitHub, following the Python 2 language deprecation on 1 JAN 2020.² Plugins written for the deprecated Python 2 version of Dshell are not compatible with this version and vice versa. The Dshell¹ GitHub repository contains the current Python
version as well as an archived Python 2 version available as a tarball. This user guide only applies to the current version.

Dshell is a tool for network forensic analysis that can be used out of the box for simple and advanced analyses, or customized to fit an end-user’s needs. Custom Dshell plugins can be developed to parse, decode, and analyze network traffic protocols and data. For a detailed guide on developing custom Dshell plugins, modifying existing plugins, and gaining an understanding of the innerworkings of the Dshell framework, please see the *Dshell Developer Guide*.3

2. **Key Features**

Dshell is a sophisticated network forensic analysis framework. Dshell can read packets from two types of sources: 1) pcap and pcapng files and 2) network interfaces. When users run Dshell they can build a chain of sequential plugins to control and filter packets. In practice, users generally only use one plugin.

The key features of Dshell include the following:

- Deep packet analysis using specialized plugins,
- Robust stream reassembly,
- IPv4 and IPv6 support,
- Multiple user-selectable output formats and the ability to create custom output handlers,
- Chainable plugins,
- Parallel processing option to divide the handling of data source into separate Python processes, and
- Development of external plugin packs to share and install new externally developed plugins without overlapping the core Dshell plugin directories.

3. **Installation**

This section details the installation of the software from the Dshell GitHub repository, Dshell’s requirements, and optional resources that support some of Dshell’s features or plugins.

3.1 **Steps**

The following steps will install Dshell and its required libraries:
1) Use Git clone or download the Dshell repo as a .zip file from GitHub.

   `git clone https://github.com/USArmyResearchLab/Dshell.git`

2) Install Dshell with pip using one of the following options:

   ```python
   python3 -m pip install Dshell/
   python3 -m pip install <Dshell-tarball>
   ```

3) Configure geoip2 by placing the MaxMind GeoLite2 data set files (GeoLite2-ASN.mmdb, GeoLite2-City.mmdb, GeoLite2-Country.mmdb) in `[...,]/site-packages/dshell/data/GeoIP/`.

### 3.2 Requirements

Dshell was developed in Python 3 for use on Linux OS but can also be used on MAC OS. With additional resources it can also be used on Windows OS (i.e., through Windows Subsystem for Linux). Several libraries are used by Dshell and installed automatically during the pip installation process. Dshell’s main requirements are listed here:

- Linux (developed on Ubuntu 20.04)
- Python 3 (developed with Python 3.8.10)
- `pypacker`
- `pcapy-ng`
- `pyOpenSSL`
- `geoip2`
  - MaxMind GeoIP2 data sets (Used to map IP addresses to country codes)
  - Configure geoip2 by placing the MaxMind GeoLite2 data set files (GeoLite2-ASN.mmdb, GeoLite2-City.mmdb, GeoLite2-Country.mmdb) in `[...,]/site-packages/dshell/data/GeoIP/`

### 3.3 Optional

Several optional resources are available online that support some of Dshell’s features or plugins. For instance, to resolve GeoIP lookups and have IP addresses mapped to country codes rather than defaulting to ‘??’ in output, follow the optional
step to download and move the MaxMind data files to the proper location. Dshell’s optional resources include the following:

- **oui.txt**
  - Used by some plugins that handle MAC addresses to look up organizational unique identifier (OUI) labels
  - Place in `<dshell>/data/`

- **elasticsearch**
  - Used in the elasticout output module
  - Only necessary if planning to use elasticsearch to store output

- **pyJA3**
  - Used in the tls plugin

### 4. Basic Usage

The following commands provide the needed information to get started using Dshell. Dshell’s functionality is provided through its `decode` command. Table 1 details the basic usage commands of Dshell and Tables 2–6 go into further detail with these basic usage commands and their options via command-line flags. These commands are used for general help, listing all available plugins and output formats, and finding more information on specific plugins (e.g., descriptions, plugin-specific options, and default Berkeley Packet Filters [BPFs]) prior to use with data sources.

The remaining commands in Table 1 apply the prior seen commands, adding `<pcap>` or `<interface>` as data sources to process data with Dshell plugins. Examples of using these commands and more are provided in Section 5, Usage Examples. Please refer to the List of Symbols, Abbreviations, and Acronyms at the end of this report for definitions of terminology used within Tables 2–6.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>decode -h</td>
<td>Display helpful information including generic command-line flags available to most plugins.</td>
</tr>
<tr>
<td>decode -l</td>
<td>List all available plugins, alongside basic information about them (flag is a lowercase “L”).</td>
</tr>
<tr>
<td>decode --lo</td>
<td>List available output modules that can be used by plugins (not every output module will be useful to every plugin [e.g., using netflow output for a plugin that looks at individual packets]) but is available for use. Output module “pcapout” is only intended to work with PacketPlugins (i.e., plugins operating at the packet level).</td>
</tr>
<tr>
<td>decode -p &lt;plugin&gt;</td>
<td>Display information about a plugin, including available command-line flags both commonly used and unique to that plugin.</td>
</tr>
<tr>
<td>decode -p &lt;plugin&gt; -h</td>
<td>Display information about a plugin—including all available command-line flags, flags unique to that plugin, and the plugin’s long description and BPF.</td>
</tr>
<tr>
<td>decode -p &lt;plugin&gt; &lt;pcap&gt;</td>
<td>Run the selected plugin on a pcap file.</td>
</tr>
<tr>
<td>decode -p &lt;plugin1&gt;+&lt;plugin2&gt; &lt;pcap&gt;</td>
<td>Chain two (or more) plugins together and run them on a pcap file.</td>
</tr>
<tr>
<td>decode -p &lt;plugin&gt; -i &lt;interface&gt;</td>
<td>Run the selected plugin live on an interface (may require super-user privileges).</td>
</tr>
</tbody>
</table>
Table 2  Dshell’s “decode -h” output, displaying helpful information including generic command-line flags available to most plugins

<table>
<thead>
<tr>
<th>Argument/flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dshell&gt; decode -h</strong></td>
<td>Usage: <code>decode.py [options] [plugin options] file1 file2 ... fileN</code></td>
</tr>
<tr>
<td><strong>Positional arguments:</strong></td>
<td></td>
</tr>
<tr>
<td>files</td>
<td>pcap files or globs to process</td>
</tr>
<tr>
<td><strong>Optional arguments:</strong></td>
<td></td>
</tr>
<tr>
<td>-c COUNT, --count COUNT</td>
<td>Number of packets to process</td>
</tr>
<tr>
<td>--debug</td>
<td>Show debug messages</td>
</tr>
<tr>
<td>-v, --verbose</td>
<td>Show informational messages</td>
</tr>
<tr>
<td>-acc, --allcc</td>
<td>Show all three GeoIP2 country code types (represented country/registered country/country)</td>
</tr>
<tr>
<td>-d PLUGIN,</td>
<td>Use a specific plugin module; can be chained with '+'</td>
</tr>
<tr>
<td>-p PLUGIN,</td>
<td>Use a specific plugin module; can be chained with '+'</td>
</tr>
<tr>
<td>--plugin PLUGIN</td>
<td>Use a specific plugin module; can be chained with '+'</td>
</tr>
<tr>
<td>--defragment</td>
<td>Reconnect fragmented IP packets</td>
</tr>
<tr>
<td>-h, ?, --help</td>
<td>Print common command-line flags and exit</td>
</tr>
<tr>
<td>-i INTERFACE,</td>
<td>Listen live on <code>INTERFACE</code> instead of reading pcap</td>
</tr>
<tr>
<td>--interface INTERFACE</td>
<td>List all available plugins</td>
</tr>
<tr>
<td>-r, --recursive</td>
<td>Recursively process all PCAP files under input directory</td>
</tr>
<tr>
<td>--unzipdir DIRECTORY</td>
<td>Directory to use when decompressing input files (.gz, .bz2, and .zip only)</td>
</tr>
<tr>
<td>--version</td>
<td>Show program version numbers and exit</td>
</tr>
<tr>
<td><strong>Multiprocessing arguments:</strong></td>
<td></td>
</tr>
<tr>
<td>-P, --parallel</td>
<td>Handle each file in separate parallel processes</td>
</tr>
<tr>
<td>-n NUMPROCS,</td>
<td>Define max number of parallel processes (default: 4)</td>
</tr>
<tr>
<td>--nprocs NUMPROCS</td>
<td>Define max number of parallel processes (default: 4)</td>
</tr>
<tr>
<td><strong>Filter arguments:</strong></td>
<td></td>
</tr>
<tr>
<td>--bpf BPF</td>
<td>Overwrite all BPFs and use provided input. (Caution: use carefully)</td>
</tr>
<tr>
<td>--ebpf BPF</td>
<td>Extend existing BPFs with provided input for additional filtering. Transform input into &quot;(&lt;original bpf&gt;) and (&lt;ebpf&gt;)&quot;</td>
</tr>
<tr>
<td>--no-vlan</td>
<td>Ignore packets with VLAN headers</td>
</tr>
<tr>
<td><strong>Output arguments:</strong></td>
<td></td>
</tr>
<tr>
<td>--lo, --list-output</td>
<td>List available output modules</td>
</tr>
<tr>
<td>--no-buffer</td>
<td>Do not buffer plugin output</td>
</tr>
<tr>
<td>-x, --extra</td>
<td>Appends extra data to all plugin output</td>
</tr>
<tr>
<td>-O MODULE,</td>
<td>Use specified output module for plugins instead of defaults (e.g., --omodule=jsonout for JSON output)</td>
</tr>
<tr>
<td>--omodule MODULE</td>
<td>Use specified output module for plugins instead of defaults (e.g., --omodule=jsonout for JSON output)</td>
</tr>
<tr>
<td>--oarg ARG=VALUE</td>
<td>Supply a specific keyword argument to plugins’ output modules. Can be used multiple times for multiple arguments. Not using an equal sign will treat it as a flag and set the value to True. Example: --oarg &quot;delimiter=:&quot; --oarg &quot;timeformat=%H %M %S&quot;</td>
</tr>
<tr>
<td>-q, --quiet</td>
<td>Disable logging</td>
</tr>
<tr>
<td>-W OUTFILE</td>
<td>Write to OUTFILE instead of stdout</td>
</tr>
<tr>
<td><strong>Plugin options:</strong></td>
<td></td>
</tr>
<tr>
<td>Additional options specific to a plugin are obtained by running “decode -p plugin-name -h”</td>
<td></td>
</tr>
</tbody>
</table>
Table 3  Dshell’s “decode -l” output (lowercase “L”): list of all available plugins and their basic information

<table>
<thead>
<tr>
<th>Module</th>
<th>Name</th>
<th>Title</th>
<th>Type</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dshell.plugins.protocol.bitcoin</td>
<td>bitcoin</td>
<td>bitcoin</td>
<td>ConnectionPlugin</td>
<td>dek</td>
<td>Extract Bitcoin traffic, including Stratum mining protocol (pooled) traffic</td>
</tr>
<tr>
<td>dshell.plugins.filter.country</td>
<td>country</td>
<td>Country Filter</td>
<td>ConnectionPlugin</td>
<td>tp</td>
<td>Filter connections by IP address country code</td>
</tr>
<tr>
<td>dshell.plugins.flows.dataflows</td>
<td>dataflows</td>
<td>dataflows</td>
<td>ConnectionPlugin</td>
<td>amm</td>
<td>Display netflows that have at least one byte transferred</td>
</tr>
<tr>
<td>dshell.plugins.dhcp.dhcp</td>
<td>dhcp</td>
<td>dhcp</td>
<td>PacketPlugin</td>
<td>dek</td>
<td>Extract client information from DHCP messages</td>
</tr>
<tr>
<td>dshell.plugins.dns.dns</td>
<td>dns</td>
<td>DNS</td>
<td>DNSPlugin</td>
<td>bg/twp</td>
<td>Extract and summarize DNS queries/responses</td>
</tr>
<tr>
<td>dshell.plugins.dns.dnscc</td>
<td>dnscc</td>
<td>DNS Country Code</td>
<td>DNSPlugin</td>
<td>bg</td>
<td>Identify country code of DNS A/AAAA record responses</td>
</tr>
<tr>
<td>dshell.plugins.protocol.ether</td>
<td>ether</td>
<td>Ethernet</td>
<td>PacketPlugin</td>
<td>dev195</td>
<td>Show MAC address information and optionally filter by it</td>
</tr>
<tr>
<td></td>
<td>followstream</td>
<td>Followstream</td>
<td>ConnectionPlugin</td>
<td>amm/dev195</td>
<td>Generates color-coded Screen/HTML output similar to Wireshark Follow Stream. Empty connections will be skipped.</td>
</tr>
<tr>
<td>dshell.plugins.http.httpdump</td>
<td>httpdump</td>
<td>httpdump</td>
<td>HTTPPlugin</td>
<td>amm</td>
<td>Dump useful information about HTTP sessions</td>
</tr>
<tr>
<td>dshell.plugins.portscan.indegree</td>
<td>indegree</td>
<td>parse indegree</td>
<td>ConnectionPlugin</td>
<td>dev195</td>
<td>Parse traffic to detect scanners based on connection to IPs that are rarely touched by others</td>
</tr>
<tr>
<td></td>
<td>inuendo-dns</td>
<td>inuendo-dns</td>
<td>DNSPlugin</td>
<td>primalsec</td>
<td>Proof-of-concept detector for INNUENDO DNS channel IPv4/IPv6 plugin</td>
</tr>
<tr>
<td>dshell.plugins.protocol.ip</td>
<td>ip</td>
<td>ip</td>
<td>PacketPlugin</td>
<td>twp</td>
<td>Detect attempts to enumerate MS15-034 vulnerable IIS servers</td>
</tr>
<tr>
<td>dshell.plugins.http.joomla</td>
<td>joomla</td>
<td>Joomla CVE-2015-8562</td>
<td>HTTPPlugin</td>
<td>bg</td>
<td>Display netflows that have at least 1 MB transferred</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Display netflows that have a duration of at least 5 min</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Detect attempts to enumerate MS15-034 vulnerable IIS servers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Extract client information from NBNS traffic</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Collect and display statistics about connections</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Generate pcap output for plugins that cannot use -o pcapout</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Generate visualizations based on connections</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Find uncommon (i.e., not TCP, UDP, or ICMP) protocols in IP traffic</td>
</tr>
</tbody>
</table>
### Table 3: Dshell’s “decode -l” output (lowercase “L”): list of all available plugins and their basic information (continued)

<table>
<thead>
<tr>
<th>Module</th>
<th>Name</th>
<th>Title</th>
<th>Type</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dshell.plugins.flows.reverseflows</td>
<td>reverseflows</td>
<td>reverse-flows</td>
<td>ConnectionPlugin</td>
<td>me</td>
<td>Generate an alert if the client transmits more data than the server</td>
</tr>
<tr>
<td>dshell.plugins.http.riphttp</td>
<td>riphttp</td>
<td>rip-http</td>
<td>HTTPPlugin</td>
<td>bg,twp</td>
<td>Rip files from HTTP traffic Real-time transport protocol (RTP) capture plugin</td>
</tr>
<tr>
<td>dshell.plugins.voip.rtp</td>
<td>rtp</td>
<td>RTP</td>
<td>PacketPlugin</td>
<td>mm/dev195</td>
<td>Search for patterns in connections (UNFINISHED) Session Initiation Protocol (SIP) capture plugin</td>
</tr>
<tr>
<td>dshell.plugins.misc.search</td>
<td>search</td>
<td>search</td>
<td>ConnectionPlugin</td>
<td>dev195</td>
<td></td>
</tr>
<tr>
<td>dshell.plugins.voip.sip</td>
<td>sip</td>
<td>SIP</td>
<td>PacketPlugin</td>
<td>mm/dev195</td>
<td>Look for SSL alert messages Look for certificate SHA1 matches in the abuse.ch blacklist</td>
</tr>
<tr>
<td>dshell.plugins.misc.synrst</td>
<td>synrst</td>
<td>SYN/RST</td>
<td>PacketPlugin</td>
<td>bg</td>
<td>Detect failed attempts to connect (SYN followed by RST/ACK)</td>
</tr>
<tr>
<td>dshell.plugins.tftp.tftp</td>
<td>tftp</td>
<td>tftp</td>
<td>PacketPlugin</td>
<td>dev195</td>
<td>Extract TFTP streams, and optionally extract the files</td>
</tr>
<tr>
<td>dshell.plugins.ssl.tls</td>
<td>tls</td>
<td>tls</td>
<td>ConnectionPlugin</td>
<td>amm</td>
<td>Extract interesting metadata from TLS connection setup</td>
</tr>
<tr>
<td>dshell.plugins.flows.toptalkers</td>
<td>toptalkers</td>
<td>Top Talkers</td>
<td>ConnectionPlugin</td>
<td>dev195</td>
<td>Find top-talkers based on byte count Only follow connections that match user-provided IP addresses and ports Use Threshold Random Walk to detect network scanners</td>
</tr>
<tr>
<td>dshell.plugins.filter.track</td>
<td>track</td>
<td>track</td>
<td>ConnectionPlugin</td>
<td>twp,dev195</td>
<td>Display basic information for web requests/responses in a connection</td>
</tr>
<tr>
<td>dshell.plugins.portscan.trw</td>
<td>trw</td>
<td>trw</td>
<td>PacketPlugin</td>
<td>dev195</td>
<td>Show 802.11 packet information</td>
</tr>
<tr>
<td>dshell.plugins.http.web</td>
<td>web</td>
<td>web</td>
<td>HTTPPlugin</td>
<td>bg,twp</td>
<td>Show SSIDs of 802.11 wireless beacons XOR every packet with a given key</td>
</tr>
<tr>
<td>dshell.plugins.wifi.wifi80211</td>
<td>wifi80211</td>
<td>802.11</td>
<td>PacketPlugin</td>
<td>dev195</td>
<td></td>
</tr>
<tr>
<td>dshell.plugins.wifi.wifibeacon</td>
<td>wifibeacon</td>
<td>Wi-fi Beacons</td>
<td>PacketPlugin</td>
<td>dev195</td>
<td></td>
</tr>
<tr>
<td>dshell.plugins.misc.xor</td>
<td>xor</td>
<td>xor</td>
<td>ConnectionPlugin</td>
<td>dev195</td>
<td></td>
</tr>
</tbody>
</table>
Table 4  Dshell’s “decode --lo” output (lowercase “L”): list of available output modules and their basic information

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alertout</td>
<td>Default format for printing a single-line alert</td>
</tr>
<tr>
<td>colorout</td>
<td>Reconstructed output with ANSI color codes</td>
</tr>
<tr>
<td>csvout</td>
<td>CSV format output</td>
</tr>
<tr>
<td>elasticout</td>
<td>Automatically insert data into an elasticsearch instance</td>
</tr>
<tr>
<td>htmlout</td>
<td>HTML format output</td>
</tr>
<tr>
<td>jsonout</td>
<td>JSON format output</td>
</tr>
<tr>
<td>netflowout</td>
<td>Flow (connection overview) format output</td>
</tr>
<tr>
<td>pcapout&lt;sup&gt;a&lt;/sup&gt;</td>
<td>Writes data to a pcap file (does not work with connection-based plugins). Use the pcapwriter plugin for more control over pcap output.</td>
</tr>
</tbody>
</table>

<sup>a</sup> Intended to work only with PacketPlugins (i.e., plugins operating at the packet level).

Table 5  Dshell’s “decode -p <plugin>” output for netflow and sweetorange plugins

Dshell> decode -p netflow
(Trimmed output from “-h” flag, commonly used flags displayed here)
(Plugin specific flags displayed here, if applicable)
Include a pcap file to get started. Use --help for more information.

Dshell> decode -p sweetorange
(Trimmed output from “-h” flag, commonly used flags displayed here)
sweetorange plugin options:
--sweetorange_variable SWEETORANGE_VARIABLE
    Variable names to search for. Default
    (“ajax_data_source”, “main_request_data_content”)
--sweetorange_color
    Display encoded/decoded lines in different TTY colors

Include a pcap file to get started. Use --help for more information.
Table 6  Dshell’s “decode -p <plugin> -h” output for netflow and sweetorange plugins

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| Dshell> decode -p netflow -h | Collects and displays statistics about connections
| | (All output from “-h” flag displayed here)
| | Netflow
| | Default BPF: “ip or ip6”

sweetorange plugin options:

--sweetorange_variable SWEETORANGE_VARIABLE
Variable names to search for. Default
(“ajax_data_source”, “main_request_data_content”)

--sweetorange_color Display encoded/decoded lines in different TTY colors.

sweetorange
Used to decode certain variants of the Sweet Orange exploit kit to redirect traffic. Looks for telltale Javascript variable names (e.g., “ajax_data_source” and “main_request_data_content”) and automatically decodes the exploit landing page contained.

Default BPF: “tcp and (port 80 or port 8080 or port 8000)”

5. Usage Examples

The following examples show how the remaining commands from Table 1 are applied. These examples demonstrate Dshell’s functionality using a variety of plugins and command-line options.

5.1 Showing DNS Lookups in Sample Traffic

Dshell> decode -p dns ~/pcap/dns.cap | sort

ID: 4146, TXT? google.com., TXT: b’\x0fv=spf1 ptr ?all’ **

ID: 63343, MX? google.com., MX: b’\x00\x05smtp4\xc0\x0c’, MX: b’\x00\x05smtp5\xc0\x0c’, MX: b’\x00\x05smtp6\xc0\x0c’, MX: b’\x00\x05smtp7\xc0\x0c’, MX: b’\x00\x05smtp8\xc0\x0c’ **


ID: 18849, LOC? google.com. **


5.2 Following and Reassembling a Stream in **Sample Traffic**

Dshell> decode -p followstream ~/pcap/v6-http.cap

Connection 1 (TCP)
Start: 2007-08-05 15:16:44.189851
End: 2007-08-05 15:16:44.219460
(300 bytes)
(2379 bytes)

GET / HTTP/1.0
Host: cl-1985.ham-01.de.sixxs.net
Accept: text/html, text/plain, text/css, text/sxml, */*;q=0.01
Accept-Encoding: gzip, bzip2
Accept-Language: en
User-Agent: Lynx/2.8.6rel.2 libwww-FM/2.14 SSL-MM/1.4.1 OpenSSL/0.9.8b

HTTP/1.1 200 OK
Date: Sun, 05 Aug 2007 19:16:44 GMT
Index of /

A directory listing is not available through this interface.

A partial directory listing is not available through this interface.

A directory listing is not available through this interface.

A directory listing is not available through this interface.

A directory listing is not available through this interface.

A partial directory listing is not available through this interface.

A directory listing is not available through this interface.

A directory listing is not available through this interface.

A directory listing is not available through this interface.

A directory listing is not available through this interface.

A directory listing is not available through this interface.

A directory listing is not available through this interface.

A directory listing is not available through this interface.

A directory listing is not available through this interface.

A directory listing is not available through this interface.

A directory listing is not available through this interface.

A directory listing is not available through this interface.
5.3 Chaining Plugins to View Flow Data for a Specific Country Code In Sample Traffic

Note: TCP handshakes are not included in the packet count.

Dshell> decode -p country+netflow --country_code=JP ~/pcap/SkypeIRC.cap

UDP  60583  33438  1 0  64  0 0.0000s
UDP  60583  33435  1 0  64  0 0.0000s
UDP  60583  33437  1 0  64  0 0.0000s
UDP  60583  33436  1 0  64  0 0.0000s

5.4 Collecting DNS Traffic from Several Files and Storing in a New pcap File

Dshell> decode -p dns+pcapwriter --pcapwriter_outfile=test.pcap ~/pcap/*.cap > /dev/null

Dshell> tcpdump -nnr test.pcap | head
reading from file test.pcap, link-type EN10MB (Ethernet)
15:36:08.670569 IP 192.168.1.2.2131 > 192.168.1.1.53: 40209+ A?
ui.skype.com. (30)
15:36:08.670687 IP 192.168.1.2.2131 > 192.168.1.1.53: 40210+ AAAA?
ui.skype.com. (30)
15:36:08.674022 IP 192.168.1.1.53 > 192.168.1.2.2131: 40209- 1/0/0 A
212.72.49.131 (46)
15:36:10.171350 IP 192.168.1.2.2131 > 192.168.1.1.53: 40210+ AAAA?
ui.skype.com. (30)
15:36:10.961350 IP 192.168.1.1.53 > 192.168.1.2.2131: 40210* 0/1/0 (85)
ui.skype.com. (30)
15:36:11.294333 IP 192.168.1.2.2131 > 192.168.1.1.53: 40211+ AAAA?
ui.skype.com. (30)
ui.skype.com. (30)
ui.skype.com. (30)

5.5 Collecting TFTP Data and Converting Alerts to Different Formats Using Sample Traffic

Note: The tftp plugin uses the alertout output module by default. The following example can be run without the “-O alertout” and will produce the same result.

Dshell> decode -p tftp -O alertout ~/pcap/tftp_*.pcap

** read rfc1350.txt (24599 bytes) **
write rfc1350.txt (24599 bytes) **

Dshell> decode -p tftp -O colorout ~/pcap/tftp_*_.pcap

Packet 1 ()
Start: 2013-05-01 08:24:11
192.168.0.253: 50618 -> 192.168.0.10: 3445 ( bytes)
read rfc1350.txt (24599 bytes)

Packet 2 ()
Start: 2013-04-27 05:07:59
192.168.0.1: 57509 -> 192.168.0.13: 2087 ( bytes)
write rfc1350.txt (24599 bytes)

Dshell> decode -p tftp -O csvout ~/pcap/tftp_*_.pcap


Dshell> decode -p tftp -O htmlout ~/pcap/tftp_*_.pcap

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Dshell Output</title>
<style>
body {
  font-family: monospace;
  font-size: 10pt;
  white-space: pre;
}
h1 {
  font-family: helvetica;
  font-size: 13pt;
  font-weight: bolder;
  white-space: pre;
}
h2 {
  font-family: helvetica;
  font-size: 12pt;
  font-weight: bolder;
  margin: 0 0;
  white-space: pre;
}
</style>
</head>
<body>
<h1>Packet 1 ()</h1>
<h2>Start: 2013-05-01 08:24:11</h2>

<h1>Packet 2 ()</h1>
<h2>Start: 2013-04-27 05:07:59</h2>

</body>
</html>
Packet 2 ()

Start: 2013-04-27 05:07:59
192.168.0.1:57509 -> 192.168.0.13:2087 ( bytes)

write rfc1350.txt (24599 bytes)

Dshell> decode -p tftp -O jsonout ~/pcap/tftp_*pcap

Dshell> decode -p tftp -O netflowout ~/pcap/tftp_*pcap

5.6 Using the pcapout Output Format and Comparing the Original and New pcap Using Sample Traffic

Dshell> tcpdump -r ~/pcap/dhcp.pcap -W test_pcapout.pcap

Dshell> tcpdump -r test_pcapout.pcap
5.7 Running a Plugin Live on an Interface

Dshell> decode -p netflow -i eth0 --nobuffer
2023-02-27 3:14:15 10.0.0.3 -> 10.0.0.4 (-- -> --) TCP 12345 9999
8 8 256 256 23.4567s
2023-02-27 3:14:16 10.0.0.3 -> 10.0.0.4 (-- -> --) TCP 12345 9999
8 8 256 256 23.4567s
^C...

$ sudo timeout 10s dshell-decode -p netflow -i eth0 --nobuffer
2023-02-27 4:8:15 10.0.0.16 -> 10.0.0.23 (-- -> --) TCP 42000 5678
8 8 256 256 0.5678s
2023-02-27 4:8:15 10.0.0.16 -> 10.0.0.23 (-- -> --) TCP 42000 5678
8 8 256 256 0.5678s

5.8 Running a Plugin Within a Separate Python Script Using Sample Traffic

```python
# Import required Dshell libraries
import dshell.decode as decode
import dshell.plugins.tftp.tftp as tftp

# Instantiate plugin
plugin = tftp.DshellPlugin()
# Define plugin-specific arguments, if needed
dargs = {plugin: {"rip": True, "outdir": "/tmp/"}}
# Add plugin(s) to plugin chain
decode.plugin_chain = [plugin]
# Run decode main function with all other arguments
decode.main(
    debug=True,
    files=["/home/user/pcap/tftp_rrq.pcap",
          "/home/user/pcap/tftp_wrq.pcap"],
    plugin_args=dargs
)
```
6. References


### List of Symbols, Abbreviations, and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSI</td>
<td>American National Standards Institute</td>
</tr>
<tr>
<td>ARL</td>
<td>Army Research Laboratory</td>
</tr>
<tr>
<td>BPF</td>
<td>Berkeley Packet Filter</td>
</tr>
<tr>
<td>CSV</td>
<td>comma-separated values</td>
</tr>
<tr>
<td>DEVCOM</td>
<td>US Army Combat Capabilities Development Command</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration Protocol</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>Dshell</td>
<td>decoder-shell</td>
</tr>
<tr>
<td>FTP</td>
<td>file transfer protocol</td>
</tr>
<tr>
<td>HTML</td>
<td>hypertext markup language</td>
</tr>
<tr>
<td>HTTP</td>
<td>hypertext transfer protocol</td>
</tr>
<tr>
<td>ICMP</td>
<td>Internet control message protocol</td>
</tr>
<tr>
<td>IIS</td>
<td>Internet Information Services</td>
</tr>
<tr>
<td>IP</td>
<td>Internet protocol</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>NBNS</td>
<td>NetBIOS Name Service</td>
</tr>
<tr>
<td>OS</td>
<td>operating system</td>
</tr>
<tr>
<td>OUI</td>
<td>organizational unique identifier</td>
</tr>
<tr>
<td>pcap</td>
<td>packet capture</td>
</tr>
<tr>
<td>RTP</td>
<td>real-time transport protocol</td>
</tr>
<tr>
<td>SIP</td>
<td>session initiation protocol or source IP</td>
</tr>
<tr>
<td>SSH</td>
<td>secure shell</td>
</tr>
<tr>
<td>SSID</td>
<td>service set identifier</td>
</tr>
<tr>
<td>SSL</td>
<td>secure sockets layer</td>
</tr>
<tr>
<td>TCP</td>
<td>transmission control protocol</td>
</tr>
<tr>
<td>TFTP</td>
<td>trivial file transfer protocol</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>TLS</td>
<td>transport layer security</td>
</tr>
<tr>
<td>UDP</td>
<td>user datagram protocol</td>
</tr>
<tr>
<td>VLAN</td>
<td>virtual local area network</td>
</tr>
</tbody>
</table>