ARL-SR-0470 ● APR 2023

DEVCOM ARMY RESEARCH LABORATORY

Dshell User Guide

by Daniel E Krych and Joshua Edwards

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.

ARL-SR-0470 ● APR 2023

Dshell User Guide

Daniel E Krych
DEVCOM Army Research Laboratory

Joshua Edwards
ICF

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
April 2023	Special Report	February 2023	March 2023

4. TITLE AND SUBTITLE
Dshell User Guide

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Daniel E Krych and Joshua Edwards

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DEVCOM Army Research Laboratory
ATTN: FCDD-RLA-ND
Aberdeen Proving Ground, MD 21005

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-SR-0470

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
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.

15. SUBJECT TERMS
Network, Cyber, and Computational Sciences; Dshell; user; analyst; guide; manual; network; cyber; forensics; traffic analysis; decode; pcap; monitor; dissection

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	26

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Daniel E Krych	(410) 278-5266

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

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.

Dshell[1] 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.

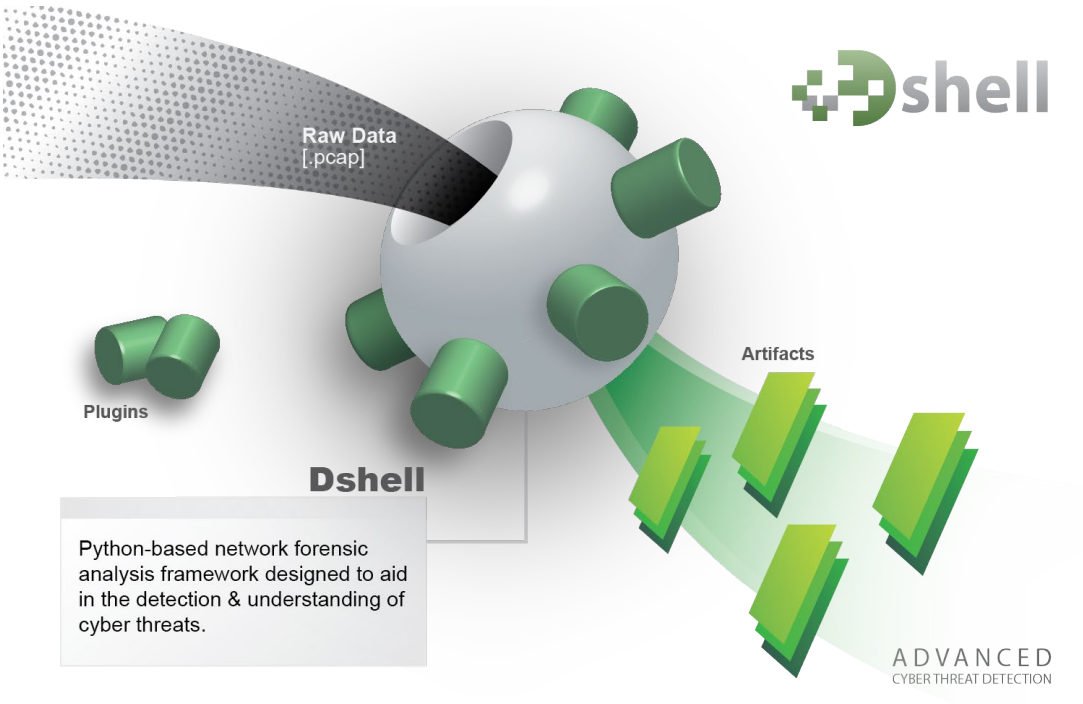

Fig. 1 Overview of Dshell—a modular, Python-based network forensic analysis framework that ingests network traffic data in the form of raw packet captures (pcap), uses plugins to parse/decode the traffic, and outputs key artifacts to aid with understanding the data

Dshell[1] 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 Dshell[1] 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.

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.[1]

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

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

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

3) Configure geoip2 by placing the MaxMind GeoLite2 data set files[8] (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[4]
- pcapy-ng[5]
- pyOpenSSL[6]
- geoip2[7]
    - MaxMind GeoIP2 data sets[8]
        - 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[9]
    - Used by some plugins that handle MAC addresses to look up organizational unique identifier (OUI) labels
    - Place in <dshell>/data/
- elasticsearch[10]
    - Used in the elasticout output module
    - Only necessary if planning to use elasticsearch to store output
- pyJA3[11]
    - 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 1 Dshell's basic usage commands and descriptions

Command	Description
decode -h	Display helpful information including generic command-line flags available to most plugins.
decode -l	List all available plugins, alongside basic information about them (flag is a lowercase "L").
decode --lo	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).
decode -p <plugin>	Display information about a plugin, including available command-line flags both commonly used and unique to that plugin.
decode -p <plugin> -h	Display information about a plugin—including all available command-line flags, flags unique to that plugin, and the plugin's long description and BPF.
decode -p <plugin> <pcap>	Run the selected plugin on a pcap file.
decode -p <plugin1>+<plugin2> <pcap>	Chain two (or more) plugins together and run them on a pcap file.
decode -p <plugin> -i <interface>	Run the selected plugin live on an interface (may require super-user privileges).

Table 2 Dshell's "decode -h" output, displaying helpful information including generic command-line flags available to most plugins

Dshell> decode -h	
usage	decode.py [options] [plugin options] file1 file2 ... fileN
Argument/flag	**description**
Positional arguments:	
files	pcap files or globs to process
Optional arguments:	
-c COUNT, --count COUNT	Number of packets to process
--debug	Show debug messages
-v, --verbose	Show informational messages
-acc, --allcc	Show all three GeoIP2 country code types (represented country/registered country/country)
-d PLUGIN, -p PLUGIN, --plugin PLUGIN	Use a specific plugin module; can be chained with '+'
--defragment	Reconnect fragmented IP packets
-h, -?, --help	Print common command-line flags and exit
-i INTERFACE, --interface INTERFACE	Listen live on INTERFACE instead of reading pcap
-l, --ls, --list	List all available plugins
-r, --recursive	Recursively process all PCAP files under input directory
--unzipdir DIRECTORY	Directory to use when decompressing input files (.gz, .bz2, and .zip only)
--version	Show program version numbers and exit
Multiprocessing arguments:	
-P, --parallel	Handle each file in separate parallel processes
-n NUMPROCS, --nprocs NUMPROCS	Define max number of parallel processes (default: 4)
Filter arguments:	
--bpf BPF	Overwrite all BPFs and use provided input. (Caution: use carefully)
--ebpf BPF	Extend existing BPFs with provided input for additional filtering. Transform input into "(<original bpf>) and (<ebpf>)"
--no-vlan	Ignore packets with VLAN headers
Output arguments:	
--lo, --list-output	List available output modules
--no-buffer	Do not buffer plugin output
-x, --extra	Appends extra data to all plugin output
-O MODULE, --omodule MODULE	Use specified output module for plugins instead of defaults (e.g., --omodule=jsonout for JSON output)
--oarg ARG=VALUE	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 "delimiter=:" --oarg "timeformat=%H %M %S"
-q, --quiet	Disable logging
-W OUTFILE	Write to OUTFILE instead of stdout
Plugin options:	
Additional options specific to a plugin are obtained by running "decode -p plugin-name -h".	

Table 3 Dshell's "decode -l" output (lowercase "L"): list of all available plugins and their basic information

Dshell> decode -l					
Module	**Name**	**Title**	**Type**	**Author**	**Description**
dshell.plugins.protocol.bitcoin	bitcoin	bitcoin	ConnectionPlugin	dek	Extract Bitcoin traffic, including Stratum mining protocol (pooled) traffic
dshell.plugins.filter.country	country	Country Filter	ConnectionPlugin	tp	Filter connections by IP address country code
dshell.plugins.flows.dataflows	dataflows	dataflows	ConnectionPlugin	amm	Display netflows that have at least one byte transferred
dshell.plugins.dhcp.dhcp	dhcp	dhcp	PacketPlugin	dek	Extract client information from DHCP messages
dshell.plugins.dns.dns	dns	DNS	DNSPlugin	bg/twp	Extract and summarize DNS queries/responses
dshell.plugins.dns.dnscc	dnscc	DNS Country Code	DNSPlugin	bg	Identify country code of DNS A/AAAA record responses
dshell.plugins.protocol.ether	ether	Ethernet	PacketPlugin	dev195	Show MAC address information and optionally filter by it
dshell.plugins.misc.followstream	followstream	Followstream	ConnectionPlugin	amm/dev195	Generates color-coded Screen/HTML output similar to Wireshark Follow Stream. Empty connections will be skipped.
dshell.plugins.ftp.ftp	ftp	ftp	ConnectionPlugin	amm,dev195	Alerts on FTP traffic and, optionally, rips files
dshell.plugins.http.httpdump	httpdump	httpdump	HTTPPlugin	amm	Dump useful information about HTTP sessions
dshell.plugins.portscan.indegree	indegree	parse indegree	ConnectionPlugin	dev195	Parse traffic to detect scanners based on connection to IPs that are rarely touched by others
dshell.plugins.dns.innuendo-dns	innuendo-dns	innuendo-dns	DNSPlugin	primalsec	Proof-of-concept detector for INNUENDO DNS channel
dshell.plugins.protocol.ip	ip	ip	PacketPlugin	twp	IPv4/IPv6 plugin
dshell.plugins.http.joomla	joomla	Joomla CVE-2015-8562	HTTPPlugin	bg	Detect attempts to enumerate MS15-034 vulnerable IIS servers
dshell.plugins.flows.largeflows	largeflows	large-flows	ConnectionPlugin	bg	Display netflows that have at least 1 MB transferred
dshell.plugins.flows.longflows	longflows	long-flows	ConnectionPlugin	bg	Display netflows that have a duration of at least 5 min
dshell.plugins.http.ms15-034	ms15-034	ms15-034	HTTPPlugin	bg	Detect attempts to enumerate MS15-034 vulnerable IIS servers
dshell.plugins.nbns.nbns	nbns	nbns	PacketPlugin	dek	Extract client information from NBNS traffic
dshell.plugins.flows.netflow	netflow	Netflow	ConnectionPlugin	dev195	Collect and display statistics about connections
dshell.plugins.misc.pcapwriter	pcapwriter	pcap writer	PacketPlugin	dev195	Generate pcap output for plugins that cannot use -o pcapout
dshell.plugins.visual.piecharts	piecharts	Pie Charts	ConnectionPlugin	dev195	Generate visualizations based on connections
dshell.plugins.protocol.protocol	protocol	Uncommon Protocols	PacketPlugin	bg	Find uncommon (i.e., not TCP, UDP, or ICMP) protocols in IP traffic

Table 3 Dshell's "decode -l" output (lowercase "L"): list of all available plugins and their basic information (continued)

Dshell> decode -l					
Module	**Name**	**Title**	**Type**	**Author**	**Description**
dshell.plugins.flows.reverseflows	reverseflows	reverse-flows	ConnectionPlugin	me	Generate an alert if the client transmits more data than the server
dshell.plugins.http.riphttp	riphttp	rip-http	HTTPPlugin	bg,twp	Rip files from HTTP traffic
dshell.plugins.voip.rtp	rtp	RTP	PacketPlugin	mm/dev195	Real-time transport protocol (RTP) capture plugin
dshell.plugins.misc.search	search	search	ConnectionPlugin	dev195	Search for patterns in connections
dshell.plugins.voip.sip	sip	SIP	PacketPlugin	mm/dev195	(UNFINISHED) Session Initiation Protocol (SIP) capture plugin
dshell.plugins.dns.specialips	specialips	special-ips	DNSPlugin	dev195	Identify DNS resolutions that fall into special IP (IPv4 and IPv6) spaces (i.e., private, reserved, loopback, multicast, link–local, or unspecified)
dshell.plugins.ssh.ssh-pubkey	ssh-pubkey	ssh-pubkey	ConnectionPlugin	amm	Extract server SSH public key from key exchange
dshell.plugins.misc.sslalerts	sslalerts	sslalerts	ConnectionPlugin	dev195	Look for SSL alert messages
dshell.plugins.ssl.sslblacklist	sslblacklist	sslblacklist	ConnectionPlugin	dev195	Look for certificate SHA1 matches in the abuse.ch blacklist
dshell.plugins.malware.sweetorange	sweetorange	sweetorange	HTTPPlugin	dev195	Decode certain variants of the Sweet Orange exploit kit redirect traffic
dshell.plugins.misc.synrst	synrst	SYN/RST	PacketPlugin	bg	Detect failed attempts to connect (SYN followed by RST/ACK)
dshell.plugins.tftp.tftp	tftp	tftp	PacketPlugin	dev195	Find TFTP streams, and optionally extract the files
dshell.plugins.ssl.tls	tls	tls	ConnectionPlugin	amm	Extract interesting metadata from TLS connection setup
dshell.plugins.flows.toptalkers	toptalkers	Top Talkers	ConnectionPlugin	dev195	Find top-talkers based on byte count
dshell.plugins.filter.track	track	track	ConnectionPlugin	twp,dev195	Only follow connections that match user-provided IP addresses and ports
dshell.plugins.portscan.trw	trw	trw	PacketPlugin	dev195	Use Threshold Random Walk to detect network scanners
dshell.plugins.http.web	web	web	HTTPPlugin	bg,twp	Display basic information for web requests/responses in a connection
dshell.plugins.wifi.wifi80211	wifi80211	802.11	PacketPlugin	dev195	Show 802.11 packet information
dshell.plugins.wifi.wifibeacon	wifibeacon	Wi-fi Beacons	PacketPlugin	dev195	Show SSIDs of 802.11 wireless beacons
dshell.plugins.misc.xor	xor	xor	ConnectionPlugin	twp,dev195	XOR every packet with a given key

Table 4 Dshell's "decode --lo" output (lowercase "L"): list of available output modules and their basic information

Dshell> decode --lo	
Module	**Description**
alertout	Default format for printing a single-line alert
colorout	Reconstructed output with ANSI color codes
csvout	CSV format output
elasticout	Automatically insert data into an elasticsearch instance
htmlout	HTML format output
jsonout	JSON format output
netflowout	Flow (connection overview) format output
pcapout[a]	Writes data to a pcap file (does not work with connection-based plugins). Use the pcapwriter plugin for more control over pcap output.

[a] 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

Dshell> decode -p netflow -h
(All output from "-h" flag displayed here)
########## Netflow
Collects and displays statistics about connections
Default BPF: "ip or ip6"
Dshell> decode -p sweetorange -h
(All output from "-h" flag 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.
########## 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[12]

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

```
[DNS] 2005-03-30 03:47:46  192.168.170.8:32795 --  192.168.170.20:53  **
ID: 4146, TXT? google.com., TXT: b'\x0fv=spf1 ptr ?all' **
[DNS] 2005-03-30 03:47:50  192.168.170.8:32795 --  192.168.170.20:53  **
ID: 63343, MX? google.com., MX: b'\x00(\x05smtp4\xc0\x0c', MX:
b'\x00\n\x05smtp5\xc0\x0c', MX: b'\x00\n\x05smtp6\xc0\x0c', MX:
b'\x00\n\x05smtp1\xc0\x0c', MX: b'\x00\n\x05smtp2\xc0\x0c', MX:
b'\x00(\x05smtp3\xc0\x0c' **
[DNS] 2005-03-30 03:47:59  192.168.170.8:32795 --  192.168.170.20:53  **
ID: 18849, LOC? google.com. **
[DNS] 2005-03-30 03:48:07  192.168.170.8:32795 --  192.168.170.20:53  **
ID: 39867, PTR? 104.9.192.66.in-addr.arpa., PTR: 66-192-9-
104.gen.twtelecom.net. **
[DNS] 2005-03-30 03:49:18  192.168.170.8:32795 --  192.168.170.20:53  **
ID: 30144, A? www.netbsd.org., A: 204.152.190.12 (ttl 82159s) **
[DNS] 2005-03-30 03:49:35  192.168.170.8:32795 --  192.168.170.20:53  **
ID: 61652, AAAA? www.netbsd.org., AAAA: 2001:4f8:4:7:2e0:81ff:fe52:9a6b
(ttl 86400s) **
```

```
[DNS] 2005-03-30 03:50:35  192.168.170.8:32795 --  192.168.170.20:53  **
ID: 32569, AAAA? www.netbsd.org., AAAA: 2001:4f8:4:7:2e0:81ff:fe52:9a6b
(ttl 86340s) **
[DNS] 2005-03-30 03:50:44  192.168.170.8:32795 --  192.168.170.20:53  **
ID: 36275, AAAA? www.google.com., CNAME: 'www.l.google.com.' **
[DNS] 2005-03-30 03:50:54  192.168.170.8:32795 --  192.168.170.20:53  **
ID: 56482, AAAA? www.l.google.com. **
[DNS] 2005-03-30 03:51:35  192.168.170.8:32795 --  192.168.170.20:53  **
ID: 48159, AAAA? www.example.com. **
[DNS] 2005-03-30 03:51:46  192.168.170.8:32795 --  192.168.170.20:53  **
ID: 9837, AAAA? www.example.notginh., NXDOMAIN **
[DNS] 2005-03-30 03:52:17  192.168.170.8:32795 --  192.168.170.20:53  **
ID: 65251, AAAA: 2001:4f8:0:2::d (ttl 600s), A: 204.152.184.88 (ttl
600s) **
[DNS] 2005-03-30 03:52:17  192.168.170.8:32796 --  192.168.170.20:53  **
ID: 23123, PTR? 1.0.0.127.in-addr.arpa., PTR: localhost. **
[DNS] 2005-03-30 03:52:17  192.168.170.8:32797 --  192.168.170.20:53  **
ID: 8330, NS: b'\x06ns-ext\x04nrt1\xc0\x0c', NS: b'\x06ns-
ext\x04sth1\xc0\x0c', NS: b'\x06ns-ext\xc0\x0c', NS: b'\x06ns-
ext\x04lga1\xc0\x0c' **
[DNS] 2005-03-30 03:52:17  192.168.170.56:1707 --   217.13.4.24:53  **
ID: 12910, SRV? _ldap._tcp.Default-First-Site-
Name._sites.dc._msdcs.utelsystems.local., NXDOMAIN **
[DNS] 2005-03-30 03:52:17  192.168.170.56:1708 --   217.13.4.24:53  **
ID: 61793, SRV? _ldap._tcp.dc._msdcs.utelsystems.local., NXDOMAIN **
[DNS] 2005-03-30 03:52:17  192.168.170.56:1709 --   217.13.4.24:53  **
ID: 33633, SRV? _ldap._tcp.05b5292b-34b8-4fb7-85a3-
8beef5fd2069.domains._msdcs.utelsystems.local., NXDOMAIN **
[DNS] 2005-03-30 03:52:17  192.168.170.56:1710 --   217.13.4.24:53  **
ID: 53344, A? GRIMM.utelsystems.local., NXDOMAIN **
[DNS] 2005-03-30 03:52:25  192.168.170.56:1711 --   217.13.4.24:53  **
ID: 30307, A? GRIMM.utelsystems.local., NXDOMAIN **
```

5.2 Following and Reassembling a Stream in Sample Traffic[12]

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
2001:6f8:102d:0:2d0:9ff:fee3:e8de: 59201 -> 2001:6f8:900:7c0::2:  80
(300 bytes)
2001:6f8:900:7c0::2:  80 -> 2001:6f8:102d:0:2d0:9ff:fee3:e8de: 59201
(2379 bytes)

GET / HTTP/1.0
Host: cl-1985.ham-01.de.sixxs.net
Accept: text/html, text/plain, text/css, text/sgml, */*;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
```

```
Server: Apache
Content-Length: 2121
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
 <head>
 <title>Index of /</title>
 </head>
 <body>
<h1>Index of /</h1>
<pre><img src="/icons/blank.gif" alt="Icon "> <a
href="?C=N;O=D">Name</a>          <a href="?C=M;O=A">Last modified</a>
<a href="?C=S;O=A">Size</a> <a href="?C=D;O=A">Description</a><hr><img
src="/icons/folder.gif" alt="[DIR]"> <a href="202-vorbereitung/">202-
vorbereitung/</a>    06-Jul-2007 14:31  -
<img src="/icons/layout.gif" alt="[  ]"> <a
href="Efficient_Video_on_demand_over_Multicast.pdf">Efficient_Video_on_d
..&gt;</a> 19-Dec-2006 03:17 291K
<img src="/icons/unknown.gif" alt="[  ]"> <a
href="Welcome%20Stranger!!!">Welcome Stranger!!!</a>   28-Dec-2006 03:46
0
<img src="/icons/text.gif" alt="[TXT]"> <a
href="barschel.htm">barschel.htm</a>      31-Jul-2007 02:21  44K
<img src="/icons/folder.gif" alt="[DIR]"> <a href="bnd/">bnd/</a>
30-Dec-2006 08:59  -
<img src="/icons/folder.gif" alt="[DIR]"> <a href="cia/">cia/</a>
28-Jun-2007 00:04  -
<img src="/icons/layout.gif" alt="[  ]"> <a href="cisco_ccna_640-
801_command_reference_guide.pdf">cisco_ccna_640-801_c..&gt;</a> 28-Dec-
2006 03:48 236K
<img src="/icons/folder.gif" alt="[DIR]"> <a href="doc/">doc/</a>
19-Sep-2006 01:43  -
<img src="/icons/folder.gif" alt="[DIR]"> <a
href="freenetproto/">freenetproto/</a>      06-Dec-2006 09:00  -
<img src="/icons/folder.gif" alt="[DIR]"> <a
href="korrupt/">korrupt/</a>        03-Jul-2007 11:57  -
<img src="/icons/folder.gif" alt="[DIR]"> <a
href="mp3_technosets/">mp3_technosets/</a>     04-Jul-2007 08:56  -
<img src="/icons/text.gif" alt="[TXT]"> <a
href="neues_von_rainald_goetz.htm">neues_von_rainald_go..&gt;</a> 21-
Mar-2007 23:27  31K
<img src="/icons/text.gif" alt="[TXT]"> <a
href="neues_von_rainald_goetz0.htm">neues_von_rainald_go..&gt;</a> 21-
Mar-2007 23:29  36K
<img src="/icons/layout.gif" alt="[  ]"> <a
href="pruef.pdf">pruef.pdf</a>        28-Dec-2006 07:48  88K
<hr></pre>
</body></html>
```

5.3 Chaining Plugins to View Flow Data for a Specific Country Code In Sample Traffic[12]

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

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

```
2006-08-25 15:32:20.766761    192.168.1.2 -> 202.232.205.123 (-- -> JP)
UDP  60583  33438   1   0    64    0 0.0000s
2006-08-25 15:32:20.634046    192.168.1.2 -> 202.232.205.123 (-- -> JP)
UDP  60583  33435   1   0    64    0 0.0000s
2006-08-25 15:32:20.747503    192.168.1.2 -> 202.232.205.123 (-- -> JP)
UDP  60583  33437   1   0    64    0 0.0000s
2006-08-25 15:32:20.651501    192.168.1.2 -> 202.232.205.123 (-- -> JP)
UDP  60583  33436   1   0    64    0 0.0000s
```

5.4 Collecting DNS Traffic from Several Files[12] 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)
15:36:09.011208 IP 192.168.1.1.53 > 192.168.1.2.2131: 40210 0/1/0 (94)
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)
15:36:10.961608 IP 192.168.1.2.2131 > 192.168.1.1.53: 40211+ AAAA?
ui.skype.com. (30)
15:36:11.294333 IP 192.168.1.1.53 > 192.168.1.2.2131: 40211 0/1/0 (94)
15:32:21.664798 IP 192.168.1.2.2130 > 192.168.1.1.53: 39862+ A?
ui.skype.com. (30)
15:32:21.664913 IP 192.168.1.2.2130 > 192.168.1.1.53: 39863+ AAAA?
ui.skype.com. (30)
```

5.5 Collecting TFTP Data and Converting Alerts to Different Formats Using Sample Traffic[12]

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

```
[tftp] 2013-05-01 08:24:11  192.168.0.253:50618 --   192.168.0.10:3445
** read rfc1350.txt (24599 bytes) **
```

```
[tftp] 2013-04-27 05:07:59   192.168.0.1:57509 --   192.168.0.13:2087 **
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

```
'tftp','2013-05-01
08:24:11','192.168.0.253',50618,'192.168.0.10',3445,'read rfc1350.txt
(24599 bytes) '
'tftp','2013-04-27
05:07:59','192.168.0.1',57509,'192.168.0.13',2087,'write rfc1350.txt
(24599 bytes) '
```

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
```

```
192.168.0.253:50618 -> 192.168.0.10:3445 ( bytes)
</h2>
<span style="color:blue;">read rfc1350.txt (24599 bytes) </span>
<h1>Packet 2 ()</h1><h2>Start: 2013-04-27 05:07:59
192.168.0.1:57509 -> 192.168.0.13:2087 ( bytes)
</h2>
<span style="color:blue;">write rfc1350.txt (24599 bytes) </span>

</body>
</html>
```

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

```
{"ts": 1367411051.972852, "sip": "192.168.0.253", "sport": 50618, "dip":
"192.168.0.10", "dport": 3445, "readwrite": "read", "filename":
"rfc1350.txt", "plugin": "tftp", "pcapfile": "/home/pcap/tftp_rrq.pcap",
"data": "read rfc1350.txt (24599 bytes) "}
{"ts": 1367053679.45274, "sip": "192.168.0.1", "sport": 57509, "dip":
"192.168.0.13", "dport": 2087, "readwrite": "write", "filename":
"rfc1350.txt", "plugin": "tftp", "pcapfile": "/home/pcap/tftp_wrq.pcap",
"data": "write rfc1350.txt (24599 bytes) "}
```

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

```
2013-05-01 08:24:11   192.168.0.253 ->   192.168.0.10 ( -> )     50618
3445
2013-04-27 05:07:59    192.168.0.1 ->   192.168.0.13 ( -> )     57509
2087
```

5.6 Using the pcapout Output Format and Comparing the Original and New pcap Using Sample Traffic[12]

Dshell> decode -p dhcp -O pcapout ~/pcap/dhcp.pcap -W test_pcapout.pcap

```
Dshell> tcpdump -r ~/pcap/dhcp.pcap
reading from file /home/pcap/dhcp.pcap, link-type EN10MB (Ethernet)
14:16:24.317453 IP 0.0.0.0.bootpc > broadcasthost.bootps: BOOTP/DHCP,
Request from 00:0b:82:01:fc:42 (oui Unknown), length 272
14:16:24.317748 IP 192.168.0.1.bootps > 192.168.0.10.bootpc: BOOTP/DHCP,
Reply, length 300
14:16:24.387484 IP 0.0.0.0.bootpc > broadcasthost.bootps: BOOTP/DHCP,
Request from 00:0b:82:01:fc:42 (oui Unknown), length 272
14:16:24.387798 IP 192.168.0.1.bootps > 192.168.0.10.bootpc: BOOTP/DHCP,
Reply, length 300

Dshell> tcpdump -r test_pcapout.pcap
reading from file test_pcapout.pcap, link-type EN10MB (Ethernet)
14:16:24.317452 IP 0.0.0.0.bootpc > broadcasthost.bootps: BOOTP/DHCP,
Request from 00:0b:82:01:fc:42 (oui Unknown), length 272
14:16:24.317748 IP 192.168.0.1.bootps > 192.168.0.10.bootpc: BOOTP/DHCP,
Reply, length 300
14:16:24.387484 IP 0.0.0.0.bootpc > broadcasthost.bootps: BOOTP/DHCP,
Request from 00:0b:82:01:fc:42 (oui Unknown), length 272
```

```
14:16:24.387798 IP 192.168.0.1.bootps > 192.168.0.10.bootpc: BOOTP/DHCP,
Reply, length 300
```

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[12]

```
# 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

1. Dshell. DEVCOM Army Research Laboratory. 2020 [accessed 2023 Mar 24]. https://github.com/USArmyResearchLab/Dshell.
2. Sunsetting Python 2. Python Software Foundation; n.d. [accessed 2023 Mar 24]. https://www.python.org/doc/sunset-python-2/.
3. Edwards J, Krych DE. Dshell developer guide. DEVCOM Army Research Laboratory (US); 2023 Apr. Report No.: ARL-SR-0471.
4. Stahn M. pypacker, Project ID: 6715044. n.d. [accessed 2023 Mar 24]. https://gitlab.com/mike01/pypacker.
5. pcapy-ng. GitHub, Inc; n.d. [accessed 2023 Mar 24]. https://github.com/stamparm/pcapy-ng/.
6. pyopenssl. GitHub, Inc; n.d. [accessed 2023 Mar 24]. https://github.com/pyca/pyopenssl.
7. GeoIP2-python. GitHub, Inc; n.d. [accessed 2023 Mar 24]. https://github.com/maxmind/GeoIP2-python.
8. GeoLite2 Free Geolocation Data. MaxMind, Inc; n.d. [accessed 2023 Mar 24]. https://dev.maxmind.com/geoip/geolite2-free-geolocation-data.
9. IEEE OUI Standard. n.d. [accessed 2023 Mar 24]. https://standards-oui.ieee.org/oui/oui.txt.
10. Elasticsearch Python Client. Elasticsearch B.V.; n.d. [accessed 2023 Mar 24]. https://www.elastic.co/guide/en/elasticsearch/client/python-api/current/index.html.
11. pyJA3. GitHub, Inc; n.d. [accessed 2023 Mar 24]. https://github.com/salesforce/ja3/tree/master/python.
12. SampleCaptures. WireShark; n.d. [accessed 2023 Mar 24]. https://wiki.wireshark.org/SampleCaptures.

List of Symbols, Abbreviations, and Acronyms

ANSI	American National Standards Institute
ARL	Army Research Laboratory
BPF	Berkeley Packet Filter
CSV	comma-separated values
DEVCOM	US Army Combat Capabilities Development Command
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
Dshell	decoder-shell
FTP	file transfer protocol
HTML	hypertext markup language
HTTP	hypertext transfer protocol
ICMP	Internet control message protocol
IIS	Internet Information Services
IP	Internet protocol
JSON	JavaScript Object Notation
NBNS	NetBIOS Name Service
OS	operating system
OUI	organizational unique identifier
pcap	packet capture
RTP	real-time transport protocol
SIP	session initiation protocol or source IP
SSH	secure shell
SSID	service set identifier
SSL	secure sockets layer
TCP	transmission control protocol
TFTP	trivial file transfer protocol

TLS	transport layer security
UDP	user datagram protocol
VLAN	virtual local area network

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CI TECH LIB
2 (PDF)	DEVCOM ARL FCDD RLA ND D KRYCH J EDWARDS