FASHIONCLEFT

Interface Control Document

Updated: 15 June 2009

Document Number: TAO.DNT_SE07_005_V1.1

28 March 2005

Last Saved 4/17/2013 09:08:00 PM
Last Printed 6/15/2009 02:15:00 PM

Table of Contents

(U//FOUO) FASHIONCLEFT ICD

1. (U) Introduction

1.1 (U) Document Description

(S//SI) This document defines the FASHIONCLEFT Computer Network Exploitation (CNE) collection protocol that enables raw data packets from the external Internet to interface to the Common Data Receptor (CDR) of Data Network Technology's (DNT's) Common System Architecture (CSA). This interface will be deployed to support Tailored Access Operations (TAO) On-Net operations, specifically implants that are required to exfiltrate collected network packets, but due to implant processing and bandwidth constraints are required to utilize a low overhead egress protocol.

(S//SI) This document defines the interface requirement that the Common Data Receptor and implant must support to enable successful reception of raw data packets sent by the implant. It includes conceptual data flow diagrams and network diagrams to define the system. It also contains detailed data format examples, data field definitions, and a description of the protocol.

1.2 (U // FOUO) Reference Documents

1. FLAXENPRECEPT External (source-to-CDR) Interface Control Document (Front End EICD), NSA/DNT Doc # TAO.DNT_SE07_001_V1.0, 16 June 2003.

2. FLAXENPRECEPT External Data Flow (CDR-to-NSA Corporate) Interface Control Document (Back End EICD), NSA/DNT Doc # TAO.DNT_SE07_002_V0.01

3. SHELLGREY binary metadata tag standard: [redacted]

1.3 (U) Background

(S//SI) The Common Data Receptor (CDR) is a DNT-designed system to perform the data reception portion of what is commonly termed a 'Listening Post.' The CDR supports a common interface that DNT developers use for data formatting and reception. In addition, the Common Data Receptor concept emphasizes the use of a Data Receptor, Operations Manager, and Network Manager vice a Listening Post.

(S//SI//X1) The introduction of the FASHIONCLEFT protocol is necessary because without the FASHIONCLEFT protocol, the implant-to-CDR communication previously did not handle raw IP data packets. In addition, an implant previously had no way to deal with the challenge of processing streaming data while minimizing the computing overhead associated with encapsulating and encrypting the data. This new protocol provides the developer a mechanism to specify the level of encryption implemented in the implant as a precursor to data transmission.

(S//SI//X1) The FASHIONCLEFT protocol supports the passing of metadata securely and with authentication.

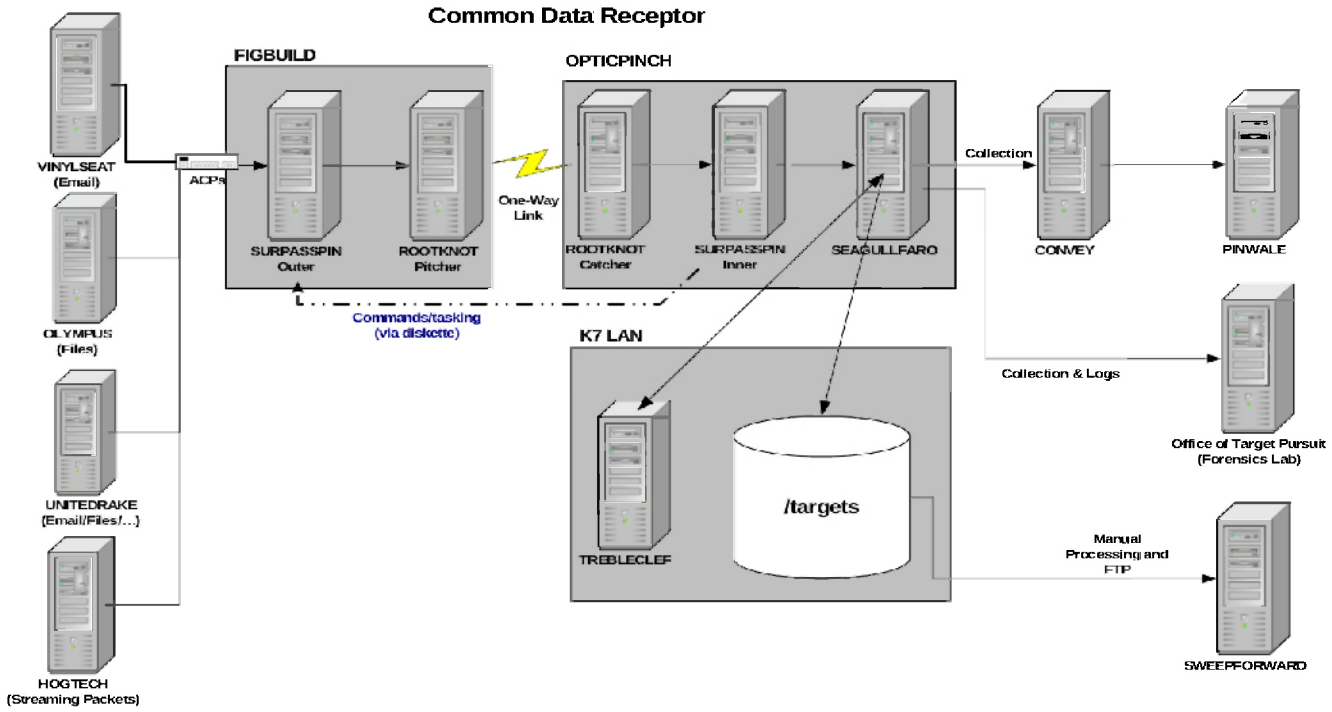
(S//SI//X1) The metadata is used to support the reconstruction of IP packets and when applicable to convey information about the session.

(S//SI//X1) The CDR Architecture includes the Outer SURPASSPIN, Inner SURPASSPIN and SEAGULLFARO subsystems. In addition, a one-way transfer device — ROOTKNOT (composed of Pitcher, Catcher, and their one-way link) — supplants the air gap previously employed (that required manual transfer by diskette). In order to support this configuration, DNT has been working closely with the Remote Operations Center (ROC) to design and build FIGBUILD (an external mission network) and OPTICPINCH (an internal mission network).

(S//SI) Figure 1.3-1 below shows the Common Data Receptor Architecture as described in the referenced FLAXENPRECEPT EICDs.

(S//SI) The interfaces supported by the CDR must also support the CDR's separation of classified and unclassified processing. Classified information, such as egress data and metadata must be protected while on the unclassified network side before passing through the one-way link. This interface also supports the CDR protection of classified information.

Figure 1.3-1 Common Data Receptor Architecture



1.4 (U//FOUO) Interface Description

(S//SI) This interface provides TAO developers a low overheard solution to exfiltrate streaming data, typically associated with implants that collect raw packets from the target environment, while providing the appropriate metadata and authentication. The basic design principle behind this low overhead approach is to redirect a copy or clone of the original packet to another host, or listening post (in this case the CDR). In-order to redirect a packet, features of the copy of the packet must be altered — particularly the source and destination addresses, and possibly the port numbers.

(S//SI) This interface is designed to support the reception of "cloned" packets and their associated metadata originating from implants, while supporting authentication of the egress data and correlation of received packets to the appropriate metadata. To provide greater flexibility for the implant, several configurable levels of encryption are supported.

(S//SI) The following assumptions are made:

- This interface supports egress of data and does not support a command and control channel.
- This interface is used by implants which for valid reasons are unable to implement egress methods that require greater resources from the target, such as encapsulation.
- Egress data must be associated with encrypted, authenticated metadata.
- This interface may not be suitable for all boundary defenses that could be encountered within a target environment. Extensibility has been designed to allow additional packet selection algorithms to be added, such that future changes to skirt other defenses can be readily included.

(S//SI) This interface implements four functions:

- Transmission of session announcements,
- Transmission of metadata,
- Transmission of data packets,
- Reconstruction of data packets using the metadata, and
- Reconstruction of sessions using metadata to track and collate data packets belonging to a particular session.

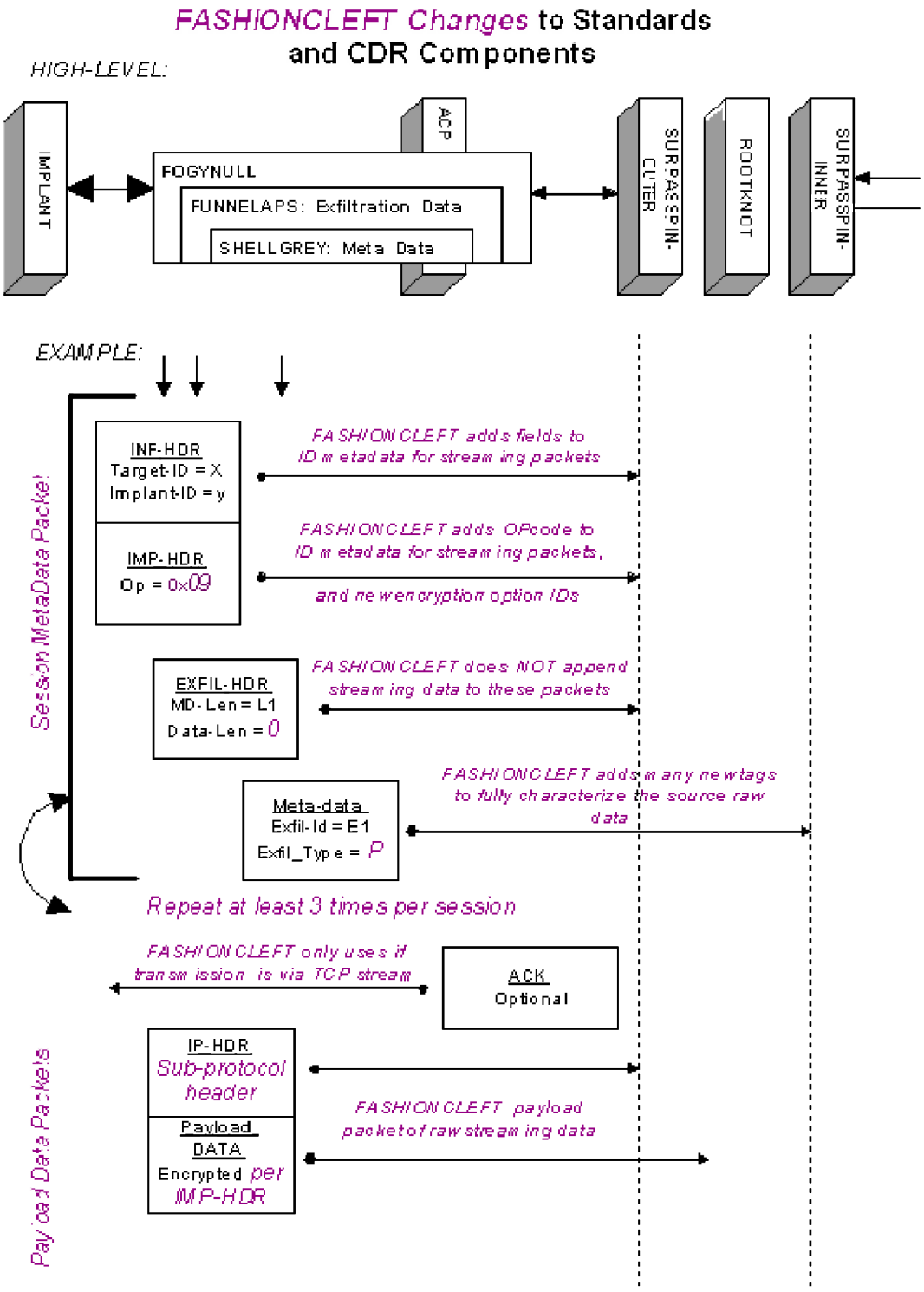
(S//SI) During session establishment, an implant authenticates itself to the CDR and requests the CDR to begin reception of data packets that belong to this session. As part of establishing the session, metadata about the session is passed, along with metadata that the CDR requires to assign data packets to the session and collate data packets within the session. The session establishment function uses the FOGYNULL protocol family to authenticate and securely pass metadata, packet collation information and data packet encryption status.

(S//SI) There are several methods that an implant could use to collate data packets to the appropriate session, but only one method can be used in any session. The implant picks the appropriate Exfil Type — either "Packet" or "Session," and either exfiltrates one modified copy of a source packet per source packet, or assembles the source data payloads from several source packets into one packet to be exfiltrated. The FASHIONCLEFT interface design allows these methods to be extensible, so additional methods can later be developed and included to meet operational constraints imposed by boundary defenses.

(S//SI) The data packets are generally a copy of a packet that was collected, and had its destination address changed to send it to the CDR. Additional fields might have been altered to support one of the methods of assigning data packets to a session and collating data within a session. If selected during the session establishment, the payload section of the packet may be encrypted, using the mode specified by the FASHIONCLEFT protocol encryption mode selection in the FOGYNULL implant header.

(S//SI) Figure 1.4-1 below shows how the FASHIONCLEFT protocol modifies the conceptual diagram of the Common Data Receptor Architecture as described in the referenced FLAXENPRECEPT EICDs. *FASHIONCLEFT changes* to the protocols previously defined for this architecture are shown in *italics* in Figure 1.4-1 and in subsequent tables listing *FASHIONCLEFT changes*.

Figure 1.4-1 Common Data Receptor with FASHIONCLEFT



2. (U//FOUO) FASHIONCLEFT Protocol Details

(S//SI) Metadata is used to announce the initiation of a session, to describe the session, and to provide original packet reconstruction information.

(S//SI) From the point of view of an implant, a streaming data transmission session is accomplished in two steps, first sending the session announcement metadata, followed by the continuous sending of the streaming data packets.

2.1. (U//FOUO) FASHIONCLEFT Session Announcement

(S//SI) FASHIONCLEFT session announcement metadata announces the initiation of a session, describes the session, and conveys data packet reconstruction (and association) information to enable data packets and sessions to be reconstructed after collection.

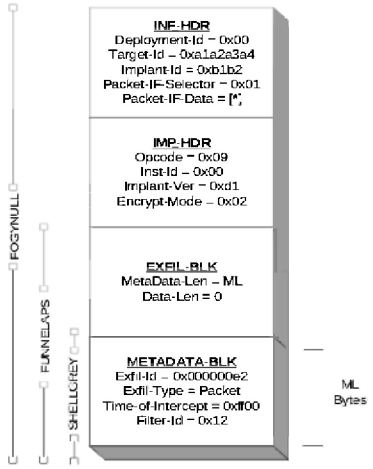
2.1.1. (U//FOUO) Session Announcement Description

(S//SI) FASHIONCLEFT adds two new fields in the FOGYNULL protocol infrastructure header to proclaim that data packets containing streaming data are to be sent using the FASHIONCLEFT protocol (Packet-IF-Selector, Packet-IF-Data); and adds a new OpCode tag (*0x09*) plus new encryption mode tags in the FOGYNULL protocol implant header to indicate that raw packets will be exfiltrated and the extent to which they are being encrypted.

(S//SI) FASHIONCLEFT also adds new tags in the SHELLGREY protocol that fully characterize the original source and original destination of streaming data packets that follow. In addition, FASHIONCLEFT uses the SHELLGREY EXFIL_TYPE metadata tag to indicate whether the target data will be exfiltrated as:

- "Packet data" (one exfil data packet per cloned target packet) or
- "Session data" (each exfil data packet contains target data payload that may have been captured prior to being encapsulated by the target source into multiple packets for transmission)

Example: Session Announcement



(S//SI) The structure of session announcement is comprised of the FOGYNULL Headers, both the Infrastructure and Implants headers; followed by SHELLGREY Meta-data, format within a FUNNELAPS protocol with no data. All multi-byte fields in the Infrastructure and Implant Headers are in network byte order (i.e. big endian).

2.1.2. (U//FOUO) FOGYNULL Headers

2.1.2.1. (U//FOUO) Infrastructure Header

(S//SI) The purpose of the following three elements of the infrastructure header (version 2 and later of FOGYNULL) is to indicate the characteristics of the origination of the packet:

- Implant ID
- Target ID
- Deployment ID (allow different encryption keys for same {ImplantID, TargetID})

(S//SI) The purpose of the next two elements of the infrastructure header (FASHIONCLEFT addition to FOGYNULL) is as follows:

Table 2.1.2.1-1: Format of Infrastructure Header Block

	0	31
+0 …	SHA-1 HASH (data bytes 0-19) Of bytes 20-127	
+20	Random	
+28	Infrastructure Header Version (0x02)	Implant ID
+32	Length	
+36	Target ID	
+40	Timestamp (sec.)	
+44	Sequence #	
+48	Deployment ID	
+52	Packet I/F Selector (*See Table 3.1-1*)	+54
+56 …	Packet I/F Data (32 bytes)	*+86*
+86 …	Random	
+124	Random (special constraints RSA)	

(S//SI) In the FOGYNULL protocol, additional elements follow to complete the 128 bytes of the infrastructure header.

- Packet I/F Selector (This field directly supports FASHIONCLEFT exfiltration.)
- Packet I/F Data (This new field uses the first 32 bytes of a formerly "random" field.)

(TS//SI) The purpose of the **new** Packet I/F Selector [0x0n] field is to allow selection of a choice of algorithm that will collate the data to the sessions. The following three choices are provided:

- No Data packets follow [0x00]

- Generic Pattern Matching Selector
- Fixed IP4 Field Matching Selector

(C//SI) The details of these selections are further discussed in Section 3. Because the choice values are extensible, it could be possible to add additional choices in the future.

(C//SI) Contents of the packet I/F data field depend on the choice previously selected in the Packet I/F Selector field.

2.1.2.2. (C//SI) Implant Header

(TS//SI) The implant header (total of 128 bytes) is composed of five parts:

- OpCode [0x09] (new value indicating Raw Packet exfil)
- Instance-ID (implant generated; see SHELLGREY Exfil-ID below)
- Implant-version (as defined by FOGYNULL)
- Encrypt-Mode (**new** values for FOGYNULL encryption mode)

Table 2.1.2.2-1 (S/SI) Generic Implant Header Block

	0 31	
0	SHA-1 HASH (data bytes 0-19) Of bytes 20-127	
+20	Random	
+28	Implant Header Version (0x02)	Implant Operation
+32	Instance ID	
+36	Implant Version	Encryption Mode (0x0001) *see Table 2.1.2.2-2, page 9*
+40	Data Length (0x00)	
+44	Random	
+52	RC6 Session Key	
+68	Random	
+76	RC6 Message Indicator (MI)	
+92	Random	
+124	Random (special constraints RSA)	

(S//SI) New encryption modes and options for encryption of data packets are added to the FOGYNULL authentication protocol, as shown in Table 2.1.2.2-2. The new values are italicized and their cells are highlighted:

Table 2.1.2.2-2 (S/SI) New Encryption Modes

Value	Session	Exfil	*Packet (Proposed)*	*Packet (Implemented)*
0x01	RC6/Session	RC6/Composite	NA	*Default*
0x02	RC6/Session	RC6/Composite	*None*	*NA*
0x03	RC6/Session	RC6/Composite	*"munge"*	*NA*
0x04	RC6/Session	RC6/Composite	*RC6/Composite*	*NA*

(S//SI) FASHIONCLEFT allows a new Exfil-Type = Packet. It was *originally proposed* that three new encryption modes would be used for Packet exfil, using the values 0x02 through 0x04; however, this proposed version was not actually implemented. Applying no encryption is just what it implies. The term "munge" implies that privacy scrambling will be applied to the data to make it a little less tractable to anyone trying to detect a third party presence in the target system. The RC6/Composite mode uses full 128-bit encryption as described in section 2.3.2.

(S//SI) As *implemented*, FASHIONCLEFT Packet exfil should always use Encryption Mode = 0x01 = Default. The actual encryption mode used for Packet exfil is then specified within the Metadata Block using SHELLGREY tags, as described in section 2.3.2.

2.1.3. (U//FOUO) Payload (Content Information) Blocks

(C//SI) The payload (FUNNELAPS CNE protocol data) characterizing the contents of the data packets is composed of two blocks — a 16-byte Exfil Header and a variable length Metadata Block that conforms to the SHELLGREY CNE data protocol. There is no Exfil Data block in this payload, and the contained data length is therefore reflected with a zero length indicator in the Exfil Header.

2.1.3.1. (C//SI) Content of the Exfil Header

(S//SI) The Exfil Header is a 16-byte block that contains the following fields:

Table 2.1.3.1-1 (S/SI) Exfil Header

Byte:	0	1	2	3
0	Version (0x0001)		Checksum	
4	Metadata Length			
8	Data Length = 0x00000000			
12	Authentication			

- **Version**: The version of this header.
- **Checksum**: A two byte CRC-16 checksum of the encoded Metadata Block.
- **Metadata Length:** A four byte unsigned integer in network byte order that represents the number of bytes in the encoded Metadata Block (including any padding).
- **Data Length:** A four byte unsigned integer in network-byte order that represents the number of bytes that comprises the Exfil Data block. The Data Length value used for FASHIONCLEFT must be 0 since there is no Exfil Data within the Session Announcement.
- **Authentication:** A copy of bytes 48-51 of the Implant Header; a match validates the successful decryption of the Exfil Header.

2.1.3.2. (U//FOUO) Content of the Metadata Block

(TS//SI) The metadata block, encoded according to the SHELLGREY metadata protocol described in References [1 and 1.2] , contains the collection-related information and packet reconstruction information. The required collection-related items are:

- Exfil-ID [must be unique for {ImplantID, TargetID, DeploymentID, InstanceID}]
- Exfil-Type [Packet or Session]
- Time-of-Intercept
- Filter-ID

(TS//SI) The reconstruction-related metadata was originally contained in the following seven tags that show the original source and routing of the packets that were exfiltrated. These tags are now provided only for informational purposes and are not used to reconstruct the original packet:

- Exfil-Sub-Type (optional field—0x01indicates the data type is VoIP—type identifiers can be extended as necessary)
- IP4-Source-Addr
- IP4-Destination-Addr
- IP4-Source-Port

- IP4- Destination -Port
- IP4-TTL
- IP-Sub-Protocol (as defined by the IP protocol)

(TS//SI) Reconstruction is now explicitly specified using SHELLGREY reconstruction tags such as (see reference [1.2] for a complete list):

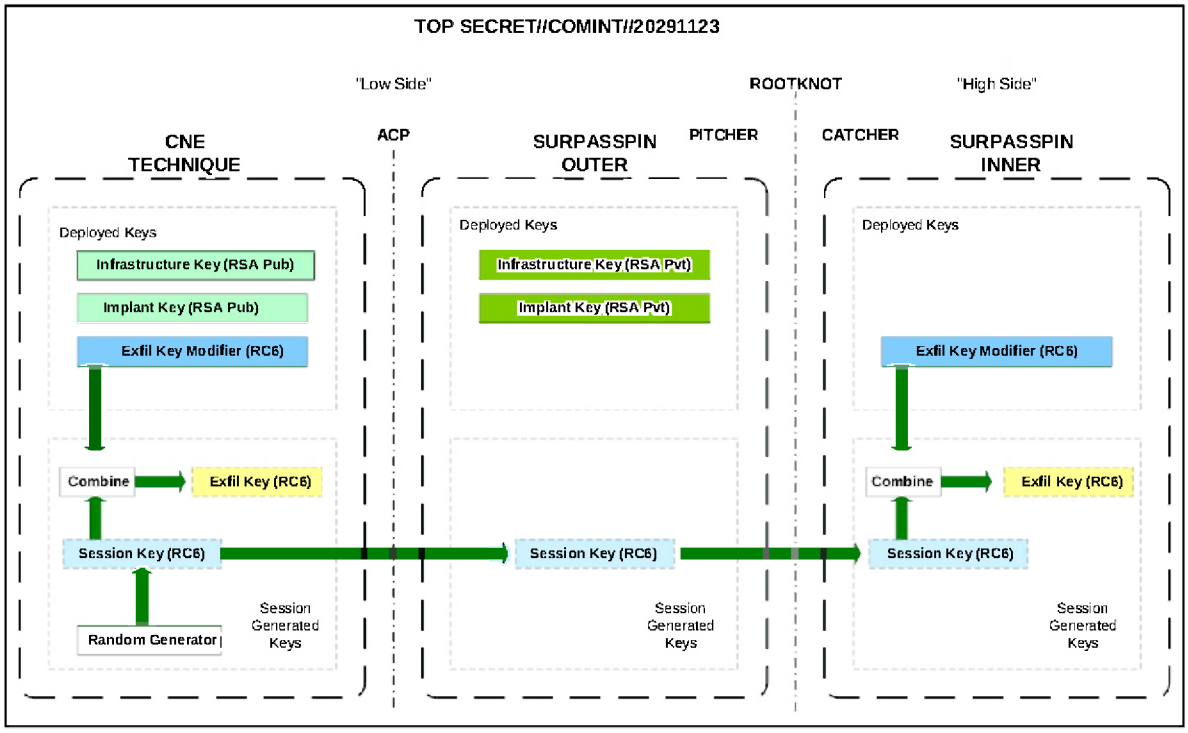
- PACKET_BIT_REPLACE
- PACKET_BIT_REPLACE_IP4
- PACKET_BYTE_REMOVE
- PACKET_BYTE_REMOVE_IP4
- PACKET_BYTE_REPLACE
- PACKET_BYTE_REPLACE_IP4
- PACKET_IP_ADDR

(TS//SI) SHELLGREY reconstruction and decryption tags (section 2.3.2) are processed in the order in which they occur. The reconstruction result is therefore highly dependent on the specific reconstruction tags used and the order in which they occur. This flexibility allows the sender (the implant) to specify how to transform the exfiltrated data and the receiver simply follows instructions without needing any knowledge of custom/proprietary data formats.

2.2. (U//FOUO) Encryption/Decryption/Validation

(TS//SI) All communication will be encrypted with the appropriate level of encryption for the information, transmission medium, and target system. RSA public keys are used to encrypt Session Announcement Infrastructure and Implant Headers; decryption is performed using RSA private keys. RC6 block ciphers are used to encrypt the Session Announcement Exfil Header and Metadata Block, and optionally Data Packets, using the RC6 Session Key and Message Indicators contained within the Session Announcement Implant Header.

(TS//SI) The FLASHHANDLE Mission Management (FMM) system is the database resource for generating/retaining crypto keys and other information (e.g. ImplantId's, TargetId's). All Session Announcements must have two 128-byte RSA-encrypted header blocks. The Infrastructure and Implant Header blocks are encrypted using the RSA-128 algorithm. Metadata blocks and (optionally) Data Packets are encrypted using the RC6-16 algorithm in Output Feedback Mode (OFB). See the Key Management diagram below.



- Figure 2.2-1: (U) Key Management

(TS//SI) Each CNE technique will contain the following deployed keys:

- Infrastructure Header Key (RSA public)
- Implant Header Key (RSA public)
- Shared Key (RC6); [called Exfil Key Modifier (RC6) in diagram above]

As shown above, the Infrastructure and Implant private keys are retained/protected within FLAXENPRECEPT (and are *never* deployed to the Implant!).

(TS//SI) The first RSA public key (the Infrastructure Key) is a global key that is common to many CNE techniques/targets and provides access to FLAXENPRECEPT; it is used to encrypt the Infrastructure Header. The Infrastructure private key is used to decrypt the Infrastructure Header, and a SHA-1 Hash of bytes 20-127 is computed and compared against bytes 0-19 to validate successful decryption.

(TS//SI) The second RSA public key (the Implant Key) is a unique key for the CNE technique to use on a specific target and will be used to support command and exfiltration functions; it is used to encrypt the Implant Header. The Implant private key is used to decrypt the Implant Header, and a SHA-1 Hash of bytes 20-127 is computed and compared against bytes 0-19 to validate successful decryption. The Implant Header contains CNE technique-specific information (i.e., Implant Operation), a Session Key (16-bytes) and Message Indicator (MI); for each session, a new Session Key (RC6) and MI are generated randomly by the implant.

(TS//SI) The Exfil Header is encrypted/decrypted using the Session Key (RC6) and MI from the Implant Header. Success is verified by comparing decrypted Exfil Header bytes 12-15 with decrypted Implant Header bytes 48-51.

(TS//SI) A Composite Exfil Key (RC6) is computed by combining the Session Key (RC6) with the Shared Key (RC6). First, populate a 32-byte array with the 16-byte Shared Key followed by the 16-byte Session Key. Next, perform a SHA1 hash of this byte array. The first 16 bytes of the resultant SHA1 hash is the Composite Exfil Key (RC6).

(TS//SI) The Metadata Block contains SHELLGREY tags and is encrypted/decrypted using the Composite Exfil Key (RC6) and MI+1, where (+1) represents adding one, represented as a big-endian 4-byte word, to the last four bytes of the MI from the Implant Header (bytes 88-91). The CRC-16 of the decrypted Metadata Block is computed and compared against decrypted Exfil header bytes 2-3 to validate successful decryption.

(TS//SI) The Data Packets are optionally RC6 encrypted or munged as specified by SHELLGREY tags as described in section 2.3.2.

2.3. (U//FOUO) Data Packets

2.3.1 (U//FOUO) Data Packet Description

(S//SI) Data packets are IP packets, which may or may not be a copy of an intercepted IP packet. These packets have a valid IP header, to allow the packet to be routed to the LP address. Other bytes within this packet may be altered to allow the packet to pass the target's egress filters, and/or satisfy a specific packet selection criterion. The SHELLGREY tags contained within the Metadata Block specify reconstruction and decryption commands that are processed in the order in which they occur, to transform the received Data Packet into the desired data format for further processing. The reconstruction result is therefore highly dependent on the specific reconstruction tags used and the order in which they occur. For example, as shown below, the received Data Packet always contains an IP header, but the SHELLGREY tag PACKET_IP4_REMOVE can be used to remove the IP4 header. Subsequent SHELLGREY commands could then operate on either the remaining "Packet buffer" layer, or on the Transport layer (if it exists).

Table 2.3.1-1 (U) Packet Layers

Packet data		
IP header (mandatory)	IP data (optional)	
	Transport header (optional)	Transport data (optional)

2.3.2 (U//FOUO) Data Packet Encryption

(C//SI) Packets will be encrypted, munged, or unencrypted. The encryption mode and the portion of the packet that is encrypted/munged are indicated in the Metadata Block using SHELLGREY tags. There are three possible encryption values for exfil type = Packet: None, Munge and RC6/Composite. None should be used only when **less** processing power is available than required to Munge the data. Munge should be used only when **less** processing power is available than required to perform full RC6/Composite mode encryption.

2.3.2.1 (U//FOUO) Encryption: NONE

(C//SI) Encryption type "NONE" is implied if none of the encryption related SHELLGREY tags described below are present within the Metadata Block.

2.3.2.2 (U//FOUO) Encryption: Munge

(C//SI) Munge is defined as a1 or 4 byte pattern XOR-ed with the data. When a 1 byte pattern is used the data is also XOR-ed with the positional index modulo 255. The munge pattern and the portion of the Packet Layer that is munged are passed during the Session Announcement in the meta-data block as one of the tags listed below. This pattern must be consistent throughout the session, and should not be changed in mid session. The SHELLGREY tags are:

Table 2.3.2.2-1 (S/SI) Munge SHELLGREY tags

Name	Tag	Values	Packet Layer
EXFIL_MUNGE_BYTE (deprecated: do not use)	3C	1-byte unsigned munge-byte	No operation
EXFIL_MUNGE_WORD (deprecated: do not use)	80, 14	2-byte unsigned munge-bytes[2]	
For all tags below, apply DEMUNGE to: specified *layer*, starting at *offset*, for *length* bytes (*length* = 0xFF = end of packet)			
PACKET _DEMUNGE8	86, 08	3-byte unsigned munge-byte, offset-byte, length-byte out[i+offset] = in[i+offset] xor munge-byte xor (i mod 255)	Packet data
PACKET _DEMUNGE8_IP4	86, 09		IP data
PACKET _DEMUNGE8_IP4_TRANSPORT	86, 11		Transport data
PACKET _DEMUNGE32	86, 0A	6-byte unsigned munge-bytes[4], offset-byte, length-byte out[i+offset] = in[i+offset] xor munge-bytes[i mod 4]	Packet data
PACKET _DEMUNGE32_IP4	86, 0B		IP data
PACKET _DEMUNGE32_IP4_TRANSPORT	86, 12		Transport data

2.3.2.3 (U//FOUO) Encryption: RC6/Composite

(C//SI) RC6/Composite is defined as RC6 encryption using Output Feedback Mode (OFB) mode, using the Composite Exfil Key (RC6), with a different MI for each data packet. For each data packet an independent packet MI is computed from the base MI sent in the Implant Header, modified by one or more bytes from the data packet IP header. This allows each data packet to be decrypted separately, regardless of previous packet losses. The MI modifier is specified as mimOffset and mimLength (in bytes) into the IP header.

(C//SI) The MI modifier defaults are (mimOffset=4, mimLength=4), which specify bytes 4-7 from the IP header::

- IP Identification Field (2 bytes)
- IP Flags, (3 bits)
- IP Fragment Offset (13 bits).

(C//SI) To compute the individual packet MI, the MI modifier bytes from the IP header are XOR-ed with the base MI from the Implant Header as follows:

- packetMI[0..15] = implantHeader.MI[0..15] = implantHeader[76..91]
- packetMI[i] ^= IPheader[i+mimOffset]; {i=0..mimLength-1}

(C//SI) The portion of the Packet Layer that is encrypted is passed during the Session Announcement in the meta-data block as one of the tags listed below. The SHELLGREY tags are:

Table 2.3.2.3-1 (S/SI) RC6/Composite SHELLGREY tags

Name	Tag	Value	Packet Layer
PACKET_DECRYPT_RC6 (deprecated: never implemented)	86, 0F	16-byte key, 16-byte MI	No Operation
PACKET_RC6_MI (deprecated: never implemented)	86, 10	16-byte MI	
PACKET_RC6_MI_MODIFIER	86, 16	2-byte unsigned mimOffset-byte, 1 < mimLength-byte < 16	IP header
For all tags below, apply RC6/Composite Decrypt to: specified *layer*, starting at *offset*, for *length* bytes (*length* = 0xFF = end of packet)			
PACKET_DECRYPT_RC6	86, 13	2-byte unsigned offset-byte, length-byte	Packet data
PACKET_DECRYPT_RC6_IP4	86, 14		IP data
PACKET_ DECRYPT_RC6_IP4_TRANSPORT	86, 15		Transport data

2.4. (U//FOUO) Transmission

(U//FOUO) Session Announcements can be transmitted in several ways. These are:

- Raw IP packet (TCP/UDP/ICMP), sent to predetermined LP and port.
- TCP connection
- Steganographlc Tunnel

(U//FOUO) Using a raw IP packet, a single IP packet would be constructed where the SA would be contained within the IP transport protocol's payload. The IP and transport protocol (TCP/UDP/ICMP) would be constructed to route the packet to a predetermined destination IP address, IP protocol, and for TCP/UDP the port number. This method provides no reliable transport. *Note: Currently, only UDP SA packets have been implemented*.

(U//FOUO) Using a TCP connection, the SA would be sent as stream data to a predetermined destination IP address and TCP port number. This method offers not only reliable network transport, by allows acknowledgement feedback from the LP, making this method reliable.

(U//FOUO) Using a steganographic tunnel the SA would hide within a higher-level network protocol. This method offers both improved OPSEC from detection, but also offers reliable transport by allowing acknowledgement feedback from the LP.

(C//SI) Transmission of data packets would be similar to raw IP packet by its nature, but only the destination IP address is predetermined, all TCP/UDP ports and ICMP packets are allowed.

(C//SI) Selection of method and parameters would be subject to the security posture imposed by a target environment.

2.5. (U//FOUO) Constraints and Timing Issues

(U//FOUO) The FASHIONCLEFT protocol is designed to operate over an unreliable network protocol where only communication from the sender is possible. Unless the Session Announcement could occur using a reliable network protocol, and even a bidirectional protocol, the data packets are clearly unreliable. To synchronize FASHIONCLEFT communications between the sender and receiver entities under all conditions, certain constraints and timing issues must be observed. Basically, the FASHIONCLEFT sender (implant) has an active role for maintaining proper synchronization, while the receiver (LP) has a passive role. What follows is a list of conditions; follow by a discussion of each.

- Sending of the Session Announcements
- New or Continuation of a terminated Session
- Termination of Session
- Sending Data Packets to Session Announcement ports
- Duplicate Data Packets
- Simultaneous Exfil of Multiple Sessions
- Early Termination

SENDING OF THE SESSION ANNOUCEMNETS

(U//FOUO) Generally the Session Announcement (SA) would be send prior to data packets be sent. If the SA data is sent using an unreliable network transport, it may be lost in transit. To overcome this, the SA data should be sent more than once. Since data packets may arrive prior to SA data, the receiver will buffer all received IP packets for **fifteen (15) minutes.** Once a valid SA arrives, the buffer will be scanned for possible data packets. To insure at least one SA arrives, more than three should be sent within the 15-minute time window, unless the SA is sent using a reliable protocol. In addition, for cases where the duration of exfiltration of data packets is short and the SA path is unreliable, the sender should also make at least three (3) attempts to send the SA.

TERMINATION OF SESSION

(U//FOUO) Once the session is established, the LP will terminate the session and stop looking for and forward data packets to the backend when the following two conditions are met:

1. No data packet has arrived for **ten (10) minutes**.
2. No SA has arrived for **ten (10) minutes.**

NEW OR CONTINUATION OF A TERMINATED SESSION

(U//FOUO) When an authenticated SA arrives, the packet selection criteria, as well as the FOGYNULL Identification fields will be used to determine if the packet is new or existing collection. If new the FASHIONCLEFT receptor will begin data packet collection. If it is redundant, the SA is ignored.

(U//FOUO) If the SA that arrives is in actuality for a session that has been terminated, the FASHIONCLEF receptor will be tasked to start data packet collection as if it was a new session. The inner portion of the CDR will rejoin these disjoint captures of the same session, by using the EXFIL_ID Meta-data tag and FOGYNULL identifiers, provided that no more than **fifteen (15) minutes** between termination of the last session and the start of the next session with the same EXFIL-ID from the implant.

(U//FOUO) To insure separate session are not inadvertently joined by the CDR, Implants should allow at least **sixty (60) minutes**, between the reuse of EXIL-ID values.

SENDING DATA PACKETS TO SESSION ANNOUNCEMENT PORTS

(U//FOUO) To minimize testing every packet, for session announcements, being sent to the FASHIONCLEFT receptor, only certain destination ports will be monitored for session Announcements. These ports can be arbitrarily configured to address the needs of specific deployed implants. If an implant chooses, or is required to by conditions imposed by the target environment, it may send data packets to these ports as well. Packets that match a data packet selection criterion are **not** checked for session announcements. Therefore, implants that desire to send announcements to the same port as data packets, must choose packet selection criteria carefully.

DUPLICATE DATA PACKETS

(U//FOUO) Implants that desire to send duplicate data packets are permitted, and may be desirable in some cases to insure that critical portions of the session are received by the FASHIOINCLEFT receptor. Duplicate data packets will occur as an artifact of packet sniffing by default. Data packets that are either duplicated by design or by coincidence generally will be removed by post processing during session reconstruction.

SIMULTANEOUS EXFIL OF MULTIPLE SESSIONS

(U///FOUO) Implants that desire to exfiltrate multiple disjoint sessions simultaneously, or even sessions back-to-back must utilize the EXFIL-ID in conjunction with the packet selection criteria. The EXFIL-ID is used to identify disjoint sessions, while a packet selection criterion is used to associate data packets to a particular session. Implants must choose both such that both are unique per session, and the above conditions are met.

EARLY TERMINATION

(U//FOUO) If an implant sends a session announcement that has a new EXFIL-ID but has a packet selection criterion that is identical to an existing criterion, the previous session will be terminated, and all new data packets that satisfy the criterion will be associated to the new EXFIL-ID.

3. (U//FOUO) Packet Selection Algorithms

3.1. (U//FOUO) Description

(S//SI) This section describes the methods used to associate meta-data with non-encapsulated egress packets that are transmitted independently from the data that is desired to be exfiltrated. Specifically, it addresses allowable contents for the "Packet-I/F-Selector" and "Packet-I/F-Data" fields of the FOGYNULL Infrastructure Header (INF-HDR). The following table identifies the specific values for the "Packet-I/F-Selector" field to select which algorithm to be used to do an association. This indicates whether the "Packet-I/F-Data" field is used to convey parameters for the selected algorithm. The "Packet-I/F-Data" field can be up to 32 bytes in length.

- Table 3.1-1. Packet Interface Algorithm Interpretation

Value	Algorithm	Contents of Data
0x00	None	Not used
0x01	Generic Pattern Matching	Matching rules
0x02	Fixed IP4 Field Matching	Matching rules

(U//FOUO) All methods other than the null case contain associated "Packet-I/F-Data". The number and format of the bytes comprising the "Packet-I/F-Data" vary based on the value of "Packet-I/F-Selector". However, to allow for cases in which a data receptor may not understand a particular "Packet-I/F-Selector" value, the "Packet-I/F-Data" field will always begin with a 1-byte field indicating the number of bytes in the "Packet-I/F-Data" field (excluding this 1-byte length field). So, for the non-null types specified above, the corresponding "Packet-I/F-Data" fields are as follows:

- Table 3.1-2. Packet-I/F Data

Length (1 byte)	Data (1-31 bytes)

(S//SI) Otherwise the contents of "Packet-I/F-Data" should be left as random data.

3.2. (U) Algorithms

3.2.1 (U) None, [0x00]

(S//SI) This represents the null case, i.e. the case in which there is no defined mapping association. Hence no packets will be collected; only the metadata is forwarded. The content of "Packet-I/F-Data" is unused.

3.2.2 (U) Generic Pattern Matching, [0x01]

(S//SI) This method implements a generic static pattern matching against specified bytes within the received packets. The content of "Packet-I/F-Data" contains a finite number of matching rules. These matching rules allow a pattern match against specified bytes following the start of an IP header; thus, they can include checks against bytes in the IP/TCP/UDP headers. (The rules could include checks against other bytes, but the header checks would probably be the norm.) Thus, this selector would allow checks against the IP identifier field as well as elements such as IP addresses, TCP window size, TCP urgent data size, ports, etc. This method is fairly general and should be able to cover any case in which portions of these headers contain fixed bytes.

The purpose of the data field is to specify the particular byte offsets and patterns to be used for matching against received packets. A non-zero number of "pattern blocks" must thus follow the length field (which specifies the number of bytes in the "pattern blocks"). Each "pattern block" consists of three bytes as described below. For this method, a received packet must match the criteria specified within ALL of the pattern blocks. Using a 1-byte length and 3 bytes/block implies there can be at most 10 "pattern blocks" within the 32-byte "Packet-I/F-Data" field. In addition, using a *7-bit* offset implies that the largest offset (from the start of the IP header) is *127* bytes. The format of this field is:

- <1-byte length> <pattern block 1>...<pattern block n>
- A pattern block has the following format:
  - <1-byte offset> - offset from start of IP header or *Transport Protocol header.*
    - *Bit 0: (LSbit) indicates start position, if 0 use start of IP header, if 1 use the start of Transport Protocol header.*
    - *Bit 1-7: offset value (multiply desired offset by 2 to shift into bits 1-7)*
  - <1-byte mask> - mask to be used with pattern
  - <1-byte pattern> - pattern to be matched

Thus, to perform a pattern block check, verify the following equality:

Packet [IP_Header_Offset + <offset>] & <mask> = <pattern> & <mask>

(The above assumes that enough packet bytes are available. For error checking purposes, the length field should always be a multiple of the pattern block size, i.e. 3, and a pattern byte bit-wised *and-ed* with the mask byte should always yield the pattern byte.)

As an example, suppose one wants to match any packet having an IP ID field of 0x1234. (The IP ID field is 2 bytes long, occurring at offsets 4 and 5 bytes from the start of the IP Header.) Then, the following data field would be used:

- `<1-byte length>: 6`
- `// Pattern block: 1`
- `<1-byte offset>: 8 = (2*4+0)`
- `<1-byte mask>: 0xFF`
- `<1-byte pattern>: 0x12`
- `// Pattern block 2`

- `<1-byte offset>:  10 = (2*5+0)`
- `<1-byte mask>:  0xFF`
- `<1-byte pattern>:  0x34`

3.2.3 (U) Fixed IP4 Field Matching, [0x02]

(S//SI) This method implements a static pattern matching against a specified subset of IP4/TCP/UDP header fields. This method is geared toward the use of the typical 5-tuple of fields used by IP and TCP (or UDP). In addition, it has the capability to handle both sides of the traffic, i.e. it allows the source/destination address and port fields to be toggled.

(S//SI) Note: The two-sided source/destination toggling described below should normally not be used. Normally only a single sided pattern should be specified. If toggling is used, there will be no way for the SHELLGREY reconstruction tags to correctly reconstruct the IP or Transport Header, since the same reconstruction tags would be applied for both directions (i.e. A → B and B → A would be received as A → X and B → X, but would reconstructed using the same value for X). The toggling mode can only be used if the IP and Transport headers will be dropped.

(S//SI) The contents of the data field are used to specify the subset of fields and associated patterns for matching against received packets. The rationale for having this method is the belief that it may often be the case that collection can be done based on the standard 5-tuple used for IP/TCP/UDP uniqueness. For this selector, the data field will always contain the following 24 bytes:

- <1-byte length> - number of bytes following this field (23)
- <1-byte mask> - indicates fields/toggles to be used
  - Bit 0: (<mask> & 0x01) if set, match on protocol field
  - Bit 1: (<mask> & 0x02) if set, match on destination IP address [Note: there is only one destination IP address, and this match is not useful when the destination IP address must be overwritten with a particular forwarding IP address]
  - Bit 2: (<mask> & 0x04) if set, match on source IP address field, use first source IP field if bit 5 of mask is unset; use both source IP fields if bit 5 of mask is set
  - Bit 3: (<mask> & 0x08) if set, match on destination port field(s) -- only use first destination port field if bit 6 of mask is unset; use both destination port fields if bit 6 of mask is set; [Note: if set, then there's an implied match of either UDP or TCP on IP protocol field.]
  - Bit 4: (<mask> & 0x10) if set, match on source port field(s) --only use first source port field if bit 7 of mask is unset; use both source port fields if bit 7 of mask is set [Note: if set, then there's an implied match of either UDP or TCP on IP protocol field.]
  - Bit 5: (<mask> & 0x20) if set, then allow source IP address to toggle between the two alternatives

- Bit 6: (<mask> & 0x40) if set, then allow destination ports to toggle between the two alternatives
- Bit 7: (<mask> & 0x80) if set, then allow source ports to toggle between the two alternatives
- <2-byte Protocol> - IP protocol field used if mask bit 0 is set
- <4-byte DstIP> - destination IP address used if mask bit 1 is set
- <4-byte SrcIP1> - source IP address #1 used if mask bit 2 is set
- <4-byte SrcIP2> - source IP address #2 used if mask bits 2 & 5 are set
- <2-byte DstPort1> - UDP/TCP destination port #1 used if mask bit 3 is set
- <2-byte DstPort2> - UDP/TCP destination port #2 used if mask bits 3 & 6 are set
- <2-byte SrcPort1> - UDP/TCP source port #1 used if mask bit 4 is set
- <2-byte SrcPort2> - UDP/TCP source port #2 used if mask bits 4 & 7 are set

(S//SI) While this method allows filtering on a particular 5-tuple (or subset), it's also intended to handle both sides of a TCP/UDP connection. Given that the destination IP address will typically be fixed (to be the forwarding IP address), the only fields that can toggle are the source IP address, destination and source ports. This method allows any or all of those fields to toggle. When toggling checks are being done, only the corresponding toggling fields should be used, e.g. SrcIP1, DstPort1, and SrcPort1 would be three values of the 5-tuple (if all are being used and all are allowed to toggle) and SrcIP2, DstPort2, and SrcPort2 would be the three toggled values of the 5-tuple (if all are being used and all are allowed to toggle).

3.2.3.1. (U//FOUO) Example, IP4 Field Matching

(S//SI) As an example, suppose a connection for the following 5-tuple is being collected:

```
TCP:  1.2.3.4/9 <-> 5.6.7.8/10

[TCP protocol, IP address 1.2.3.4 using port 9 and IP address
5.6.7.8 using port 10]
```

(S//SI) In addition, suppose the collection is to be forwarded to IP address A.B.C.D and all but the destination IP address of forwarded packets to remain unchanged. Then the data field would be as follows:

```
<1-byte length>:     23
<1-byte mask>:       0xFF
<2-byte Protocol>:   TCP
<4-byte DstIP>:      A.B.C.D
<4-byte SrcIP1>:     1.2.3.4
<4-byte SrcIP2>:     5.6.7.8
<2-byte DstPort1>:   10
<2-byte DstPort2>:   9
```

```
<2-byte SrcPort1>:   9
<2-byte SrcPort2>:   10
```

(S//SI) The above data field indicates a full match is required on all fields of the 5-tuple. It would match either of the following 5-tuples:

TCP: 1.2.3.4/9 -> A.B.C.D/10
(Protocol: SrcIP1/SrcPort1 -> DstIP/DstPort1)

TCP: 5.6.7.8/10 -> A.B.C.D/9
(Protocol: SrcIP2/SrcPort2 -> DstIP/DstPort2)

(S//SI) To indicate a match on just the IP addresses, the following data field could be used:

```
<1-byte length>:     23
<1-byte mask>:       0x26
<2-byte Protocol>:   <unused>
<4-byte DstIP>:      A.B.C.D
<4-byte SrcIP1>:     1.2.3.4
<4-byte SrcIP2>:     5.6.7.8
<2-byte DstPort1>:   <unused>
<2-byte DstPort2>:   <unused>
<2-byte SrcPort1>:   <unused>
<2-byte SrcPort2>:   <unused>
```

(S//SI) The above data field would match any IP packet having either of the following two IP pairs:

```
1.2.3.4 -> A.B.C.D
5.6.7.8 -> A.B.C.D
```

(S//SI) Finally, suppose that forwarded packets were to use a spoofed source IP address of E.F.G.H and a destination port of 7777, but the source port would be allowed to toggle. In that case, the following could be used:

```
<1-byte length>:     23
<1-byte mask>:       0x5F
<2-byte Protocol>:   TCP
<4-byte DstIP>:      A.B.C.D
<4-byte SrcIP1>:     E.F.G.H
<4-byte SrcIP2>:     <unused>
<2-byte DstPort1>:   7777
<2-byte DstPort2>:   <unused>
<2-byte SrcPort1>:   9
<2-byte SrcPort2>:   10
```

(S//SI) This would match either:

```
TCP:  E.F.G.H/9  -> A.B.C.D/7777
TCP:  E.F.G.H/10 -> A.B.C.D/7777
```

Appendix A Glossary

(U) Glossary of Terms Used

Glossary Term	Meaning	Source / Remarks
Exfiltrate	To extract [data] through a target's defenses.	
Implant	A software or hardware subcomponent that is surreptitiously emplaced in a target environment (CPU, router, etc) to pass selected information back to NSA, where it is processed for analysis.	
IP	Internet protocol	
	Note: All abbreviations / Acronyms are expanded where they are first used.	