DivertIt!

The TCP Connection Diverter

I)ruid <druid@caughq.org>

Brief Background

Diverters for network traffic or services have been in use for years:

- Proxy Servers (http, wingate, etc.)
- Session Redirects/Forwarders (ssh)
- Anonymizer[1]
- Onion Routing[2] (OR)
- Network Address Translation (NAT)

Diverting Today

Most current network diverter technologies:

- Are single-hops; They require a manual setup process to chain multiple diverters
- Are not designed to hide the connection origin
- Only provide diverting for a single service or port
- Employ crypto overhead to achieve resistance to traffic analysis or privacy

Onion Routing

- Uses specific pre-defined ports for traffic routing
- Requires either a proxy server or an invasive protocol stack modification
- Designed for one way communications (return communication may take an entirely different return path)
- Usually utilizes a public node network
- Provides a method for hidden services

How DivertIt! Is Similar to OR

- Resist traffic analysis by anonymizing the identity of the connection initiator
- Connections are application independent
- No centralized, trusted component

How DivertIt! is Different from OR

- DivertIt! does not provide hidden services
- Onion Routing (Tor) does not provide stealth features
- OR requires a software agent that can perform session data encryption. Protecting the session content is not a design goal, so we can use multiple diverter methods
- DivertIt! packets take the same route to and from the final destination
- DivertIt! uses randomly-chosen high ports (>1024) whereas OR has pre-defined ports

How DivertIt! is Different from OR

- QoS metrics are defined per diverter node

DivertIt!

TCP Connection Diverter-Chain Management Tool

What is DivertIt!?

- DivertIt! is a tool to manage TCP connection diverter-chains
- It makes it easy to connect to a remote host on a remote port via a chain of diverter hosts
- It abstracts the details of diverter chain setup from the user and provides a local port to connect to

DivertIt! Design Goals

- Anonymity - Mask the origin or the connection initiator from the target system
- Ease of Use - Little-to-no configuration or command-line options needed client-side
- Low Latency - quick, responsive connections relative to the number of diverters used
- Modular - provide a framework and control method independent of actual diverter types or methods

Architecture

- DivertIt! utilizes a client/server architecture to provide an abstracted client/server TCP connection
- The divertit client generates and sends a control message to the first agent
- Agents receive the control message from either the client or another agent, modify it, and pass it along to the next agent in the chain
- Agents act as both a client and a server to interact with each other

DivertIt! Client

- Interfaces with the user
- Generates a control message defining the diverter-chain to be setup
- Sends the control message to the first diverter agent in the chain
- Sets up a diverter on a local port to the first diverter in the chain

DivertIt! Agent

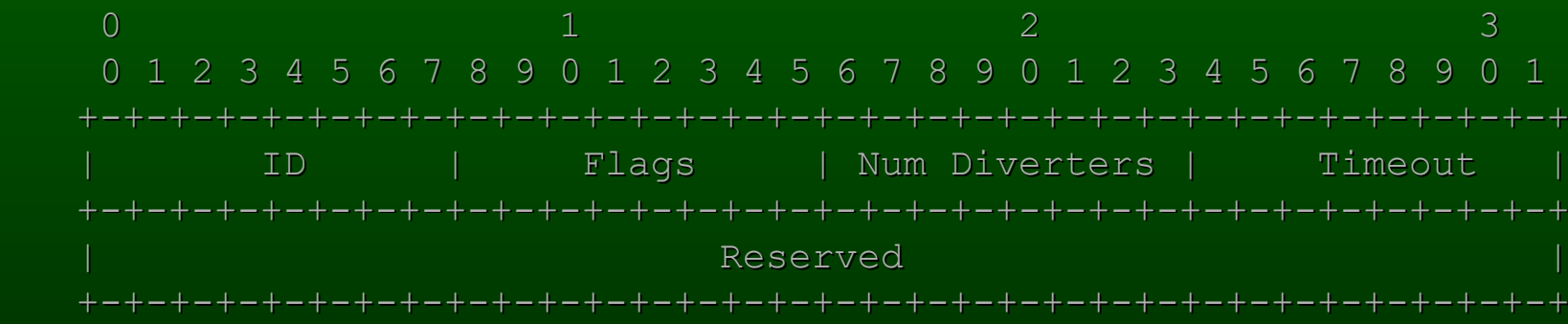
- Listens on network interfaces for a control message
- When it receives a control message, it sets up a diverter as defined by a diverter record in the control message
- Strips it's diverter record from control message
- Re-packages the control message and sends it to the next diverter in the chain, if any diverters remain to be messaged

DiVertor Agent Control Protocol

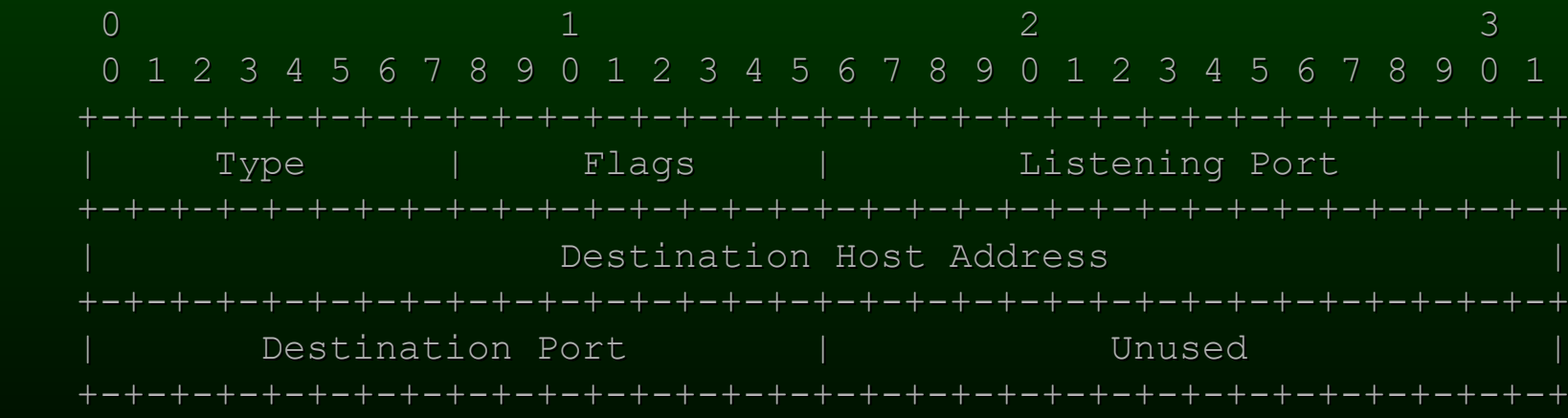
- DVACP provides a control information format for the DivertIt! Agent
- Consists of a header and an arbitrary number of diverter records

DVACP Packet Format

DVACP Header:



DVACP Diverter Record:



Header Fields

- ID - Session ID
- Flags - Session option toggles
- Number of Diverters - Number of diverters being chained together for this session
- Timeout - The amount of time diverters should listen for and divert new connections

Diverter Record Fields

- Type - The type of diverter that is being requested to be set up
- Flags - Diverter option toggles
- Listening Port - The TCP port the diverter should listen on for new connections
- Destination Address - The network address that the diverter should forward connections to
- Destination Port - The TCP port the diverter should forward connections to

Some Protocol Features

- Simple replay protection via a session ID cache in the Agent
- Protocol allows for an arbitrary number of diverters to be defined
- Unused/Reserved space in the current packet format for future extensibility

Carrier Packet

- Currently uses an ICMP type 8 (Echo) packet
- Padded to standard ICMP Echo payload length of 54 bytes if control message happens to be shorter
- Payload may or may not be obfuscated by an XOR against a SHA-1 hash of a shared secret (configurable)

Single Packet Authentication?

- Originally, NMRC's SPA protocol was to be used for delivery of the control message
- The SPA spec was not ready at the time that I began developing DivertIt! (I began writing an SPA implementation and couldn't finish)
- Got tired of waiting on the final spec so I rolled my own interim delivery method
- SPA provides additional features I want: encryption, authentication, multiple protocols
- I'll eventually replace what I used with SPA

Available Diverter Types

- Native Mode
- NetFilter
- stunnel

Native Mode

Native mode is a bi-directional byte by byte copy of data between two network sockets

- Works on all supported platforms
- Noisy; spawns two processes for each diverter (timer and diverter), then one process for each connection sent through the diverter

NetFilter Mode

NetFilter mode uses the Linux kernel to port-forward a connection using both source and destination NAT

- Quiet; No extra processes created
- Only works on Linux hosts with IP forwarding enabled.
- Adds rules to the system's iptables policy:
  - Adds rules for the actual NAT of the connection packets
  - Adds a permanent stateful accept rule for established/related packets (if it doesn't already exist)

stunnel Mode

Uses external stunnel tool to wrap a native mode connection

- Provides session data privacy
- Only works on hosts with stunnel available
- This mode will probably be replaced with a public-key enhancement to native mode

Which Diverter to Use?

- Probably all of them!
- Each type has it's own pro's and cons.
- It really depends on the platform capabilities of your available diverter hosts.

Future Diverter Types

- netcat | netcat
- LKM version of Native Mode
- BSD packet filter (behave like NetFilter)
- SSH Proxy
- Onion Router Interface?
- Proxy Server Interface?

Example of Operation

The user wants to connect to <target> on port <x>, from localhost on local port <y>, while having the connection pass through 3 diverters (default) in route:

divertit -p <y> <target> <x>

(divertit <target> <x>)

Command-line Options

Usage: divertit [options] targethost targetport

options:

-d # Number of diverters used (default: 3)

-p # Local diverter port # to use

-s Enable Stealth Features

-t # Time Window (in seconds, default: 30)

-v Increase verbosity (repeat for more)

-V Print version information and exit

-z gr33tz

Stealth Features

Activate stealth features with the -s command-line argument

- Turns on stealth flag in command packet header
- Client & Agents send indirect command packets via spoofed source addresses
- Enables basic process-hiding techniques

Future Features

- Diverter host capabilities preference options
- Diverter classes for selection:
  - Geography
  - Owner / Management Domain
- Required diverter identification for targeted RFC 1918 address space
- More diverter types
- Reverse connection mode
- More carrier packets for control message (UDP) with optional specific port to listen on

Support Tools

…making things even easier…

divcaps.sh

divcaps.sh is a script which will analyze a potential diverter host system and provide a profile of which capabilities that system supports

Capabilities include:

- Which diverter types can be used
- Address space owner for management domain/geographic classification

Tool Release

DivertIt! proof-of-concept code should be available via sourceforge.net shortly after this presentation in the forms of:

- Source package release
- CVS

Feedback Welcome!

- This is considered Alpha
- This is a work in progress
- Design / architecture is flexible and may change
- Ideas, criticism, code patches, etc. welcome

Q & A

References

- Anonymizer – http://www.anonymizer.org
- Onion Routing - http://www.onion-router.net