RFC SPIDER

SPIDER PROTOCOL

PROTOCOL SPECIFICATION

October 2017

prepared for

EPITECH Paris Graduate School of Digital Innovation

24 rue Pasteur

94270 Le Kremlin-Bicêtre

France

by

DEBRAY Victor – FAIFE Jeremy – CORDIER Nicolas

MENDIELA Paul – BLONDEAU Roman – PENAVAIRE Theo

EPITECH Paris – PROMO 2020

October 2017

SPIDER Protocol

TABLE OF CONTENTS

PREFACE ......................................................... 2

1. INTRODUCTION .................................................... 3

1.1 Global View .................................................. 3

2. FUNCTIONAL SPECIFICATION ........................................ 4

2.1 Header specification ......................................... 4

2.2 Header organization .......................................... 4

2.3 AData specification .......................................... 5

2.4 AData organization ........................................... 5

2.5 Commands Server/Client ....................................... 6

Informational ....................................................... 7

[Page 1]

PREFACE

This document describes the Spider Transmission Protocol.

The Spider Project is a remote keylogger operating with multiple servers and clients.

The connection is done with the Boost Asio Library and the data is transmitted after serialization and via OpenSSL to encrypt it.

The current version of the Spider is able to remotely steal keyboard inputs, mouse coordinates and the active window name.

This is the first version of the protocol.

[Page 2]

PROTOCOL SPECIFICATION

1. INTRODUCTION

This global view explains the way the data, which was stolen by the keylogger, is stored in order to be sent.

* 1. Global View

+-----------------------------------+

| <structure> HEADER |

+-----------------------------------+

| |

| - int magicNumber |

| - long long int dataSize |

| - enum datatype |

| |

+-----------------------------------+

|

+-----------------------------------+

| AData |

+-----------------------------------+

| |

| - std::string id |

| - int timeStamp |

| - std::string ipExtern |

| - std::string ipLocal |

| - Type\_e Type |

| |

+-----------------------------------+

+-----------------------------------+ +-----------------------------------+

| LogInput | | LogMouse |

+-----------------------------------+ +-----------------------------------+

| | | |

| - std::string windowName | | - std::string windowName |

| - std::string input | | - std::pair<int, int> windowSize |

| | | - std::pair<int, int> mouseCoord |

+-----------------------------------+ | |

+-----------------------------------+

+-----------------------------+

| Command |

+-----------------------------+

| |

| - std::string message |

| |

+-----------------------------+

+-----------------------+

| enum Type\_e |

+-----------------------+

| |

| - COMMANDS |

| - LOGINPUT |

| - LOGMOUSE |

| |

+-----------------------+

[Page 3]

PROTOCOL SPECIFICATION

1. FUNCTIONAL SPECIFICATION

This section shows how each packet containing all the elements are organized in memory.

* 1. Header specification

The header allows the server to verify the integrity of an incoming packet by indicating the size and the data type it is suppose to carry.

* 1. Header organization

+-----------------------------------+

| <structure> HEADER |

+-----------------------------------+

bytes

0 - - - - - - - 4 - - - - - - - 8 - - - - - - - 24

| | | |

| magicNumber | dataSize | dataType |

| | | |

+ - - - - - - - + - - - - - - - + - - - - - - - - |

4 4 12

[Page 4]

* 1. AData specification

An AData object is refered as a “Packet”. It contains all the information stolen by the client, and is sent to the server. The data is stored in a precise order because it is serialized before the sending and deserialized once it reaches the server.

The data is as follow :

* ID : The unique identifier of the client
* Timestamp : Time in seconds of when the input was done ( Format : time spent since 01/01/1970 )
* Extern IP : Extern IP of the client’s host
* Local IP : Local IP of the client’s host
* Window Name : Name or location of the active window where the input was done
* Input
* Mouse Coordinates : Coordinates in X ( horizontal axis ) and Y ( vertical axis ) on click
* Type : Type of the data contained in the packet : Keyboard, Mouse or Command.
  1. AData organization

+-----------------------------------+

| AData |

+-----------------------------------+

bytes

+ - - - - -+ - - - - - + - - - - - - - - + - - - - - - - - + - - - - - +

| | | | | |

| id | timeStamp | ipExtern | ipLocal | Type |

| | | | | |

+ - - - - -+ - - - - - + - - - - - - - - + - - - - - - - - + - - - - - +

sizeof(id) 4 sizeof(ipExtern) sizeof(ipLocal) 12

bytes bytes

+ - - - - - - - - + - - - - - - - + + - - - - - - - +

| | | | |

| windowName | Input | | Message |

| | | | |

+ - - - - - - - - + - - - - - - - + + - - - - - - - +

sizeof(windowName) sizeof(Input) sizeof(Message)

bytes

+ - - - - - - - - + - - - - - - - - - + - - - - - - - +

| | | |

| windowName | windowSize | mouseCoord |

| | | |

+ - - - - - - - - + - - - - - - - - - + - - - - - - - +

sizeof(windowName) 8 8

[Page 5]

2.5 Commands Server/Client

In order to cummunicate, both client and server must possess several commands to execute precise routines.

COMMANDS DESCRIPTION

START Start the client Program

STOP Stop the client Program

PAUSE Interrupt communication between client and server.

Data is still written in cache on client side.

LOGS DESCRIPTION

<CLIENT\_ID>START\_OK Client has started

<CLIENT\_ID>START\_KO Client failed to start

<CLIENT\_ID>STOP\_OK Client has stopped

<CLIENT\_ID>STOP\_KO Client failed to stopped

<CLIENT\_ID>PAUSE\_OK Packet sending has stopped but data is still

written on cache

<CLIENT\_ID>PAUSE\_KO Command Pause failed

[Page 6]

INSTRUCTIONNAL

Full Copyright Statement

Copyright (C) BLONDEAU Roman (2017). All Rights Reserved.

[Page 7]