Project Report: Demonstration of a Secure Credential Transfer Protocol

Daniel Tweed

School of Information Studies
McGill University
Montreal, Canada

February 2017

2017/02/28

Chapter 1

Introduction

1.1 Background and Motivation

From day-to-day transactions for shopping, banking, and investing, to employment-related activities such as distributed online project management, e-commuting or electronic repositories, increasingly our daily lives are being conducted online. As a result, the issue of securing personal information and credentials such as login names and passwords, credit card numbers, and banking details, as well as corporate and institutional secrets and intellectual property, is receiving increased attention towards enhancing security and reducing online risk. Many major websites and employers now employ two (or more) factor authentication, with the most secure requiring additional out-of-band authentication, including completely offline methods such as postal delivery or face-to-face exchanges of secure keys. Unfortunately, while these mechanisms can help deal with many of the issues arising from interception and theft of credentials in transit, they ignore the persistent threat posed by infected end-points, i.e. resident malware infecting user PCs and servers on which credentials are entered and validated.

1.2 Secure Credential Transfer Protocol

If we assume that all end-points are infected with malware, noting that this is the safest assumption, then secure credentials transfer is only possible if the PC itself is taken out of the equation. In order to facilitate secure credential transfer, without resorting to completely offline solutions, [1] proposes a novel compartmentalized computer architecture which would isolate the hardware used for credential transfer from the main PC. A basic configuration for such an architecture is

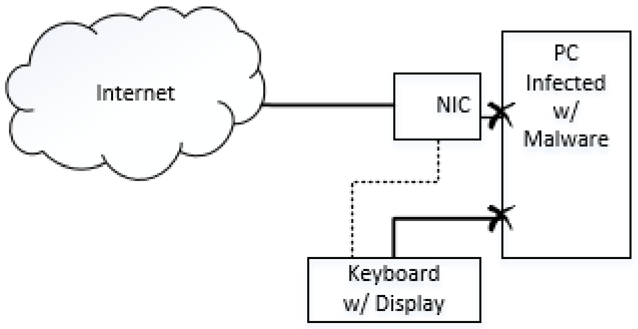

Fig. 1.1 Basic system architecture for SecCTP

depicted in Fig. 1.1. The hardware connections used for normal PC operations are shown with a solid line, and are disconnected during a secure transaction in favour of the dashed line connections which cut out the infected PC from the transaction. The malware infecting the PC may be “aware“ of the transaction occurring but has no access to the confidential details.

To support the compartmentalized architecture and credential exchange, the Secure Credential Transfer Protocol (SecCTP) was developed and a draft specification is detailed in [2]. SecCTP is a text-based client-server protocol to provide end-to-end credential protection. Based on the HTTP, SecCTP uses a Hello, Request, and Response messages to initiate, specify, and complete SecCTP transactions. SecCTP employs datagram transport layer security (DTLS) for confidentiality of transmitted messages and two-way authentication via X.509 certificates. Additionally, SecCTP details the following requirements for a SecCTP enabled network interface card (SecNIC):

1. Has its own processor and memory
2. Has input/output capabilities
3. Capable of acquiring networks configuration parameters from designated network devices
4. Is solely programmable by keyboard connected via a secure circuit

The SecNIC is solely responsible for sending, receiving, and processing secure credential transactions, and the four requirements translate into effective isolation from credential interception by malware infected user devices.

Chapter 2

SecCTP Demonstration System

2.1 Demonstration Overview

In order to demonstrate the capabilities of the SecCTP protocol, a simplified system which implements the behaviour of a SecCTP capable NIC (SecNIC) and server has been designed. The demonstration system is implemented using two Raspberry Pi (R-Pi) single-board computers, acting as SecNIC and SecCTP server, with a connected user PC. The demonstration system and core traffic flows for a SecCTP transaction are shown in Fig. 2.1. For the demonstration, the R-Pi

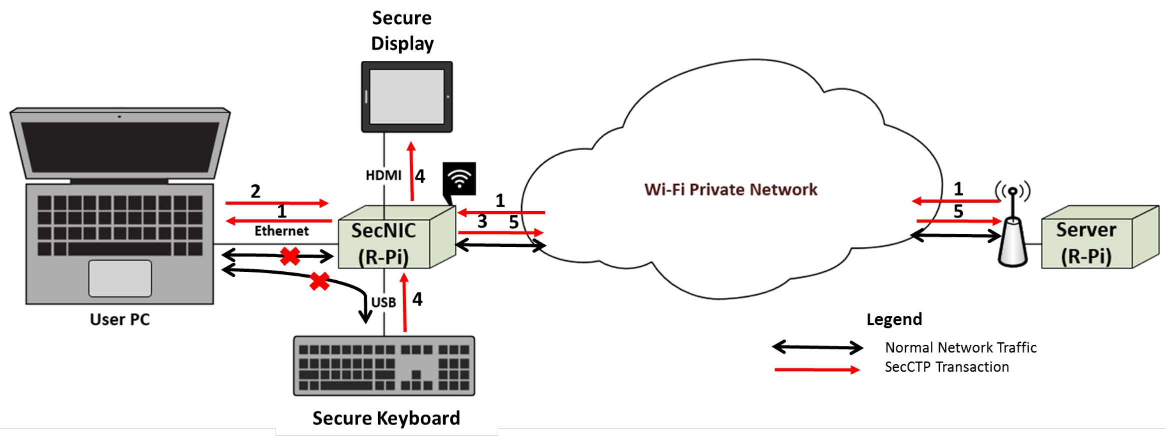

Fig. 2.1 Demo set-up and traffic flows

hosting the SecCTP server also acts as access point for a local Wi-Fi network and web server for the demonstration pages. The SecNIC R-Pi has a connected display and keyboard for User I/O. Custom software applications have been developed to run on the User PC, SecNIC, and SecCTP server to implement the SecCTP client-server protocol, associated facilitating functions such as connection tracking and monitoring, and a basic web server.

Under normal operation, i.e. not requiring SecCTP transactions, the SecNIC acts as network interface for the User PC and simply passes traffic between the user and the outside network. When a SecCTP transaction is required for log in, based on web page requests received, the server transmits an HTTP 303 redirect message back to the same page (labelled (1) in Fig. 2.1). The monitoring application detects the HTTP 303 message and signals the SecCTP client running on the SecNIC to initiate a secure credential transfer (2). For payment type transactions, the web server signals the SecCTP server that authorization is required and the server initiates the SecCTP transaction. In either case, the web server queues the related request until it is notified of the outcome of the SecCTP authorization. As outlined in the draft specification [2], a series of messages are exchanged between the SecCTP client and server to initiate the SecCTP transaction, initialize the DTLS transaction, validate the server, and authenticate the server using X.509 certificates (3). Once the request has been validated, the SecNIC displays the request details to the user on the secure display and prompts the user for the required input, login credentials in the each case of the demonstration (4). The SecCTP transaction is completed when the server authenticates the user (5) and completes any necessary transaction acceptance steps. Finally, the web server is notified of the outcome and forwards any queued pages or an appropriate message in the case of failed authentication.

2.2 Component Details

The demonstration system is comprised of 3 primary hardware components, as shown in Fig. 2.1, and 5 custom C/C++ applications. The applications were developed under Linux and make use of several system and open-source libraries for implementing the required functionality.

The SecCTP server is implemented using a R-Pi running Arch Linux ARM and acts as both SecCTP/web server and access point for a private Wi-Fi network. Required Wi-Fi private network functions are managed by *hostapd* for wireless access point functions and *dnsmaq* for DHCP and DNS functions [3, 4] . A basic web server was written using the *libmicrohttpd* library to provide basic request processing, page delivery, and post processing functionality to test the demonstration SecCTP client and server [5]. The core SecCTP server is a custom out-of-band authentication application based on DTLS provided by the *gnutls* library and adhering to the SecCTP draft specification as closely as possible, within the restrictions imposed by the project

[2, 6]. Inter-process communication (IPC) is facilitated by POSIX message queues and system signals.

The SecNIC is implemented using a R-Pi with on-board display and attached keyboard. The SecNIC acts as both front-facing network interface for the connected User PC and as SecCTP client for secure transactions. In order to ensure that only SecCTP transaction requests to or from servers with which the User PC has active ongoing connections are honoured, a monitoring application was written using the *libtins* library which tracks current connections between the User PC and the outside network [7]. The core SecCTP DTLS client was developed in tandem with the SecCTP server, again based on the *gnutls* library [6]. IPC is achieved in the same way as in the server, with the addition of communication with the User PC monitoring application. The monitoring application inspects client and server HTTP messages and signals the SecNIC if the HTTP 303 trigger is received. Since the monitoring application runs on a separate computer and is connected via Ethernet, basic TCP client/server functions are added to the monitoring application and SecCTP client application.

Chapter 3

Instructions for SecCTP Demonstration

The demonstration system is comprised of 3 primary hardware components and associated accessories and cables. The SecCTP server is the R-Pi in the black covered case. The R-Pi acting as SecNIC has an open case to facilitate attaching the 5" LCD screen on the GPIO header and included HDMI connector. This R-Pi will also require a connected keyboard and Ethernet cable to connect to the User PC.

Once assembly is complete, the following steps are necessary to run a demonstrations:

1. Power on the SecCTP server by plugging in the USB power supply
   – The software access point, SecCTP server and web server will automatically run at startup
2. Set up the Ethernet connection on the user PC
   - Static IP on the range `192.168.0.200~255`
   - Subnet mask: `255.255.255.0`
   - Default gateway: `192.168.0.100`
   - Preferred DNS: `192.168.0.100`
3. Connect the Ethernet cable between the SecNIC and User PC
4. Power on the SecNIC and logon (username:*daniel* password:*secctp*)
5. Type the following command at the prompt: `./run_demo`
   – This will start the applications for tracking active connections, monitoring for SecCTP trigger messages, and the SecCTP client application and user interface

6. On the User PC, open Firefox (or Chrome) and enter `secnic.demo:8888` in the address bar
7. The Demonstration Login page will be displayed
8. Clicking the `Login` button will trigger a SecCTP transaction and you will be prompted for credentials on the SecNIC secure display
9. Login using the credentials username: `secctp` password: `pass`
   – Up to 3 attempts can be made before the authentication process fails and the web server will return the “Unauthorized“ page
10. Once valid credentials have been entered, the SecCTP demo Secure Payments page will be returned by the web server
11. On the payments page, enter a payment amount and click the `Submit` button
12. Again, this will trigger a SecCTP transaction and prompt for credentials on the secure display – repeat step (9)
13. Once valid credentials have been entered, the “Success“ page will be returned by the web server
14. Repeat as desired

Chapter 4

Physical Keyboard Disconnect

As a future enhancement, it is desirable to have a single keyboard support the system and be either physically or logically disconnected during a SecCTP transaction. From the perspective of a SecNIC implementation in production, the SecNIC would provide the necessary USB keyboard connection and no physical keyboard would be attached to the potentially infected User PC. In this case, custom drivers would be responsible for directing USB traffic from the keyboard either to the User PC under normal operation or to the SecCTP client during a SecCTP transaction.

To emulate this behaviour in the demonstration, the following options have been considered:

1. Existing keyboard sharing software
   – To the best of my knowledge, no free and open-source software (FOSS) for Windows are currently available. For example, the Synergy application is available for various distributions of Linux (github.com/symless/synergy), but is now a paid product for Windows (symless.com/synergy/)
   – Modifications to the demo to use this option include: Launching and disabling the software from the SecNIC client application; Running the client application under a desktop environment (such as LXDE) rather than from the consolve; and running a suitable Linux distribution on the User PC (not as a guest OS)

2. Modified switched USB hub or Custom relay-based USB switching board
   – I have not been able to source a reasonable software controlled USB switch; however, there inexpensive options which offer switching of a USB device between two computers, e.g. [8]. It is likely that such a hub could be modified from a mechanical switch to relay

control. Also, there are many available relays, including USB controlled relay boards for systems such as Raspberry-Pi, but they are generally designed for home automation applications and may not be suitable for USB
– Modifications to the demo include: Implementing the required software control of the USB switch in the SecNIC client application; If USB control is not an option, GPIO is possible but the secure display would be in the way and a suitable housing and connector would have to be sourced. At present, the display connects to the GPIO header for convenience with the GPIO pins being used for touchscreen functionality which has not been implemented for the demo. Power being supplied by the GPIO header can be moved to the included micro-USB input on the display.

Option (1) would potentially be the simplest, assuming a suitable OS for the User PC is installed and Synergy is configured to force focus to the User PC when it is connected. Code implementing these functions had been written into the client application but later discarded when it was no longer feasible (it exists in previous commits on the project GitHub repository).

Option (2) requires some additional electronics design for either the modified hub or custom switch. In the case of the custom switch, the electrical profile of USB must be considered and the design must switch the four connections in the correct sequence (i.e. power then data) but the timing constraints for USB enumeration (plug-and-play) are quite strict. It was outside the scope of the project to research or design this further but the design should be quite feasible in either case, and both have likely been done before.

References

[1] J. McAlear, "Compartmentalization Architectures for Protection of Internet Credentials and Transactions from CPU Resident Malware," in *World Congress on Internet Security (World-CIS*, 2012, pp. 60–65.

[2] C. Davis, M. Maheswaran, and J. McAlear, "Secure Credential Transfer Protocol (SecCTP)," Draft IETF Specification, January 2015, unpublised.

[3] J. Maline, "IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator," Website. [Online]. Available: w1.fi/hostapd

[4] S. Kelley, "Dnsmasq," Website. [Online]. Available: www.thekelleys.org.uk/dnsmasq/doc.html

[5] C. Grothoff, "GNU Libmicrohttpd," Website. [Online]. Available: www.gnu.org/software/libmicrohttpd/

[6] "The GnuTLS Transport Layer Security Library," Website. [Online]. Available: www.gnutls.org

[7] M. Fontanini, "libtins: Package Crafting and Sniffing Library," Website. [Online]. Available: libtins.github.io

[8] "Plugable usb 2.0 switch for one-button swapping of usb device/hub between two computers (a/b switch)." [Online]. Available: http://www.amazon.ca/Plugable-One-Button-Swapping-Between-Computers/dp/B006Z0Q2SI