

NASA/TM-2014-218555



Implementation and Validation of a Two-Tier Light-Weight Method for Securing Embedded Controllers
Securing the Raspberry Pi as a Development Platform

Ethan G. Ganzy
The Universities Space Research Association
Johnson Space Center, Houston, Texas

September 2014

NASA STI Program ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- **TECHNICAL PUBLICATION.** Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.
- **TECHNICAL MEMORANDUM.** Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.
- **CONTRACTOR REPORT.** Scientific and technical findings by NASA-sponsored contractors and grantees.

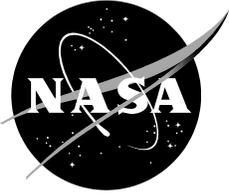
- **CONFERENCE PUBLICATION.** Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.
- **SPECIAL PUBLICATION.** Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- **TECHNICAL TRANSLATION.** English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at <http://www.sti.nasa.gov>
- E-mail your question to help@sti.nasa.gov
- Fax your question to the NASA STI Information Desk at 443-757-5803
- Phone the NASA STI Information Desk at 443-757-5802
- Write to:
STI Information Desk
NASA Center for AeroSpace Information
7115 Standard Drive
Hanover, MD 21076-1320

NASA/TM-2014-218555



Implementation and Validation of a Two-Tier Light-Weight Method for Securing Embedded Controllers
Securing the Raspberry Pi as a Development Platform

Ethan G. Ganzy
The Universities Space Research Association
Johnson Space Center, Houston, Texas

September 2014

Acknowledgments

I would like to thank those at NASA Johnson Space Center, and especially EV8, a Branch of the AVIONIC SYSTEMS DIVISION (EV) responsible for assuring system level performance integrity and ensuring interface compatibility of: Communications and tracking; Command and data handling; Instrumentation systems' and Provides RF spectrum management analysis. In particular, my mentor Chatwin Lansdowne who chose me for this experience and allowed me to undertake this research endeavor.

Available from:

NASA Center for AeroSpace Information
7121 Standard Drive
Hanover, MD 21076-1320

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161

This report is also available in electronic form at <http://ston.jsc.nasa.gov/collections/TRS>

Abstract

Protecting information and equipment at NASA is an area of increasing concern, after a GAO report in February (see also 2009), and an Inspector General release in March. Supervisory, Control and Data Acquisition (SCADA) systems are especially vulnerable because these systems have lacked standards, use embedded controllers with little computational power and informal software, are connected to physical processes, have few operators, and are increasingly also being connected to corporate networks.

The opportunity exists with IEEE 1877 to “build in” durable, scalable, effective security features. As expressed in the CIO’s 2011 strategic plan, Goal 2, the standard interface needs standard security features that can support a variety of standards-based or proprietary architectures and be flexible enough to enable a response if some of these standard features are compromised, while supporting rather than interfering with an automated operations where one or a few operator(s) control many networked modules in a secure way. Similar themes are expressed in the ISA technical report TR-99.00.02.

An approach was developed during the Summer of 2013 which remained to be validated in one of the test beds. The goal was to implement and evaluate approaches to both server-side security and client-side security, and tools for interacting with the interface such as browser plug-ins. The approaches developed may be refined as a result of this work.

1. INTRODUCTION

The National Aeronautics and Space Administration (NASA) faced with increasing budget cuts has sought shared development utilizing existing Commercial Off-The-Shelf (COTS) opportunities. An area that was explored for such cost considerations was the test orchestration and automation efforts at Johnson Space Center, Houston, Texas. The Integrated Power and Avionics System (iPAS), is one such test bed that is utilized as an evaluation environment for hardware and software geared towards specific missions. The environment uses a common interface framework, the mREST software developed by METECS to collect data from tests, simulations, hardware and monitoring applications. The software is a derivative of the existing Representational State Transfer (REST), a common application programming interface (API) on the internet.

REST is an architectural style that uses identification of resources; manipulation of resources through representations; self-descriptive messages; and, hypermedia as the engine of application state, to build distributed systems that are scalable and resilient to change. REST is utilized for designing networked applications. The idea is that, rather than using complex mechanisms such as CORBA, RPC or SOAP to connect between machines, simple HTTP is used to make calls between machines.

The standard, currently in the working group phase, will be published in the proposed IEEE Standard P1877. The definitions how the proposed system will work and the interaction amongst the devices will be specified in P1877, Test Orchestration Interface. The security section of the working standard is merely a suggestion at this point and further development is needed.

Due to constraints of the Government furlough and limited resources, the initial project of implementation and validation of the mREST code was restructured to address security concerns centered on the Raspberry Pi as a Logical System Element (LSE). The primary concern of the validation efforts was the review of existing suggestions regarding security and ease of management in the LSE. Security standards were evaluated and tested in the lab setup which consisted of several Raspberry Pi's behind NASA's external firewall and protected with an individual router. A variety of encryption algorithms were explored and tested to document system performance on the LSE. Since the device would be accessed in the future from an external source, security was addressed keeping in mind simplicity of operation and key management.

2. SYSTEM COMPONENTS AND SETUP

A. *The Raspberry Pi*

The Raspberry Pi Model B was utilized as a demonstrator LSE. Central to the device is a Broadcom BCM2835 system-on-chip processor running at 700MHz, with a VideoCore IV GPU running at 250MHz. A single 256MB module of Hynix LPDDR memory running at 400MHz provides RAM for both the CPU and GPU, with the typical split leaving around 186MB of memory available for the user.

The Raspberry Pi always needs to boot off of an SD card loaded with an operating system (OS) disk image. There are many OS versions offered for the Raspberry Pi, however, for this execution, Raspbian “Wheezy” was chosen as the OS. Configuration of a Raspberry Pi as a webserver is a relatively straightforward process with a routine establishment of a LAMP environment. It is termed a LAMP server which is one of the most common configuration for webserver which stands for:

- Linux – operating system
- Apache – webserver (http) software
- Mysql – database server
- PHP or Perl – programming languages

Configuration is done at the command line. This provides the ability and advantage of remotely managing and installing the server. It also means that the computer can spend less processor time drawing a GUI.

Since Raspbian is Debian based, you need to install the following packages:

- php-xml-parser
- php5-common
- php5-cli
- php5-dev
- php5-mysqldb
- php5-pgsql
- php5-sqlite
- apache2
- libapache2-mod-dnssd

Before installing these packages, make sure that your definitions are up to date:

Become root and run the following:

```
root prompt#> apt-get update
```

Then:

```
root prompt#> apt-get install php-xml-parser php5-common php5-cli php5-dev php5-  
mysqlnd php5-pgsql php5-sqlite apache2 libapache2-mod-dnssd
```

The install is interactive and will most likely pick up some other dependent packages.

With dependent packages installed, a copy of mREST was placed on the Raspberry Pi. For purposes of this implementation and validation, the TFDM was not available. Installation instructions to copy the mrest-3.1.2.tgz archive somewhere on the Pi are included for future expansion of the Raspberry Pi as an LSE.

The mREST package is copied to a unique directory on the Pi. From that directory, run the following command (you may or may not need to be root depending on where you are in the file system):

```
root prompt#> zcat mrest-3.1.2.tgz | tar xvf -
```

This will create the mrest-3.1.2 directory. Contained within this directory are limited example configuration files for apache and php. Since Rasbian is a Debian based distribution, selection of the Ubuntu samples are warranted.

For Apache:

```
<path to mrest-3.1.2 directory>/mrm/config/system/samples/ubuntu/apache2.conf  
Copy this file to /etc/apache2  
Document the location of DocumentRoot locations for the Virtual hosts.
```

For PHP:

```
<path to mrest-3.1.2 directory>/mrm/config/system/samples/php.ini  
Copy this file to /etc/php5/cli/php.ini  
Document the library path in this file. You should be able to run a php script from the  
command line. If the php script is unable to run because of an error locating a library, check this  
file and adjust the path accordingly (or install the missing library if it is not installed).
```

To manually start apache, issue the following command as root:

```
root prompt#> /etc/init.d/apache2 start
```

To have apache as the web server start at boot, issue the following command as root:

```
root prompt#> update-rc.d apache2 defaults
```

B. Test Flow and Data Manager

The Test Flow and Data Manager (TFDM) is the primary system manager that was envisioned to be used in testing. The system currently runs on Fedora Linux and is designed to provide overall coordination and control of the requests sent to individual LSE. These requests and responses are handled via a standard web browser. Functioning as a webserver, it will allow

any authorized computer to connect to the TFDM to monitor or initiate tests in cooperation with the LSE devices.

C. Desktop System Manager

For the purpose of testing a standard desktop PC was utilized as a system manager. Connections to the Raspberry Pi were established through the Command Line Interface utilizing SSH as well as through the web browser when accessing the Raspberry Pi as a web server.

3. INITIAL CONSIDERATIONS

Confidentiality, Integrity, and Availability (CIA) is a model designed to guide policies for information security within an organization. In this context, confidentiality is a set of rules that limits access to information, integrity is the assurance that the information is trustworthy and accurate, and availability is a guarantee of ready access to the information by authorized people. Judgments and considerations regarding ease of use, key length and system management were all taken into account in the final recommendations.

A. Requirements

Focusing on the security of the Raspberry Pi, there were several requirements highlighted in the initial design for the LSEs.¹ Those directly applicable to the pi environment were:

- Must be able to accommodate LSEs with limited hardware resources.
- The system must be fairly automated. It shall be designed in a way which is conducive for batch or shell scripts to handle the majority of the setup with as little input required from the test conductor as possible.
- The implementation cannot require the test conductor to memorize more than one or two sets of credentials. Anything more than this and he or she is likely to write them down, causing another security risk.
- Most importantly, the security standards set forth in IEEE P1877 must comply with all federal regulations set forth by the National Institute of Standards and Technology (NIST), Federal Information Processing Standards (FIPS), and any other regulation at the federal level that dictates how a network must be secured.

4. ENCRYPTION

A. *Symmetric-key*

Symmetric key cryptography is also known as shared key cryptography. As the name suggests, it involves 2 people using the same private key to both encrypt and decrypt information. Public key cryptography, on the other hand, is where 2 different keys are used – a public key for encryption and a private key for decryption.

Because symmetric key cryptography uses the same key for both decryption and encryption, it is much faster than public key cryptography, is easier to implement, and generally requires less processing power. The current specification for this recommended by the federal government is Advanced Encryption Standard (AES) with options for 128, 192 or 256 bit keys. The use of a 256 bit key is approved for the transmission of any data at the top secret classification level or below.²

For IEEE P1877, 192 bit was chosen. Implementations in the test bed were demonstrated using each of the bit keys with minimal notice of decreased speed. While 256 bit may take longer due to a larger key, testing shows that the Raspberry PI, 0.8 MB/s (6.4 Mbit/s) demonstrates good performance as an AES-256-CBC throughput. It is important to note this exceeds some DD-WRT compatible routers.

A disadvantage of symmetric key cryptography is that the 2 parties sending messages to each other must agree to use the same private key before they start transmitting secure information. This may be impossible depending on the circumstances – because the 2 parties who want to communicate with each other through a secure means may be on different sides of the world. And this means that they will need a secure way to tell each other what the private key will be – if there were a secure way to do this, then the cryptography would not have been necessary in the first place in order to create that secure channel.

B. *Public-key*

Public-key cryptographic systems use two keys -- a public key known to everyone and a private or secret key known only to the recipient of the message. When Bob wants to send a secure message to Alice, he uses Alice's public key to encrypt the message. Alice then uses her private key to decrypt it.

An important element to the public key system is that the public and private keys are related in such a way that only the public key can be used to encrypt messages and only the corresponding private key can be used to decrypt them. Moreover, it is virtually impossible to deduce the private key if you know the public key. The use of this form of encryption, while sophisticated in approach, is manageable for the test bed implantation through the generation of self-signed certificates. For the LSEs public key to be valid in actual production, the key must be signed by a Certificate Authority (CA).

Signed Certificates are a common function of the Internet. They are used for all HTTP Secure (HTTPS) connections between a client and third party. The use of CAs in an environment such as this would require a private CA. The benefits of this form of security implementation:

- LSEs communicate without the messages being routed through the server.
- There are a greater number of keys in this implementation used for communication.
- If one key is compromised, the entire network has a lower level of risk.

Auto-signed SSL certificates are easily generated and you will have exactly the same security and encrypting level than any official certificate. However, this certificate will not be officially recognized over the Internet. When connecting to your site, your Web browser will warn you about the impossibility to guaranty your security connecting to this site, and you have to accept this. For the test bed implementation, this was within acceptable requirements.

How to generate an auto signed certificate:

Install OpenSSL:

```
root prompt#>sudo apt-get install openssl
```

Generate your self signed certificate:

```
root prompt#>sudo mkdir -p /etc/ssl/localcerts
root prompt#>openssl req -new -x509 -days 3650 -nodes -
out/etc/ssl/localcerts/autosigned.crt -keyout/etc/ssl/localcerts/autosigned.key
root prompt#>chmod 600 /etc/ssl/localcerts/*
```

SSL is an acronym that stands for Secure Sockets Layer. It is the standard behind secure communication on the Internet, integrating data cryptography into the protocol. The data is encrypted before it even leaves your computer, and is decrypted only once it reaches its intended destination. In theory, if the encrypted data were intercepted or eavesdropped before reaching its destination, there is no hope of cracking that data. But as computers become ever faster as each year passes, and new advances in cryptanalysis are made, the chance of cracking the cryptography protocols used in SSL is starting to increase.

SSL and secure connections can be used for any kind of protocol on the Internet, whether it be HTTP, POP3, or FTP. SSL can also be used to secure Telnet sessions. While any connection can be secured using SSL, it is not necessary to use SSL on every kind of connection. It should be used if the connection will carry sensitive information.

OpenSSL is capable of message digests, encryption and decryption of files, digital certificates, digital signatures, and random numbers. The advantage of OpenSSL is that it is more than just the API, it is also a command-line tool. The command-line tool can do the same things as the API, but goes a step further, allowing the ability to test SSL servers and clients.

5. CONCLUSIONS

The initial goals of the project were to:

- Elevating privileges and becoming familiar with the mREST project source code.
- Attempt to port existing PHP code over to an embedded device and SoC.
- Add HTTPS and AES security on both sides and evaluate for inclusion with the LES.
- Evaluate performance improvement and ease of use.
- Evaluate the requirements document to determine any refinements.

Due to hardware and software failures on the TFD, and SME access from METECS personnel, the scope of the project was gradually switched to an investigation of using the Raspberry Pi as a model for the LSE and evaluating security options. In consideration, the government furlough along with pre and post-furlough events, posed additional difficulties and stretched available resources.

The Raspberry Pi was utilized as a development platform for Offline/Online development using inexpensively replicated non-ACES platforms. The Raspberry Pi is a low cost credit-card sized computer, with an ARM-based CPU. It uses very little power (only 3 Watt), so it's ideal for a server that's always-on.

The results of the project demonstrated that establishing OpenSSL and the OpenVPN server is very easy due to the fact that the Raspberry Pi is running (a modified) Debian Weezy called Raspbian. Utilizing both OpenVPN and OpenSSL does slow performance since the CPU will have to encrypt/decrypt on the fly establishing one additional "encryption layer." This layered approach may lose OpenSSL efficiency. However, testing shows that the Raspberry PI, 0.8 MB/s (6.4 Mbit/s) demonstrates good performance as an AES-256-CBC throughput. It is important to note this exceeds some DD-WRT compatible routers.

Future projects may investigate the benefits of coding the PHP to establish a secure connection as envisioned in the original intent of the project. However, at this time the key management issues would be labor intensive with no promise of increasing overall security design.

References

¹Behe, Robert, "A Standardized Approach for Securing Automated Test Orchestration Interfaces", USRA Summer Internship Report, Houston, TX. 2013

²Security Requirements for Cryptographic Modules, FIPS PUB 140-2

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188		
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.					
1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE September 2014	3. REPORT TYPE AND DATES COVERED Technical Memorandum			
4. TITLE AND SUBTITLE Implementation and Validation of a Two-Tier Light-Weight Method for Securing Embedded Controllers Securing the Raspberry Pi as a Development Platform			5. FUNDING NUMBERS		
6. AUTHOR(S) Ethan G. Ganzy					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Lyndon B. Johnson Space Center Houston, Texas 77058			8. PERFORMING ORGANIZATION REPORT NUMBERS S-1168		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001			10. SPONSORING/MONITORING AGENCY REPORT NUMBER TM-2014-218555		
11. SUPPLEMENTARY NOTES					
12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited Available from the NASA Center for AeroSpace Information (CASI) 7115 Standard Hanover, MD 21076-1320 Category: 61			12b. DISTRIBUTION CODE		
13. ABSTRACT (Maximum 200 words) Protecting information and equipment at NASA is an area of increasing concern, after a GAO report in February (see also 2009), and an Inspector General release in March. Supervisory, Control and Data Acquisition systems are especially vulnerable because these systems have lacked standards, use embedded controllers with little computational power and informal software, are connected to physical processes, have few operators, and are increasingly also being connected to corporate networks. The opportunity exists with IEEE 1877 to "build in" durable, scalable, effective security features. The standard interface needs standard security features that can support a variety of standards-based or proprietary architectures and be flexible enough to enable a response if some of these standard features are compromised, while supporting rather than interfering with an automated operations where one or a few operator(s) control many networked modules in a secure way. An approach was developed during the Summer of 2013 which remained to be validated in one of the test beds. The goal was to implement and evaluate approaches to both server-side security and client-side security, and tools for interacting with the interface such as browser plug-ins. The approaches developed may be refined as a result of this work.					
14. SUBJECT TERMS security; Raspberry Pi; automation; IEEE P1877; OpenVPN; computer software; computer hardware; computer networks; algorithms			15. NUMBER OF PAGES 16	16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT Unlimited		
