Application Note: Configuring and Enabling Management Security
Application Note: Configuring and Enabling Management Security

Chapter 1
Introduction

Chapter 2
Enabling Management Security

Certificate Generation ................................................................. 2-1
Configuring Secure Shell .............................................................. 2-2
Disabling Insecure Access ........................................................... 2-3
Configuring Secure Socket Layer ............................................... 2-3
Preventing Insecure Web Sessions .............................................. 2-4

Appendix A
Certificate Generation Scripts

SSH .................................................................................................. A-1
SSL ............................................................................................... A-1
SSL Helper Files ........................................................................... A-3
Chapter 1
Introduction

In the past, network communications were simply a matter of packaging frames of information and shipping them over the wire to their destination. Protocols gave little thought to who might be viewing the frames as they crossed the wire, or what illegitimate parties might do with the information so gleaned. More and more, security has become an ever-present concern amongst the members of the networking community. Networking infrastructure is far too important to risk abuse by hackers, whether they are malevolent or simply mischievous. As a whole, the community has turned to encryption as a means of ensuring the security of network transactions.

Interactive login is a mainstay for providing a means to control and/or configure an entity across the network. For decades the telnet protocol has provided this capability for devices wishing to provide interactive login over a network. However, these protocols are chief culprits with regard to the transmission of sensitive information (e.g. passwords) over the network unprotected. The current de facto standard for providing interactive login in a secure fashion is the Secure Shell (SSH). SSH provides a number of services in a secure manner. These include port forwarding, file transfer, X11 forwarding, and interactive login. Of these, currently only interactive login is of interest for the NETGEAR managed switch software.

Managing devices with a web browser has been standard practice for several years. Unfortunately, standard HTTP transactions are no more secure than telnet. This was one of the original barriers to the success of "e-commerce". The solution (then and now) is the use of the Secure Sockets Layer (SSL) protocol. SSL provides a means of abstracting an encrypted connection between two stations. Once established, such a connection is virtually no different to use than an unsecured connection. This allows an established protocol (e.g. HTTP) to operate in a secure manner on an open network.

A third component of management on a modern networking appliance is SNMP. The SNMP protocol has it own security mechanisms outside of SSH and SSL. Consequently discussion of security for SNMP transactions is outside the scope of this document.
Enabling management security is a two-step process. The first step involves generating and loading appropriate authentication keys (SSH) and security certificates (SSL). Optionally a reputable third party such as RSA Security, Inc. or Entrust, Inc. can validate these certificates and keys but for evaluation purposes validation is unnecessary.

The second step involves enabling either SSL or SSH and optionally disabling the insecure versions of telnet and web management. Once enabled, subsequent management connections may be made in a secure manner.

Certificate Generation

To generate self-signed credentials, the open source applications ssh-keygen and openssl can be used to create the seven files used to form the security certificates and authentication keys. Both of these applications are well documented by the open source community. Detailed descriptions will not be repeated here as the user can check the man pages for detailed help. Two scripts are included in the appendix at the end of this application note along with some helper files. This set of files can be freely modified and used to generate the appropriate self-signed credentials. Generation of these credentials has been verified using both cygwin and Linux.

Once the component files are created, the credentials must be loaded onto the switch running the NETGEAR managed switch. This is accomplished using the copy command from a tftp server.

1. From privileged EXEC mode, issue the following command:

   ```
   #copy tftp://192.168.77.122/rsa1.key nvram:sshkey-rsa1
   ```

   where the IP address of the tftp server should be substituted as appropriate.
2. Repeat the **copy** command for all the other authentication components:

   - rsa2.key nvram:sshkey-rsa2
   - dsa.key nvram:sshkey-dsa
   - dh512.pem nvram:sslpem-dhweak
   - dh1024.pem nvram:sslpem-dhstrong
   - server.pem nvram:sslpem-server
   - rootcert.pem nvram:sslpem-root

The SSL and SSH credentials may be uploaded separately as needed. However, it is likely that if security is required for one access method it will be required for all access methods. Therefore, best practice is to create the certificates and authentication key simultaneously.

## Configuring Secure Shell

Once the authentication credentials are loaded and the certificates and authentication keys are formed, management security may be configured on the NETGEAR managed switch device.

1. From privileged EXEC mode, issue the command

   ```
   # ip ssh
   ```

   as described in the Command Line Interface Reference manual. This will allow secure shell sessions to be instantiated on the NETGEAR managed switch.

2. Check the message log to determine the success or failure of the command.

Messages indicating successful start of the ssh service look like this:

   ```
   0 days 00:17:07 Unit: 1 : File: sshd_main.c : Line: 349 :
   SSHD: Done generating server key
   0 days 00:17:06 Unit: 1 : File: sshd_main.c : Line: 639 :
   SSHD: successfully loaded RSA2 key
   0 days 00:17:06 Unit: 1 : File: sshd_main.c : Line: 627 :
   SSHD: successfully opened file ssh_host_rsa_key
   0 days 00:17:06 Unit: 1 : File: sshd_main.c : Line: 605 :
   SSHD: successfully loaded DSA key
   ```
If a secure connection cannot be established, entries such as the following indicate the nature of the problem.

```
0 days 00:17:06 Unit: 1 : File: sshd_main.c : Line: 592 : SSHD: successfully opened file ssh_host_dsa_key
0 days 00:17:06 Unit: 1 : File: sshd_control.c : Line: 400 : SSHD: sshdListenTask started
```

In this case, the authentication credentials were invalid and should be regenerated.

### Disabling Insecure Access

To disable insecure access to the NETGEAR managed switch, issue the following command:

```
# no telnet
```

**Note:** Caution should be exercised before issuing this command as once the active telnet sessions are terminated, no new telnet sessions will be allowed. Please consult the appropriate Command Reference for more information on configuring remote sessions.

### Configuring Secure Socket Layer

 Optionally or in concert with SSH, SSL may be enabled. Once again the message log is the best source of feedback for problem determination.

1. To enable SSL, issue the privileged EXEC mode command:

   ```
   # ip http secure-server
   ```

2. Attempt a secure web access using https. Consult the message log for to determine the success or failure status.
Valid certificates are indicated by a message log entry that looks like the following:

```
0 days 01:25:29 Unit: 1 : File: sslt_util.c : Line: 303 :
SSLT: Successfully loaded all required SSL PEM files
```

Certificate information may be accessed using browser-specific methods. With Internet Explorer, the lock icon along the bottom message line can be checked for certificate details. Additionally, when connecting to a NETGEAR managed switch that uses self-generated credentials, Explorer will warn the user about the authenticity of the certificate. When secure certificates are acquired from a third party this warning will no longer occur.

**Preventing Insecure Web Sessions**

Insecure web sessions may be prevented by disabling the http server using the privileged EXEC mode command:

```
# no ip http server
```

As with secure shell, the best guide for information on NETGEAR managed switch commands that control HTTP and HTTPS access is the Command Line Interface Reference manual.
Appendix A
Certificate Generation Scripts

The following two scripts and two helper files can be used to generate self-signed certificates and authentication keys.

**SSH**

**sshKeygen.sh**

```bash
#!/bin/sh

# Generate key files for rsa and dsa
#
# RSA V1
/usr/bin/ssh-keygen -q -t rsa1 -f rsa1.key -C '' -N ''
# RSA V2
/usr/bin/ssh-keygen -q -t rsa -f rsa2.key -C '' -N ''
# DSA for V2
/usr/bin/ssh-keygen -q -t dsa -f dsa.key -C '' -N ''
```

**SSL**

Exercise care when using cut and paste to generate this file as formatting can wrap some command lines. For example:

```
/usr/bin/openssl req -newkey rsa:1024 -sha1 -keyout rootkey.pem -out rootreq.pem -config root.cnf -passout pass:NETGEAR
```

should appear on a single line in the shell script.
pemCreate.sh

#!/bin/sh

# Ensure that OpenSSL is installed and set the location correctly
OPENSSL=/usr/bin/openssl

# Set the password to something unique
PASSWORD=NETGEAR

# Set the number of days the certificates will be valid for
VALID_NUM_DAYS=3650

# Generate the Self Signed Trusted Root Certification Authority (CA) and Private Key
${OPENSSL} req -newkey rsa:1024 -sha1 -keyout rootkey.pem -out rootreq.pem -config root.cnf -passout pass:${PASSWORD}

${OPENSSL} x509 -req -days ${VALID_NUM_DAYS} -in rootreq.pem -sha1 -extfile root.cnf -extensions certificate_extensions -signkey rootkey.pem -out rootcert.pem -passin pass:${PASSWORD}

cat rootcert.pem rootkey.pem > root.pem
rm rootkey.pem rootreq.pem

# Generate the Trusted Server Certificate signed by the Root CA
${OPENSSL} req -newkey rsa:1024 -sha1 -keyout serverkey.pem -nodes -out serverreq.pem -config server.cnf -reqexts req_extensions -passout pass:${PASSWORD}

${OPENSSL} x509 -req -days ${VALID_NUM_DAYS} -in serverreq.pem -sha1 -extfile server.cnf -extensions certificate_extensions -CA root.pem -CAkey root.pem -CAcreateserial -out servercert.pem -passin pass:${PASSWORD}

cat servercert.pem serverkey.pem rootcert.pem > server.pem
rm serverkey.pem servercert.pem serverreq.pem

# Generate the Diffie-Hellman weak and strong parameters
${OPENSSL} dhparam -check -text -5 512 -out dh512.pem
${OPENSSL} dhparam -check -text -5 1024 -out dh1024.pem
SSL Helper Files

**root.cnf**

```
# default settings for example.
[ ca ]
default_ca = exampleca

[ exampleca ]
dir = /opt/exampleca
certificate = $dir/cacert.pem
database = $dir/index.txt
new_certs_dir = $dir/certs
private_key = $dir/private/cakey.pem
serial = $dir/serial

default_crl_days = 7
default_days = 365
default_md = sha1

policy = exampleca_policy
x509_extensions = certificate_extensions

[ exampleca_policy ]
commonName = supplied
stateOrProvinceName = supplied
countryName = supplied
emailAddress = supplied
organizationName = supplied
organizationalUnitName = supplied

[ req ]
default_bits = 2048
default_keyfile = privkey.pem
default_md = sha1
prompt = no
distinguished_name = req_distinguished_name
x509_extensions = req_extensions

# the following sections are specific to the request being built

[ certificate_extensions ]
basicConstraints = CA:true
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer:always

[ req_distinguished_name ]
countryName = US
stateOrProvinceName = California
localityName = Santa Clara
organizationName = NETGEAR, Inc.
```
organizationalUnitName = Support
commonName = NETGEAR Root CA
emailAddress = support@netgear.com

[ req_extensions ]
basicConstraints = CA:true

server.cnf

# default settings for example.
[ ca ]
default_ca = exampleca

[ exampleca ]
dir = /opt/exampleca
certificate = $dir/cacert.pem
database = $dir/index.txt
new_certs_dir = $dir/certs
private_key = $dir/private/cakey.pem
serial = $dir/serial
default_crl_days = 7
default_days = 365
default_md = shal

policy = exampleca_policy
x509_extensions = certificate_extensions

[ exampleca_policy ]
countryName = supplied
stateOrProvinceName = supplied
localityName = supplied
organizationName = supplied
organizationalUnitName = supplied
commonName = supplied
emailAddress = supplied

[ req ]
derault_bits = 2048
default_keyfile = privkey.pem
default_md = shal
prompt = no
distinguished_name = req_distinguished_name
x509_extensions = req_extensions

# the following sections are specific to the request being built

[ certificate_extensions ]
basicConstraints = CA:false
subjectAltName = DNS:localhost
[ req_distinguished_name ]
countryName            = US
stateOrProvinceName    = California
localityName           = Santa Clara
organizationName       = NETGEAR, Inc.
organizationalUnitName = Support
commonName             = localhost
electricityAddress    = support@netgear.com

[ req_extensions ]
basicConstraints = CA:true
subjectAltName        = DNS:localhost