

# SSLSmart–Smart SSL Cipher Enumeration

By Gursev Singh Kalra

Managing Consultant

McAfee® Foundstone® Professional Services

## Table of Contents

Introduction	3
SSLSmart Features	3
SSLSmart Installation	6
SSLSmart GUI	8
SSLSmart Sample Use Cases	12
Using SSLSmart APIs	13
Conclusion	16
About The Author	16

### Introduction

While performing web application penetration testing, PCI compliance audits and network assessments, correctly verifying secure sockets layer (SSL) versions and cipher suites supported by the web server is an important test case. A number of tools allow users to test for supported SSL ciphers suites. Most of the existing tools, however, provide testers with a fixed set of cipher suites. Further, testing itself is performed by initiating an SSL socket connection with one cipher suite at a time. This approach is inefficient, fraught with false positives, and often does not provide a clear picture of the true vulnerability of the server.

SSLSmart is designed to combat these shortcomings. SSLSmart is a highly flexible and interactive tool aimed at improving efficiency and reducing the false positives during SSL testing.

### SSLSmart Features

SSLSmart offers a wide range of features to improve testing efficiency and reduce false positives.

#### Content/CONNECT scans

During the initial SSL handshake, the web server and the browser negotiate the cipher suite to be used for the communication channel. If both the browser and the server support a common cipher suite, a communication channel is established for data transfer. If not, the end result is one of the following two scenarios:

1. The web server terminates the connection during initial handshake and no data is transferred.
2. The web server completes the handshake and transfers minimum data in the form of an error page and terminates the connection. Any future attempts with these unsupported cipher suites encounter the same result.

Since a complete SSL connection is established, and data is transferred in the second scenario above, most of the SSL testing tools (including commercial scanners) falsely conclude that the cipher suite is supported. It is therefore important for the testers to be able to see the actual server response for each cipher suite. SSLSmart offers two different types of scans to investigate and weed out false positives owing to this behavior.

1. *Content scan (default)*—Exact server response can be seen in HTML and text forms for each cipher suite selected for the test URL. Figures 1, 2, and 3 below show various server error messages received for weak cipher suites from live systems.
2. *CONNECT scan*—Focuses only on the success or failure of SSL socket connection with various cipher suites. This behavior does not offer any advantage over existing SSL testing tools and is thus likely to have similar issues with false positives. However, this scan is faster and consumes fewer network and CPU resources.

Test?	SSL Version	Cipher Name	Key Length (bits)	Supported?	Response Code
1	<input checked="" type="checkbox"/>	SSLv2	DES-CBC-MD5, EXP-RC2-CBC-MD5, EXP-...	Yes	HTTP/1.1 500 Internal Server Error
2	<input checked="" type="checkbox"/>	SSLv3	ADH-DES-CBC-SHA, EDH-RSA-DES-CBC-...	Yes	HTTP/1.1 500 Internal Server Error
3	<input checked="" type="checkbox"/>	TLSv1	ADH-DES-CBC-SHA, EDH-RSA-DES-CBC-...	Yes	HTTP/1.1 500 Internal Server Error

HTML | Text

**SSL Protocol Alert**

The SSL protocol version that your browser uses is SSLv2 and it is not compatible with the server settings.

Please try the following:

- Check the SSL protocol settings on your browser for SSLv3/TLSv1 protocol support and enable the same.

Figure 1. This image shows error message with SSLv2 support disabled on web server.

Test?	SSL Version	Cipher Name	Key Length (bits)	Supported?	Response Code
1	<input checked="" type="checkbox"/>	SSLv2	DES-CBC-MD5, EXP-RC2-CBC-MD5, EXP-...	Yes	HTTP/1.1 500 Internal Server Error
2	<input checked="" type="checkbox"/>	SSLv3	ADH-DES-CBC-SHA, EDH-RSA-DES-CBC-...	Yes	HTTP/1.1 500 Internal Server Error
3	<input checked="" type="checkbox"/>	TLSv1	ADH-DES-CBC-SHA, EDH-RSA-DES-CBC-...	Yes	HTTP/1.1 500 Internal Server Error

HTML | Text

**SSL Alert**

The browser and the web-site cannot communicate securely because there are no common encryption algorithms.

Please try the following:

- Check the SSL protocol settings on the browser for SSLv3/TLSv1 protocol support.
- The secure web-site may be using high-strength encryption algorithms(128 bit). Check the SSL settings on your browser, if it supports high-strength encryption algorithms.
- Upgrade your browser to latest version.

Figure 2. Image shows error page when only strong cipher suites are supported by remote server.

Test?	SSL Version	Cipher Name	Key Length (bits)	Supported?	Response Code
1	<input checked="" type="checkbox"/> SSLv2	DES-CBC-MD5, EXP-RC2-CBC-MD5, EXP-I --		Yes	HTTP/1.1 500 Internal Server Error
2	<input checked="" type="checkbox"/> SSLv3	ADH-DES-CBC-SHA, EDH-RSA-DES-CBC- --		Yes	HTTP/1.1 500 Internal Server Error
3	<input checked="" type="checkbox"/> TLSv1	ADH-DES-CBC-SHA, EDH-RSA-DES-CBC- --		Yes	HTTP/1.1 500 Internal Server Error

HTML | Text

**SSL Alert**  
The browser and the web-site cannot communicate securely because there are no common encryption algorithms.  
Please try the following:  

- Check the SSL protocol settings on the browser for SSLv3/TLSv1 protocol support.
- The secure web-site may be using high-strength encryption algorithms(128 bit). Check the SSL settings on your browser, if it supports high-strength encryption algorithms.
- Upgrade your browser to latest version.

Figure 3. Image shows server response for DES-CBC-SHA (56 bits) cipher suite.

### Dynamic cipher suite support

Most SSL testing tools provide a fixed set of cipher suites. SSLSmart hooks into Ruby<sup>1</sup> OpenSSL<sup>2</sup> bindings and offers dynamic “on the fly” cipher suite generation capabilities. Various options are listed below:

1. Cipher suite grid can be created as per requirements using OpenSSL filters.<sup>3</sup>
2. Individual cipher suites can be selected or deselected. This allows fine-grained control over cipher suites to test.
3. SSL cipher suites can be grouped as per SSL version to perform quick SSL version and cipher group checks for higher efficiency.

### Certificate verification

SSLSmart performs server certificate verification. It uses the Firefox Root CA Certificate<sup>4</sup> repository to perform Root CA verification. Additional Root CA Certificates can be added to the rootcerts.pem file or a custom .pem file can be supplied for Root CA Certificate verification.

### Proxy support

SSLSmart provides web proxy support. For results to be accurate, it is important to use a transparent proxy.<sup>5</sup>

### Reporting

Reports can be generated in XML, HTML, and text formats along with their verbose versions. Verbose report versions include complete application response for each cipher suite and full details of the server certificate.

Application programming interfaces (APIs) Monkey patched Ruby APIs that form the backbone of SSLSmart can be consumed to write custom Ruby scripts for quick tests. These APIs can be consumed by users who work with the SSLSmart gem.

1. <http://ruby-lang.org/>

2. <http://openssl.org/>

3. <http://www.openssl.org/docs/apps/ciphers.html>

4. <http://curl.haxx.se/ca/cacert.pem>

5. [http://en.wikipedia.org/wiki/Proxy\\_server](http://en.wikipedia.org/wiki/Proxy_server)

### Platform support and installers

SSLSmart has been tested to work on the following platforms and versions of Ruby:

1. *Microsoft Windows*—Ruby 1.8.6 with wxruby<sup>6</sup> (2.0.0) and builder<sup>7</sup> (2.1.2).
2. *Linux*—Ruby 1.8.7/1.9.1 with wxruby (2.0.0) and builder (2.1.2).
3. Installer is available in two formats:
  - » *Windows installer*—Windows installer is created using ruby 1.8.6 code base. Ruby installation is not required to get this working
  - » *SSLSmart gem*— SSLSmart gem includes source code and can be consumed by users who have Ruby installed on their systems

### SSLSmart Installation

The SSLSmart .zip file containing the Microsoft Windows installer and gem can be downloaded from <http://www.mcafee.com/us/downloads/free-tools/index.aspx>. Installation details for Windows setup and SSLSmart gem are provided below.

### Installing with SSLSmart Windows setup

After extracting the contents from the .zip file, go to the directory where the .zip file is extracted and run SSLSmart1.0.exe. This will begin the installation process. The installer is self-explanatory. Figure 4 below show the first installation screen on Windows.

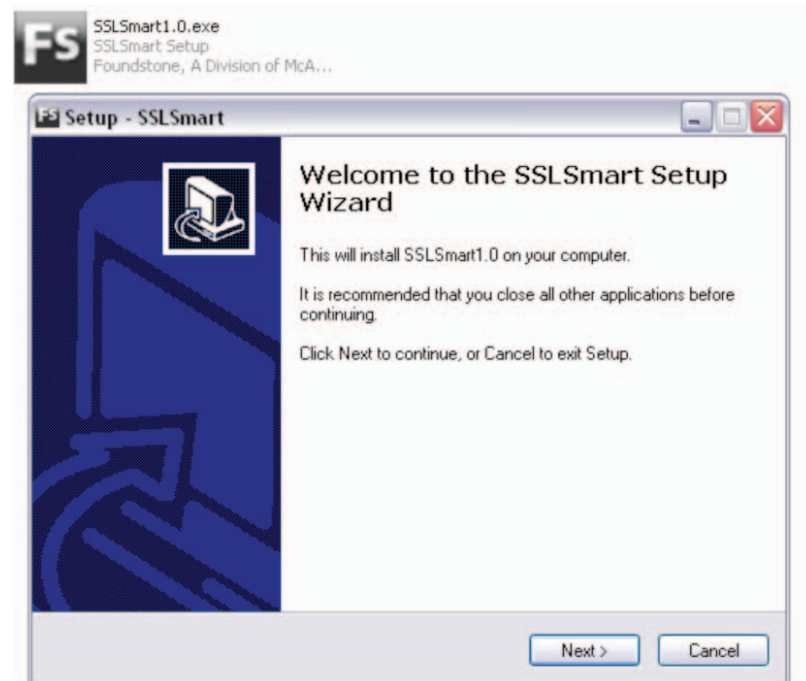


Figure 4. SSLSmart setup.

### Installing SSLSmart gem

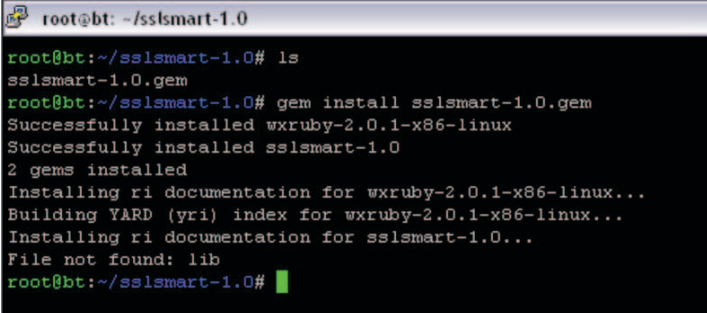
A working Ruby installation is required to install the SSLSmart gem and execute the extracted code. From the command shell, go to the directory where the contents of the SSLSmart zip file are extracted. Inside the directory, execute the following command:

```
gem install sslsmart-1.0.gem
```

The above-mentioned command will install SSLSmart gem along with its dependencies (wxruby and builder).

Sample installation paths for Linux and windows are shown below:

- Linux: /usr/lib/ruby/gems/1.8/gems/sslsmart-1.0/
- Windows: C:\ruby\lib\ruby\gems\1.8\gems\sslsmart-1.0\

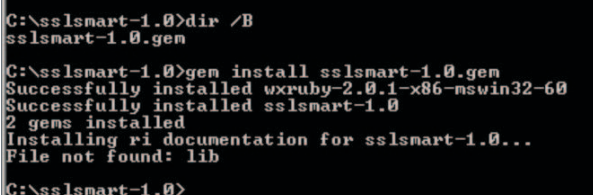


```

root@bt: ~/sslsmart-1.0
root@bt:~/sslsmart-1.0# ls
sslsmart-1.0.gem
root@bt:~/sslsmart-1.0# gem install sslsmart-1.0.gem
Successfully installed wxruby-2.0.1-x86-linux
Successfully installed sslsmart-1.0
2 gems installed
Installing ri documentation for wxruby-2.0.1-x86-linux...
Building YARD (yri) index for wxruby-2.0.1-x86-linux...
Installing ri documentation for sslsmart-1.0...
File not found: lib
root@bt:~/sslsmart-1.0#

```

Figure 5. Image shows SSLSmart installation on Linux (BackTrack 4).



```

C:\sslsmart-1.0>dir /B
sslsmart-1.0.gem
C:\sslsmart-1.0>gem install sslsmart-1.0.gem
Successfully installed wxruby-2.0.1-x86-mswin32-60
Successfully installed sslsmart-1.0
2 gems installed
Installing ri documentation for sslsmart-1.0...
File not found: lib
C:\sslsmart-1.0>

```

Figure 6. SSLSmart gem installation on Windows.

Once the gem is installed, go to the bin directory inside the installation path to find the SSLSmart source code. Run the following command to start SSLSmart GUI.

```
ruby sslsmartgui.rb
```

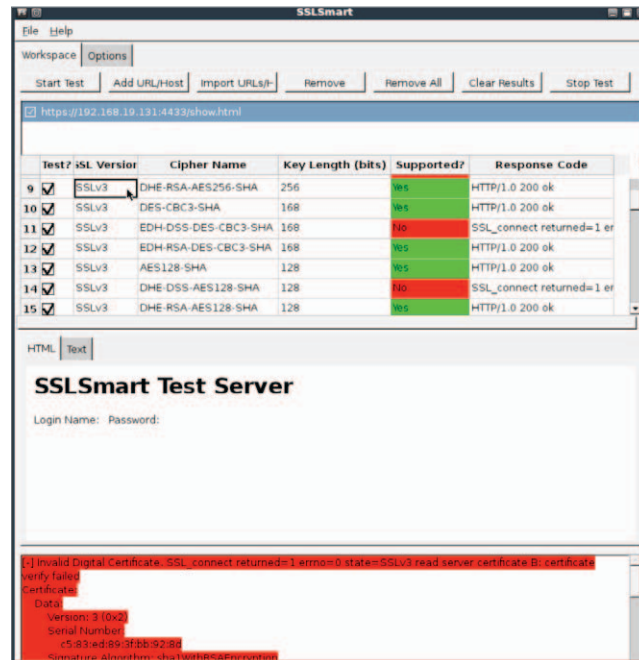


Figure 7. Image shows SSLSmart running on Linux (BackTrack 4).

## SSLSmart GUI

We will now look at various components of the SSLSmart GUI.

### The workspace

The workspace tab contains the controls required to start/stop scans, import hosts, view application response and results, and view progress. Figure 8 shows a screenshot and is followed by a brief discussion of various controls seen on the workspace tab.

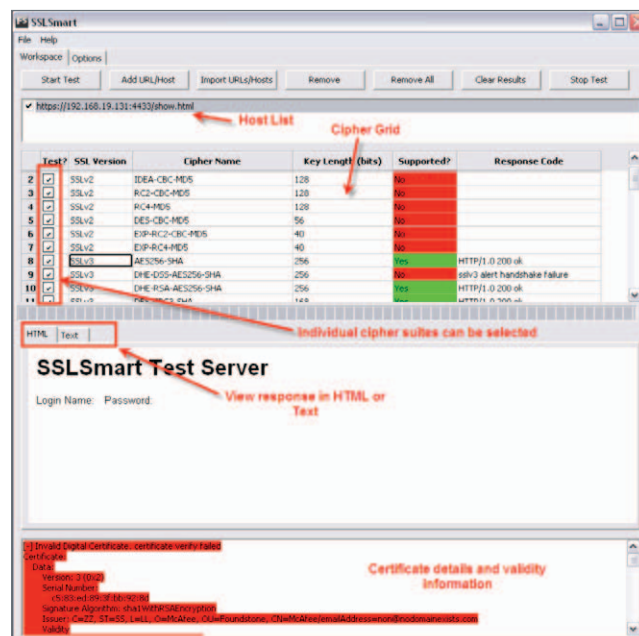


Figure 8. Image shows SSLSmart GUI workspace.



### Various buttons

1. *Start Test*—Starts test on all the added hosts. Once a test is started, the Options tab is not accessible until the test completes or is stopped.
2. *Add URL/Host*—Provides an input box to add a single URL.
3. *Import URLs/Hosts*—Allows several URLs/hosts to be imported for testing. Example entries are provided below:

```
http://www.example.com
http://www.example.com/file.jsp
https://www.example.com
https://www.example.com:449?query=123
www.example.com
11.11.11.11
www.example.com?query=123
```

4. *Remove*—Removes selected (checked or highlighted) hosts from the host list.
5. *Remove All*—Removes all hosts from the host list.
6. *Clear Results*—Clears all results if no scan is in a running state.
7. *Stop Test*—Stops a running test.

### Cipher grid

The cipher grid displays all the cipher suites that will be tested. This cipher grid is dynamic and can be changed at runtime (as long as no test is running) by applying OpenSSL filters from the Options tab. Individual cipher suites can be selected or deselected from the cipher grid. Only the cipher suites that are selected (checked) are tested.

### HTML/text

Once a URL is selected, server response for any chosen cipher suite is displayed in HTML and text formats. Occasionally, the HTML returned by server cannot be rendered for viewing. In such scenarios, the text tab helps view the exact server response for each cipher suite.

### Certificate pane

The certificate pane shows certificate details and validity information. The text background color turns green when a certificate is valid, red when a certificate is invalid.

### SSLSmart options

The Options tab contains various settings for SSLSmart. These settings allow the user to modify the type of scan, groups cipher suites as per SSL version, change proxy settings, and modify the cipher grid as per OpenSSL filters. Figure 9 is a screenshot of the Options dialog and is followed by a discussion of various controls seen on this tab.

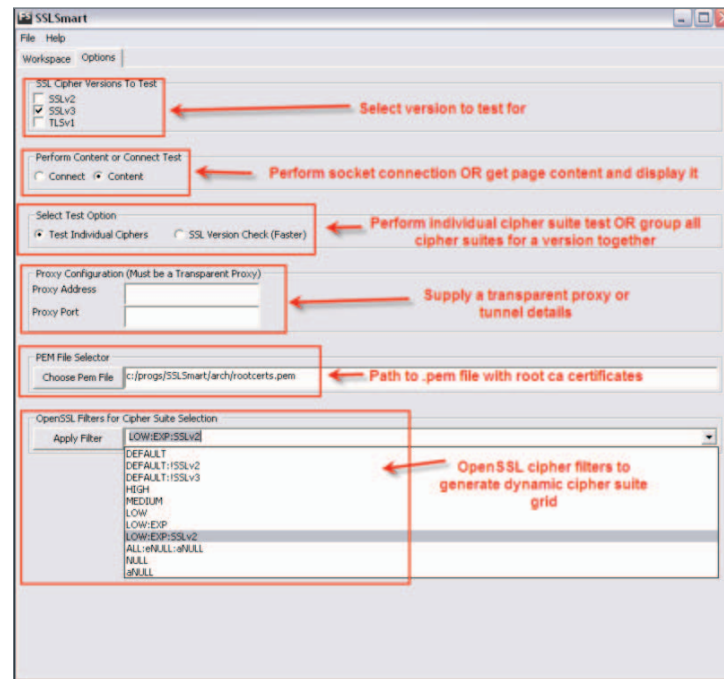


Figure 9. Image shows various SSLSmart options available.

**SSL Cipher Versions to Test:** Allows the user to select or deselect SSL versions to test. Selecting or de-selecting a checkbox next to the SSL version modifies the cipher grid in the workspace tab.

**Perform Content or Connect Test:** This group of radio buttons designates if a content or connect scan is performed.

1. *Content Scan*—Selecting this radio button allows the user to view a server response for a given cipher suite. This behavior is default and desirable to weed out false positives.
2. *Connect Scan*—Selecting this radio button allows the user to perform a quick SSL socket connection with given cipher suite. Success/failure in establishing a socket connection determines the result. This scan is faster and consumes fewer resources but is less accurate as it does not render any content in HTML or text tabs for review.

**Select Test Option:** This group of radio buttons determines if the cipher suites are individually tested or are grouped together for quick tests.

1. *Test Individual Ciphers (default)*—When this radio button is selected, all the cipher suites are individually tested.
2. *SSL Version Check (faster)*—When this radio button is selected, cipher suites are grouped as per SSL version. The cipher suites grouped for each version are determined by the OpenSSL Filter in the Options tab. For example, if the filter is set to NULL, and the Test Option is set to “SSL Version Check,” the ciphers for a particular SSL version will be as shown below:

	Test?	SSL Version	Cipher Name	Key Length (bits)	Supported?	Response Code
1	<input checked="" type="checkbox"/>	SSLv3	NULL-SHA, NULL-MD5	---		
2	<input checked="" type="checkbox"/>	TLSv1	NULL-SHA, NULL-MD5	---		

Figure 10. Image shows NULL ciphers grouped as per SSL version.

**Proxy Support:** The proxy settings provided here are used by SSLSmart to route all traffic. It is important to note that the provided proxy must be a transparent proxy for accurate results. If your proxy is not transparent, you will end up with results that show the cipher suites supported by the web proxy.

**PEM File Selector:** The .pem file provided here is used by Firefox to perform Root CA Certificate validation. The default file provided with SSLSmart is sourced from the cURL<sup>8</sup> project. You can generate your own .pem file with custom Root CA Certificates and supply it as an input here.

**OpenSSL Filters for Cipher Suite Selection:** Various OpenSSL filters can be used to generate the cipher suite grid. To select any particular filter, select the filter and click on Apply Filter. The corresponding cipher grid will be changed in the workspace tab. You can also type in your own OpenSSL filters for custom cipher grid generation.

### SSLSmart reporting

XML, HTML, text and their corresponding verbose versions are available to users. In addition to details in the normal report, verbose versions include complete application responses for each cipher suite and full details of server certificates.

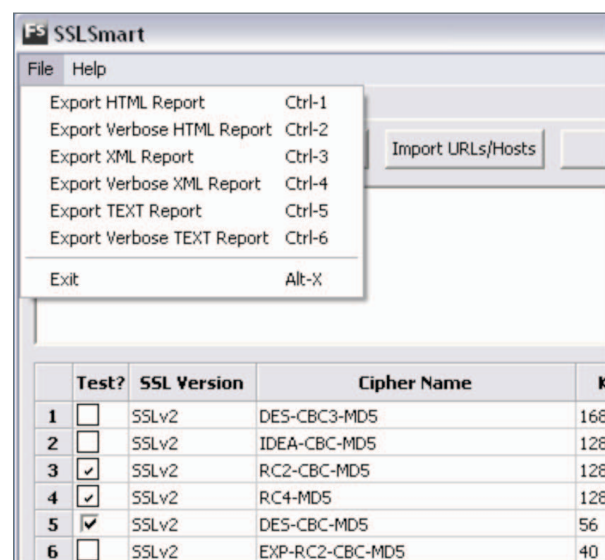


Figure 11. Image shows SSLSmart reporting options.

SSLSmart Results				
https://192.168.19.131:4433/show.html				
Supported?	Version	Cipher Suite	Bits	Response Code
No	SSLv3	ADH-RSA-MD5	128	ssh3 alert handshake failure
No	SSLv3	IDEA-CBC-SHA	128	ssh3 alert handshake failure
Yes	SSLv3	RC4-MD5	128	HTTP 1.0 200 ok
Yes	SSLv3	RC4-SHA	128	HTTP 1.0 200 ok
Certificate Details				
Validity	Invalid Digital Certificate: certificate verify failed			
Subject	C=ZZ ST=SSL O=McAfee OU=Foundstone CN=McAfee emailAddress=non@nodomaineists.com			
Issuer	C=ZZ ST=SSL O=McAfee OU=Foundstone CN=McAfee emailAddress=non@nodomaineists.com			
Valid Not before	Sun Jan 16 17:40:04 UTC 2011			
Valid Not after	Mon Jan 16 17:40:04 UTC 2012			

Figure 12. Image shows SSLSmart HTML report for medium strength SSLv3 ciphers.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <SSLSmart>
-   <url value="https://192.168.19.131:4433/show.html">
-     <cipher_suite>
-       <version>SSLv3</version>
-       <name>ADH-RC4-MD5:IDEA-CBC-SHA:RC4-SHA:RC4-MD5</name>
-       <bits>--</bits>
-       <supported>true</supported>
-       <response_code>HTTP/1.0 200 ok</response_code>
-     <full_response>
-       <CDATA[
-         HTTP/1.0 200 ok
-         content-type: text/html
-
-         <!--
-         <!--SSLSmart Test Server-->
-         Login Name: <input type="text"/>
-         Password: <input type="password"/>
-         <input type="submit" value="Login"/>
-         -->
-       ]>
-     </full_response>
-   </cipher_suite>
- </certificate>
- <validity>false</validity>
- <validity_msg>Invalid Digital Certificate.&nbsp;certificate verify failed</validity_msg>
- <subject>/C=ZZ/ST=SS/L=LL/O=McAfee/OU=Foundstone/CN=McAfee/emailAddress=non@nodomainexists.com</subject>
- <issuer>/C=ZZ/ST=SS/L=LL/O=McAfee/OU=Foundstone/CN=McAfee/emailAddress=non@nodomainexists.com</issuer>
- <not_before>Sun Jan 16 17:40:04 UTC 2011</not_before>
- <not_after>Mon Jan 16 17:40:04 UTC 2012</not_after>
- <version>2</version>
- <serial>14232480421156459149</serial>
- <signature_algorithm>sha1WithRSAEncryption</signature_algorithm>
- <extensions>
-   <extension>
-     <name>subjectKeyIdentifier</name>
-     <value>EC:B5:0A:16:5D:7B:23:93:E9:7E:74:2F:CF:16:0F:73:38:3F:DF:5B</value>
-   </extension>
-   <extension>
-     <name>authorityKeyIdentifier</name>
```

Figure 13. Image shows sample SSLSmart XML Verbose report.

## SSLSmart Sample Use Cases

We will now see a few use cases for SSLSmart and how some of the options can be used to perform efficient and accurate testing.

**Testing Only for SSLv2 Support:** In this use case, all the SSLv2 cipher suites are grouped together to communicate with the web server being tested. The resulting behavior is equivalent to a web browser talking SSLv2 to the web server.

1. **Options > Select Test Option > SSL Version Check (faster).**
2. Go to **Options > OpenSSL Filters for Cipher Suite Selection.**
3. Select **SSLv2** or type in **SSLv2** as the filter and click on **Apply Filter.**
4. Go to **Options**, and unselect **SSLv3** and **TLSv1** checkboxes from the **SSL Cipher Versions To Test** checkbox group.

**Performing Exhaustive Cipher Suite Check:** In this use case, all the cipher suites (including Anonymous and NULL ciphers) supported by the OpenSSL bindings are tested.

1. Go to **Options > OpenSSL Filters for Cipher Suite Selection**.
2. Select **DEFAULT:aNULL:eNULL** as the filter, and click on **Apply Filter**. This will add all the ciphers supported by current OpenSSL bindings to the cipher grid.
3. Add hosts and run the test to perform exhaustive cipher suites tests.

**Performing Certificate Verification Only:** In this use case, SSL certificate verification is performed for all hosts. No cipher verification is done.

1. Go to **Options**, and unselect all versions from **SSL Cipher Versions To Test**.
2. Add/import hosts and run the scan.
3. Only certificate verifications will be performed and certificate details can be seen in the certificate pane and the exported reports.

**Individual Cipher Suite Check for Low and Export Ciphers:** In this use case, LOW (56-bit key length) and Export (40-bit key length) grade SSL ciphers are tested individually.

1. Go to **Options > OpenSSL Filters for Cipher Suite Selection**.
2. Select **LOW:EXP** as the filter and click on **Apply Filter**.
3. Add hosts and run test.

**Group Cipher Check for Low and Export Ciphers:** In this use case, LOW (56-bit key length) and Export (40bit key length) grade SSL ciphers are grouped and tested as per SSL version.

1. **Options > Select Test Option > SSL Version Check (Faster)**.
2. Go to **Options > OpenSSL Filters for Cipher Suite Selection**.
3. Select **LOW:EXP** as the filter and click on **Apply Filter**.
4. Add hosts and run test. Various cipher groups will be tested.

### Using SSLSmart APIs

The APIs that form the backbone for SSLSmart tests can be used to write custom scripts. A few example scenarios are discussed below:

Query cipher suites supported by OpenSSL bindings:

```

irb(main):001:0> require 'rubygems'
=> true

irb(main):002:0> require 'sslsmartlib'
=> true

irb(main):003:0> a=OpenSSL::SSL.get_cipher_suites("DEFAULT") #Use OpenSSL's DEFAULT
filter to view SSL Cipher suites supported.
=> [{"DHE-RSA-AES256-SHA", "TLSv1/SSLv3", 256, 256}, {"DHE-DSS-AES256-SHA", "TLSv1/
SSLv3", 256, 256}, {"AES256-SHA", "TLSv1/SSLv3", 256, 256}, {"EDH-RSA-DES-CBC3-
SHA", "TLSv1/SSLv3", 168, 168}, ...]

irb(main):004:0> a=OpenSSL::SSL.get_mod_cipher_suites("DEFAULT")
=> [{"DES-CBC3-MD5", "SSLv2", 168, 168}, {"RC2-CBC-MD5", "SSLv2", 128, 128}, {"RC4-
MD5", "SSLv2", 128, 128}, {"DES-CBC-MD5", "SSLv2", 56, 56}, {"EXP-RC2-CBC-MD5",
"SSLv2", 40, 128}, ... <REMOVED> ... {"DES-CBC-SHA", "TLSv1", 56, 56}, {"EXP-EDH-
RSA-DES-CBC-SHA", "TLSv1", 40, 56}, {"EXP-EDH-DSS-DES-CBC-SHA", "TLSv1", 40, 56},
{"EXP-DES-CBC-SHA", "TLSv1", 40, 56}]

irb(main):005:0> a=OpenSSL::SSL.get_mod_cipher_suites("SSLv2") # View cipher suites
supported for SSLv2
=> [{"DES-CBC3-MD5", "SSLv2", 168, 168}, {"DES-CBC-MD5", "SSLv2", 56, 56}, {"EXP-
RC2-CBC-MD5", "SSLv2", 40, 128}, {"RC2-CBC-MD5", "SSLv2", 128, 128}, {"EXP-RC4-
MD5", "SSLv2", 40, 128}, {"RC4-MD5", "SSLv2", 128, 128}]

irb(main):006:0> get_cipher_suites("LOW")
=> [{"ADH-DES-CBC-SHA", "TLSv1/SSLv3", 56, 56}, {"EDH-RSA-DES-CBC-SHA", "TLSv1/
SSLv3", 56, 56}, {"EDH-DSS-DES-CBC-SHA", "TLSv1/SSLv3", 56, 56}, {"DES-CBC-SHA",
"TLSv1/SSLv3", 56, 56}, {"DES-CBC-MD5", "SSLv2", 56, 56}]

```

## Querying certificates:

```

irb(main):001:0> require 'rubygems'
=> true

irb(main):002:0> require 'sslsmartlib'
=> true

irb(main):003:0> q = Net::HTTP.new('192.168.19.131',4433) #create a new HTTP object
=> #<Net::HTTP 192.168.19.131:4433 open=false>

irb(main):004:0> cert = q.get_cert_details #get certificate details
=> #<ResultContainer:0x2cda788 @status=true, @data=#<OpenSSL::X509::Certificate
  subject=/C=ZZ/ST=SS/L=LL/O=McAfee/OU=Foundstone/CN=McAfee/emailAddress=non@
  nodomainexists.com,issuer=/C=ZZ/ST=SS/L=LL/O=McAfee/OU=Foundstone/CN=McAfee/
  emailAddress=non@nodomainexists.com, serial=14232480421156459149, not_before=Sun
  Jan 16 17:40:04 UTC 2011, not_after=Mon Jan 16 17:40:04 UTC 2012>>

irb(main):005:0> cert.class
=> ResultContainer

irb(main):006:0> cert.data #access data attribute for ResultContainer class
=> #<OpenSSL::X509::Certificate subject=/C=ZZ/ST=SS/L=LL/O=McAfee/OU=Foundstone/
  CN=McAfee/emailAddress=non@nodomainexists.com,issuer=/C=ZZ/ST=SS/L=LL/
  O=McAfee/OU=Foundstone/CN=McAfee/emailAddress=non@nodomainexists.com,
  serial=14232480421156459149, not_before=Sun Jan 16 17:40:04 UTC 2011, not_after=Mon
  Jan 16 17:40:04 UTC 2012>

irb(main):007:0> cert.data.class
=> OpenSSL::X509::Certificate

irb(main):008:0> puts cert.data.to_text #display certificate information in text
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      c5:83:ed:89:3f:bb:92:8d
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=ZZ, ST=SS, L=LL, O=McAfee, OU=Foundstone, CN=McAfee/
    emailAddress=non@nodomainexists.com
    Validity
      Not Before: Jan 16 17:40:04 2011 GMT
      Not After : Jan 16 17:40:04 2012 GMT
.... Removed for Brevity

```

## Querying cipher suites supported by web server:

```

irb(main):001:0> require 'rubygems'
=> true

irb(main):002:0> require 'sslsmartlib'
=> true

irb(main):003:0> q = Net::HTTP.new('192.168.19.131',4433)
=> #<Net::HTTP 192.168.19.131:4433 open=false>

irb(main):004:0> r = q.verify_ssl_config(:SSLv3, "AES256-SHA", Net::HTTP::CONTENT,
"/show.html") # "AES256-SHA" is supported for SSLv3
=> => #<ResultContainer:0x2cab000 @status=true, @data=#<Net::HTTPOK 200 ok
readbody=true>>

irb(main):005:0> r.status
=> true

irb(main):006:0> puts r.data.body # View response body
<html>
<h1>SSLSmart Test Server</h1>
Login Name: <input><br/>
Password: <input type="password"><br/>
<br/><input type="submit"><br/>
</html>
=> nil

irb(main):006:0> q.verify_ssl_config(:SSLv2,nil,Net::HTTP::CONTENT,"/show.html")
# SSLv2 is not supported.
=> #<ResultContainer:0x2c9cad0 @status=false, @data=OpenSSL::SSL::SSLError>

```

## Querying SSL version supported by web server:

```

irb(main):001:0> require 'rubygems'
=> true

irb(main):002:0> require 'sslsmartlib'
=> true

irb(main):003:0> q = Net::HTTP.new('192.168.19.131',4443)
=> #<Net::HTTP 192.168.19.131:4443 open=false>

irb(main):004:0> q.set_ssl_config(:SSLv3) #use only SSLv3
=> 0

irb(main):005:0> r = q.get("/show.html") #get "/show.html" path
=> #<Net::HTTPOK 200 OK readbody=true>

irb(main):006:0> q.set_ssl_config(:SSLv2) #use only SSLv2. This is not supported and
will result in error message when page is requested.
=> 0

irb(main):007:0> r = q.get("/show.html")
OpenSSL::SSL::SSLError: SSL_connect returned=6 errno=0 state=SSLv2 read server
hello A
    from /usr/lib/ruby/1.8/net/http.rb:586:in `connect'
    from /usr/lib/ruby/1.8/net/http.rb:586:in `connect'
    from /usr/lib/ruby/1.8/net/http.rb:553:in `do_start'
    from /usr/lib/ruby/1.8/net/http.rb:542:in `start'
    from /usr/lib/ruby/1.8/net/http.rb:1035:in `request'
    from /usr/lib/ruby/1.8/net/http.rb:772:in `get'
    from (irb):7

```

### Conclusion

To meet ever-growing business challenges, organizations continue to develop and deploy web applications that transmit sensitive information. To protect this sensitive information in transit, web servers must be tested and configured to support strong cryptographic algorithms. SSLSmart offers a reliable and efficient mechanism to test the strength of this SSL configuration to ensure the communication is as secure as it can be.

### About The Author

Gursev Singh Kalra serves as a Managing Consultant at McAfee Foundstone Professional Services. Gursev focuses on web application penetration testing, external assessments, and mobile application security testing. Gursev has also developed internal tools for internal and external network assessments. At McAfee Foundstone, Gursev has had the opportunity to work with some of the biggest financial, technology, and telecom companies.

### About McAfee Foundstone Professional Services

McAfee Foundstone Professional Services offers expert services and education to help organizations continuously and measurably protect their most important assets from the most critical threats. Through a strategic approach to security, McAfee Foundstone identifies and implements the right balance of technology, people, and process to manage digital risk and leverage security investments more effectively. The company's professional services team consists of recognized security experts and authors with broad security experience with multinational corporations, the public sector, and the U.S. military.

### About McAfee

McAfee, a wholly owned subsidiary of Intel Corporation (NASDAQ:INTC), is the world's largest dedicated security technology company. McAfee delivers proactive and proven solutions and services that help secure systems, networks, and mobile devices around the world, allowing users to safely connect to the Internet, browse and shop the web more securely. Backed by its unrivaled global threat intelligence, McAfee creates innovative products that empower home users, businesses, the public sector, and service providers by enabling them to prove compliance with regulations, protect data, prevent disruptions, identify vulnerabilities, and continuously monitor and improve their security. McAfee is relentlessly focused on constantly finding new ways to keep our customers safe. <http://www.mcafee.com>

