Foundstone®
STRATEGIC SECURITY

WHITE PAPER

The Need for Strong SSL Ciphers
Using Foundstone SSLDigger™ to Test SSL Security

By Rudolph Araujo

July 2004

Foundstone

Introduction

It has long been assumed that as long as a website or service is protected using the Secure Sockets Layer (SSL), it is secure. However, we now know that this is not true and now more than ever, it is critically important to recognize that SSL is by no means a panacea. Security researchers have long shown that the strength of any cryptographic approach is dependent on the algorithms and key lengths used by the underlying primitives. Consequently, the security of an SSL protected service is strongly correlated to the cipher suite in use as part of the protocol.

Background

The Secure Sockets Layer protocol was invented in 1994 by Netscape, developers of the popular browser and web server. The protocol had a major design goal of being easily extensible. It was therefore built to easily adopt newer and stronger protocols, for instance the Advanced Encryption Standard (AES) as they became available. This was to become SSL version 1.0. A few months later Netscape released version 2.0.

The SSL protocol is essentially composed of two major phases. The handshake phase involves authenticating one or both parties, exchanging keys and other such configuration information. Once the handshake is completed, the secure data exchange phase begins and all traffic between the client and the server is encrypted and has integrity protection. These two phases are depicted in Figure 1. It is important to note that the handshake phase in itself is a multi-step process. The various messages exchanged during this phase are also shown in Figure 1. Some of these messages are optional. For instance most secure websites on the Internet do not insist on client authentication and hence the client certificate does not need to be sent to the server by the client.

A major outcome of the handshake phase is to establish a cipher suite. A cipher suite is a set of cryptographic primitives and their configuration information. The cipher suite includes among other information, the key exchange algorithm to use, the authentication algorithm, encryption and hashing parameters. SSL for the most part uses symmetric key cryptography. However to deal with the key exchange problem in a shared key system, SSL chooses to first exchange a randomly generated session key using the slower public key cryptography. The key exchange phase thus involves setting up a special session key that both the client and server can then use to communicate securely. The authentication module as the name suggests is used to authenticate most commonly the server to the client. However, as mentioned above an optional message can also enforce that the client is successfully authenticated by the server before any encryption. The most commonly used technologies to implement this phase are digital certificates and the trusted third parties.

A digital certificate is meant to be much like a business card, albeit a secure one. An excellent analogy is with the concept of a drivers' license. A license is issued to a person by the Department of Motor Vehicles. In our analogy, the DMV would therefore represent the issuing authority. The DMV in turn is authorized by the government to issue licenses. The government therefore is analogous to the root certification authority. Finally, the license contains among other things the name and address of the licensee as well as what class of vehicles he/she is authorized to drive. Similarly, a digital certificate simply asserts that the certificate was issued for a particular purpose and to a particular person. It further states how long the certificate is valid, who the certification authority is and information on how to build a cipher chain that can be easily validated.

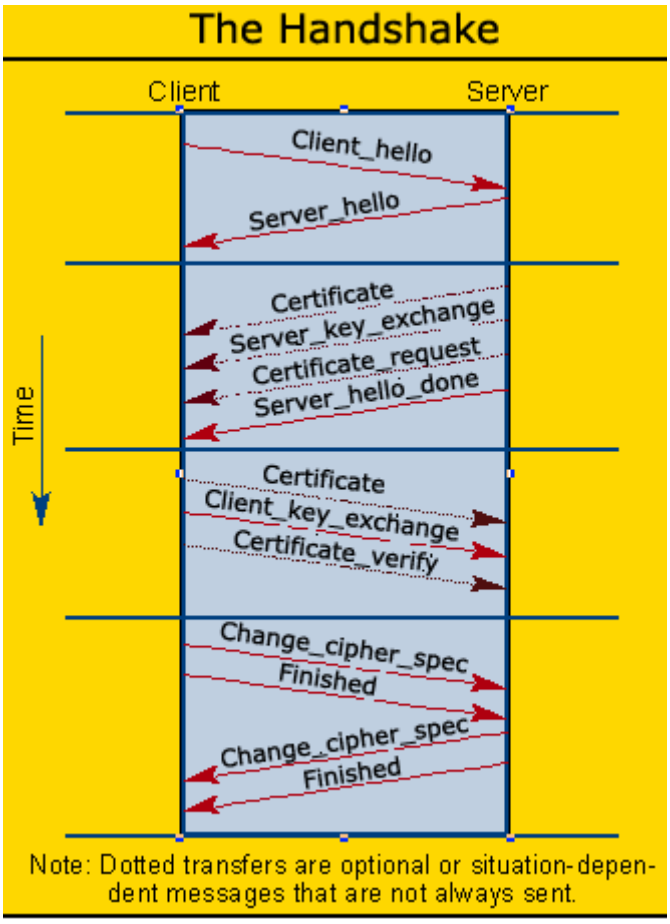

Figure 1: The SSL Handshake[1]

Perhaps unknowingly to most users of SSL or secure HTTP, the certificate store on Windows machines running IE and IIS contains a number of supposedly trusted third parties which can therefore act as root

[1] Network Magazine | SSL and TLS | December 4, 2002

certification authorities (CA). A screenshot of the partial list of trusted third parties is shown in Figure 2. The CA plays a critical role in this process and acts primarily like a matchmaker. When the client receives the server's certificate (sample shown in Figure 3), it has to first verify that the certificate was indeed issued by a party the client chosen to trust or that a chain of certificates can be built so that it ends at a party the client does trust. The CAs thus help to solve the initial key exchange problem.

The two other components of an SSL cipher are the encryption algorithm used to actually encipher the data and finally the hashing or Message Authentication Code (MAC) that helps ensure the recipients that the data has not been tampered with since it left the server.

Commonly used key exchange algorithms include Diffie Hellman and RSA. Perhaps the most popular authentication algorithm is RSA. However, servers are also known to use the digital signature standard (DSS). For encryption algorithms, both block and stream ciphers have been used. RC2, RC4, DES and AES are some of the more commonly used algorithms. Key sizes for these symmetric cryptographic primitives vary anywhere between 40 bits and 256 bits. Finally, for the MAC algorithm, the most frequently used algorithms are the 128 but long Message Digest 5 (MD5) and the 160 bit long Secure Hashing Algorithm or SHA1.

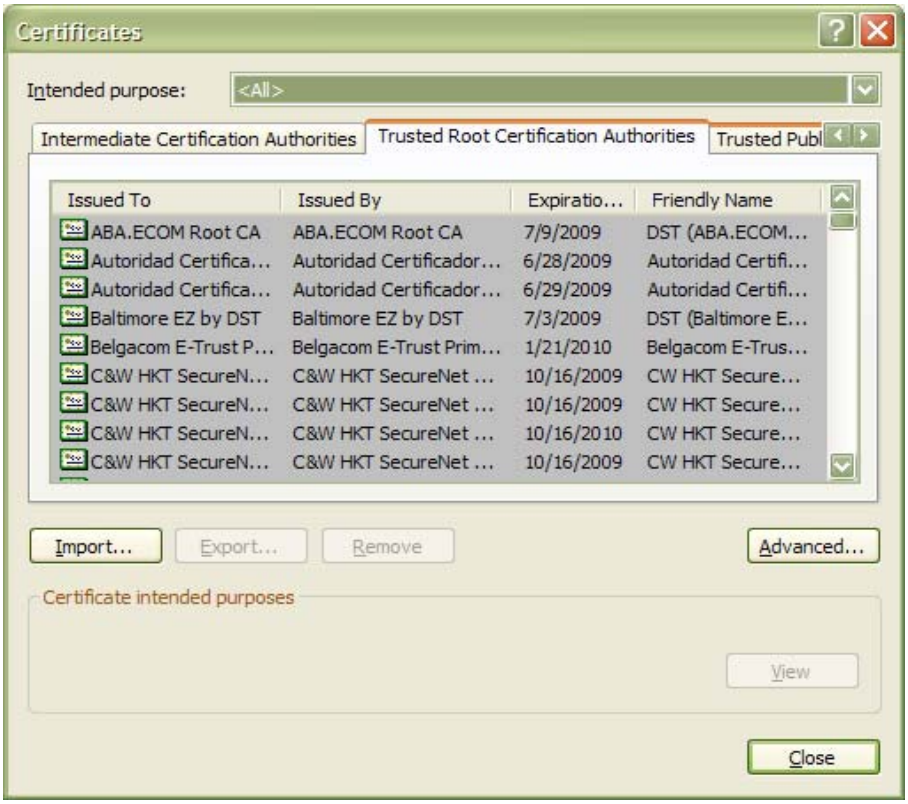

Figure 2: Microsoft Internet Explorer® List of Trusted Root Certification Authorities

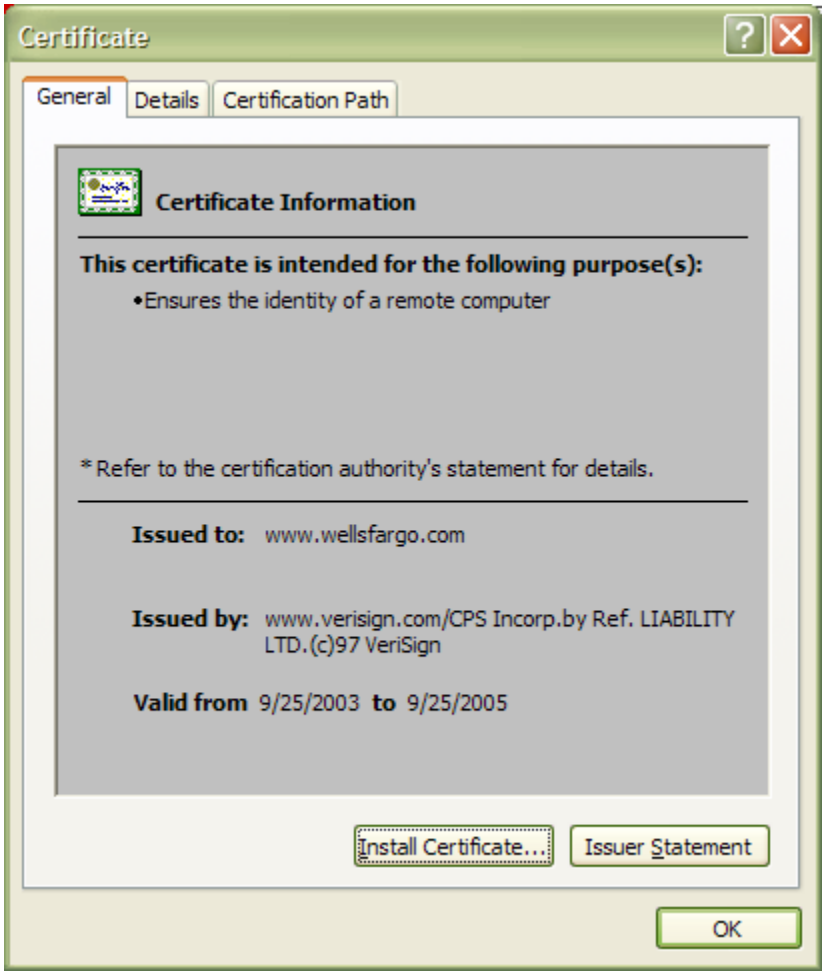

Figure 3: A certificate issued to a public bank by the certification authority VeriSign

SSL Version 2 Flaws

Security researchers have discovered two major protocol flaws with SSL version 2.0 that preclude its usage for effective security.

The Downgrade Attack: SSL v2 has no notion of integrity checking for handshake packets. Thus, an attacker could change the algorithms and key lengths chosen by the client while the appropriate handshake message was on the wire. This can result in a weak SSL connection being setup between the client and the server even though this is not what the two intended. Armed with this weak connection, the attacker could log all the traffic going by during the data exchange and could then use a brute force application to attack the weak encryption.

The Truncation Attack: SSL v2 does not allow the parties involved to distinguish when the connection is ended by one of them or by a malicious third party. Thus an attacker can freely interrupt secure client-server connections. If the attacker additionally understands the application, the semantics and ordering of messages being exchanged, then he/she can potentially alter the meaning of a message by interrupting the connection at precisely the correct instant.

In response to these two glaring security holes, in November 1996, Netscape publicly released the specification for version 3.0 of the protocol which fixed these two issues. SSL 3.0 formed the core of the IETF protocol Transport Layer Security version 1 (TLS v1) which was standardized in RFC 2246.

Why Are Key Lengths So Important?

All versions of SSL are heavily dependent on the key lengths for their strength. This is an important security attribute of the protocol because shorter key sizes make it far quicker for an attacker to guess the key. In any cryptographic implementation, the strength of the primitives is dependant on the secrecy (and therefore the length) of the key. This is not to say that the algorithms used are unimportant. On the contrary it is vital from a security perspective to use algorithms that have been tried and tested and are widely accepted by the community to be secure. With this in mind symmetric key lengths of less than 128 bit are generally considered insufficient. For instance, on July 14 1995, a challenge was issued to break an RC4 – 40 bit SSL session. This was completed in 8 days by among others a French student[2] using free CPU cycles on 120 computers in his school lab. Netscape at that time estimated that such an effort will cost approximately $10,000 in computing time. Most researchers have disputed this figure and since then, RSA has issued a number of such challenges that have attempted to break even larger key lengths. Table 1 details the successful attempts and the time taken as well as, where available, the costs involved.

Date	Winner	Time	Cost
June 1997	DESCHALL	96 days	
February 1998	Distributed.Net	41 days	100,000 computers
July 1998	EFF Deep Crack	56 hours	$210,000 (40MHz chips)
January 1999	Distributed.Net + EFF	22 hours	
September 2002 (RC5-64)	Distributed.Net	4 Years	331,252 volunteers lending spare time on their computers

Table 1: DES 56 Bit Cracking Efforts

[2] http://pauillac.inria.fr/~doligez/ssl/index.html

In January 1996, a committee comprising of notables including Whitfield Diffie, Bruce Schneier and Ronald Rivest among others, suggested that a 40 bit key could be broken in approximately 5 hours using specialized hardware costing a mere $400. With Moore's law and the decreasing cost of hardware, coupled with the fact that per-unit prices of chips are far lower when chips are bought in large number, one can be fairly convinced that the cost is far lower in 2004. To provide an estimate it is interesting to consider the research done by Arjen Lenstra and Eric Verheul[3].

Lenstra and Verheul based their analysis on four hypothesis:

- When DES was standardized in 1982 it was assumed that 5×10^5 MIPS (mega-instructions per second) Years (MY) was the security needed by commercial applications. 1 MY was defined to be the equivalent of 1 year of computation on a VAX 11/780. By the current computing standards in 1999, when Lenstra and Verheul published their paper, this was the equivalent of 20 hours on a 450 MHz Pentium II computer. Therefore, 5×10^5 MY would be equal to 14000 months on a 450 MHz PII or 2 months on 7000 such processors.

- The amount of RAM like computing power (Moore's Law) available per dollar roughly doubles every 18 months. This equates to roughly 100 times more power and RAM for the same dollar amount every 10 years.

- The budgets of corporations roughly doubles every 10 years. One can think of the corporations as attackers in this context. This hypothesis was based on the United States Gross National Product which doubles every ten years when measured in contemporary dollars.

- Finally, cryptanalytic progress halves the computational effort required to invert the half keys used in the cryptographic algorithms like DES and RSA. This hypothesis is consistent with cryptanalysis algorithmic improvement through the 70s, 80s and 90s.

Combining the first three of the above, we can extrapolate as follows: if in 1982, 5×10^5 MY was assumed to be infeasible then, $100 \times 2 \times 5 \times 10^5$ MY or 10^8 MY would be infeasible in 1992 or 2×10^{10} MY in 2002, or 4×10^{12} MY would be infeasible in 2012. Thus, their paper proposes that the minimum key lengths needed to be secure in 2005 for instance would be 1149 for an asymmetric key implementation and 74 for a symmetric key algorithm. Appendix A provides the minimum key lengths based on the Lenstra and Verheul analysis for a given year.

[3] Selecting Cryptographic Key Sizes, November 24, 1999

Based on the above table, for 2004, a corporation or a nation state willing to make a significant investment could thus successfully break encryption up to 64 bits at a relatively reasonable cost. The motivation to do this must obviously exceed the cost of cracking the encryption.

What complicates matters even more is that until January 2000, cryptographic products exported outside the United States were restricted since they were classified as munitions. The export controls attempted to prevent the general use of symmetric encryption with key lengths greater than 40 bits and asymmetric encryption with key lengths greater than 512 bits. As a result of these controls, browsers like Netscape Navigator and Microsoft Internet Explorer, which were exported from the United States internationally, could only perform cryptographic operations with keys of length 40bits or less. However, to allow financial institutions that did business abroad and had a requirement for strong encryption, an SSL extension called Server-Gated Cryptography (SGC)[4][5] was introduced in 1998. SGC certificates (also known as Global ID server certificates) issued by Verisign and Thawte were signed by a special CA certificate which enabled strong encryption in export browsers. The sequence in setting up a connection is very similar to the default SSL behavior. The browser would first initiate an SSL handshake with an export grade cipher. However, on receiving and verifying this special Global ID certificate, the browser would upgrade the cipher suite and essentially engage in an SSL handshake all over again.

In the light of the above, while building the SSLDigger tool we have attempted to classify the ciphers tested into 4 different categories:

No Security Ciphers: These are the NULL ciphers that essentially use no form of encryption. As is fairly obvious then these ciphers provide no security.

Weak Security Ciphers: All ciphers with key lengths less than 128 bits are classified in this category. Ciphers using anonymous Diffie Hellman are also classified as weak irrespective of key lengths since this key exchange protocol provides for no authentication and thus is highly susceptible to a man-in-the-middle attack.

Strong Security Ciphers: This category includes ciphers with key strengths between 128 bits and 256 bits. These are perhaps the most commonly used ciphers today by modern browsers and web servers.

Excellent Security Ciphers: Ciphers using 256 bit AES encryption fall into this category and are widely regarded to be the most secure.

[4] http://httpd.apache.org/docs-2.0/ssl/ssl_howto.html#upgradeenc
[5] http://www.microsoft.com/windows2000/en/server/iis/htm/core/iistesc.htm

Disabling Weak Cipher Suites

The methods for disabling specific SSL cipher suites vary based on the web server and the underlying operating system.

- Microsoft Internet Information Services (IIS): Microsoft Knowledge Base article 216482 and 245030[6] provides the method for disabling specific ciphers by editing the Windows registry.
- Apache: The Apache 2 HOWTO guide[7] describes the mechanism to allow/disable groups of ciphers. Apache 1.3 uses a similar process and is documented within the mod_ssl HOWTO guide[8].

The Need for SSLDigger

As more and more users become concerned with online security and privacy, we believe that a tool like SSLDigger is invaluable to both site administrators as well as web site penetration testers. For increased accuracy in determining which cipher suites are supported by a given web server, the tool performs a full SSL connection to the server with each of the ciphers under test described later in this document. While this has a minor performance impact (due to the need to wait for timeouts when ciphers are not supported), it does prevent any false positives or negatives.

This tool is also useful from the perspective of determining compliance to regulatory and industry standards. For instance from the VISA Cardholder Information Security Program (CISP) certification process[9]:

> *Requirement 4: Encrypt transmission of cardholder and sensitive information across public networks.*
> *4.3 Implement strong cryptography and appropriate key controls to safeguard data during transmission.*
> *4.3.a Verify that at least **128 bit encryption** is used during data transmission.*
>
> *Requirement 10: Regularly test security systems and processes.*
> *10.1 Test security controls, limitations, network connections, and restrictions routinely to make sure they can adequately identify or stop any unauthorized access attempts.*
> *10.1.a Confirm through inquiry that periodic security testing of the devices within the VISA cardholder environment occurs.*

[6] http://support.microsoft.com/default.aspx?scid=kb;en-us;216482
http://support.microsoft.com/?kbid=245030
[7] http://httpd.apache.org/docs-2.0/ssl/ssl_howto.html
[8] http://www.modssl.org/docs/2.8/
[9] http://usa.visa.com/business/merchants/cisp_index.html?ep=v_sym_cisp

Similarly, the Health Insurance Portability and Accountability Act[10] (HIPAA) of 1996 also has a requirement for encryption:

> *Transmission Security*
> *Implement technical security measures to guard against unauthorized access to electronic protected health information that is being transmitted over an electronic communications network.*
> *164.312(e)(1) Encryption (A)*
> *Implement a mechanism to encrypt electronic protected health information whenever deemed appropriate.*

[10] http://aspe.hhs.gov/admnsimp/pl104191.htm

SSLDigger Features

The major features in version 1.0 of the SSLDigger tool include:

- Full Browser Support using the Microsoft Internet Explorer Browser Control
- Support for operating the tool in batch mode for operating on multiple sites simultaneously. This is very useful in testing a large number of websites. The input to the tool is provided in this case via a text file. A sample input file is shipped with the tool (sample-urllist.txt). The contents of this file are simply new line-separated websites.
- The tool supports reporting in three different formats. An XML format is supported for providing access to the raw data gathered. The CSV format allows the user to open the report in a spreadsheet program. The most graphical format is the HTML report that provides links into this whitepaper as well as links to the URLs that have been tested.
- Limited Support for Server Gated Cryptography. This support provides additional information to the user for use while interpreting the results and letter grade below.
- The tool tests for 26 commonly used ciphers that are detailed in Appendix B and are classified as described above into No Security, Weak, Strong and Excellent Security ciphers.
- Based on the testing, the site is assigned a grade. The following grading scheme is used to determine a letter grade:

SSL Results	Grade
Only Excellent Security Ciphers supported	A+
Only Strong and Higher Security Ciphers supported	A
Any Weak Security Cipher supported	B
Only Weak Security Ciphers supported	C
Any No Security Cipher supported	D
Only Weak and No Security Ciphers supported	D-
Only No Security Ciphers supported	F

Table 2: Grading Scheme

- The application is configurable via the Foundstone.FreeTools.SSLDigger.exe.config file shipped with the tool. Parameters that can be changed are listed in Table 3 below:

Parmeter	Definition
LogFilePath	File system path where log files will be created. Default: <INSTALL_DIR>\logs\. Ensure that the path includes the trailing \ character.
HighestSSLVersion	The maximum version of SSL supported. Currently supported values are "3.0" (SSL v3) and "3.1" (SSL v3.1/TLS v1). Default: "3.1"
ClientSSLCert	The path to a client certificate file if one is to be used. Default: None
ServerTimeoutMS	The timeout in milliseconds to be used when connecting to the web server. Default: 3000 ms

Table 3: Foundstone.FreeTools.SSLDigger.exe**.config Parmeters**

SSLDigger Installation

- Pre-requisites: SSLDigger is a .NET managed assembly built using C#. It requires the use of the Microsoft .NET framework version 1.1. This maybe obtained using Windows update or by visiting the following URL: http://msdn.microsoft.com/netframework/howtoget/default.aspx. SSLDigger has been tested on Windows XP workstations running the .NET v1.1. While it has not been tested on other versions of Windows, we do believe that it should execute successfully on all Windows operating systems that can support the 1.1 framework.

- Installation Steps: SSLDigger can be downloaded from the Foundstone website at http://www.foundstone.com/services. After double clicking the setup for the tool, the splash screen shown in Figure 4 will be shown. Click Next to proceed in the installation. Figure 5 displays the license agreement that must be accepted in order to install the tool. On clicking Next, the user is then asked to specify the directory into which the tool is to be installed. By default this path is C:\Program Files\Foundstone Free Tools\Foundstone SSLDigger\. A reports directory will be created under the installation directory which represents the default path where reports of SSL tests will be stored. Similarly, the tool will attempt to create logs of errors encountered in the logs sub directory. Figures 6, 7, 8, 9 and 10 represent the remaining steps in the installation wizard and are fairly straightforward. Figure 9 details the known issues with this tool that are also described in more detail in a later section.

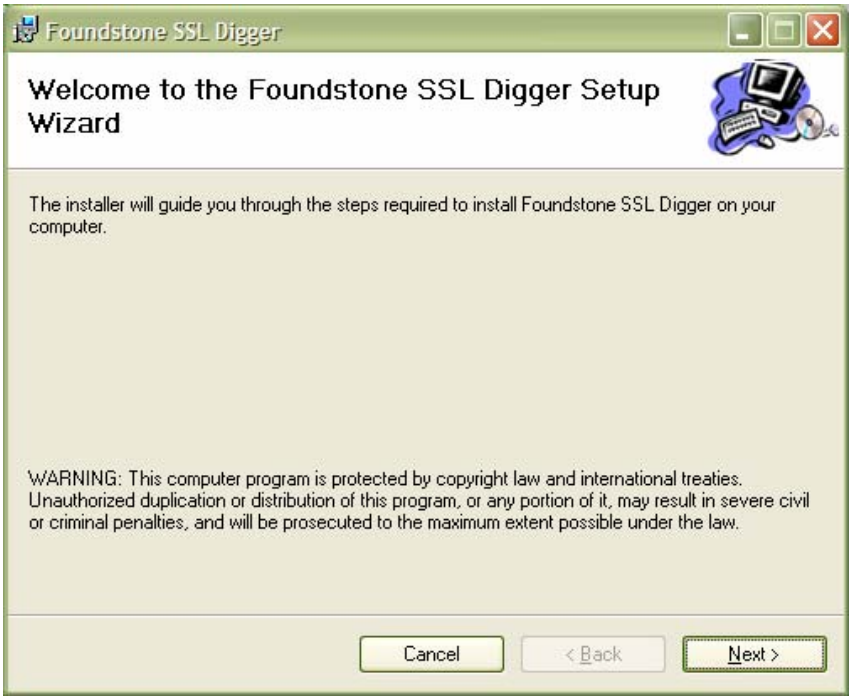

Figure 4

Figure 5

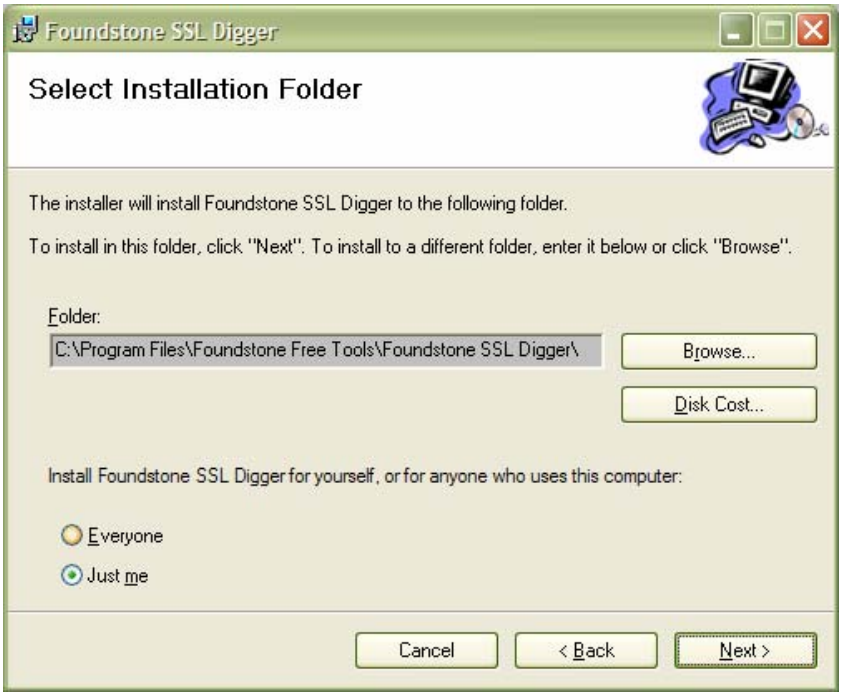

Figure 6

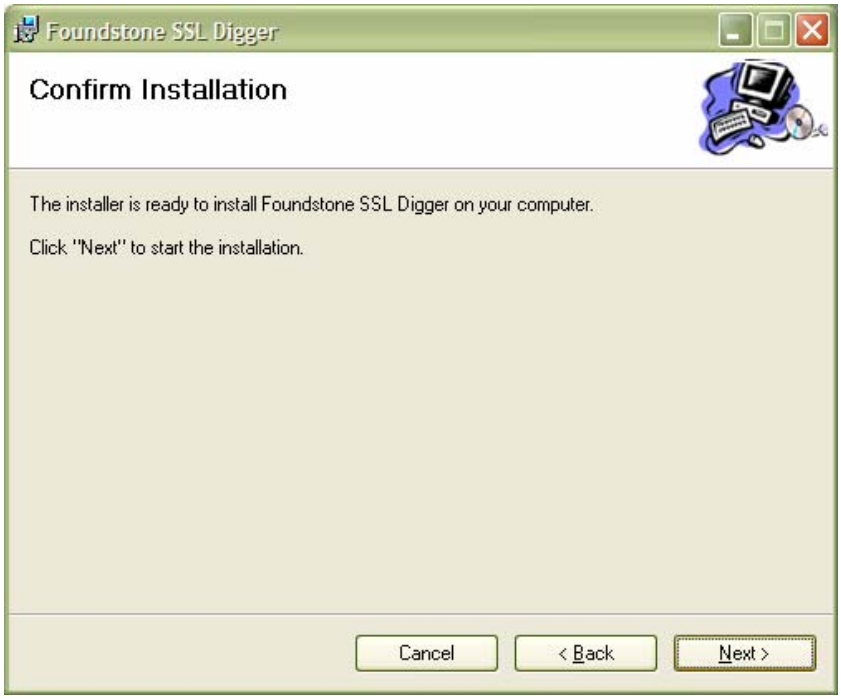

Figure 7

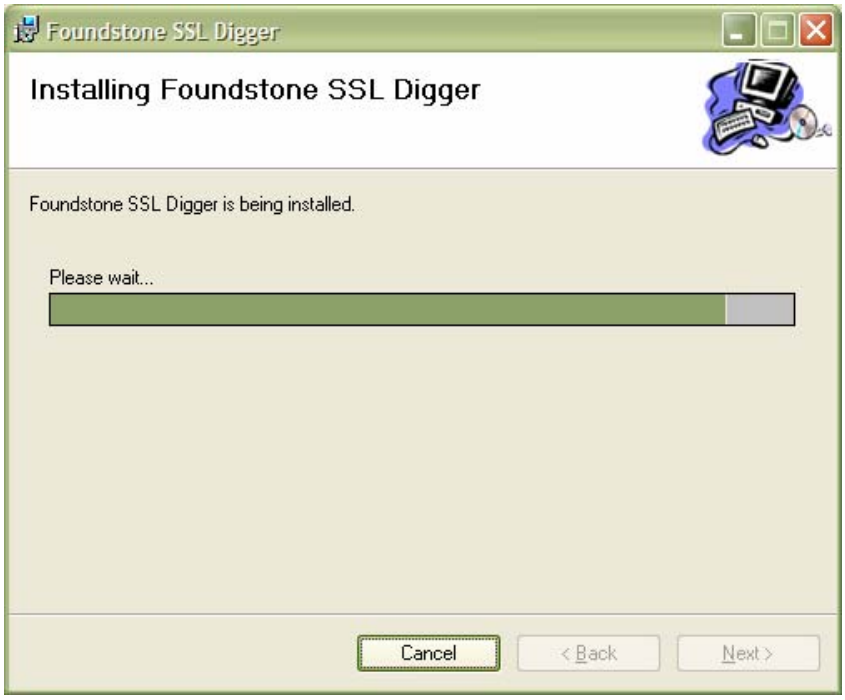

Figure 8

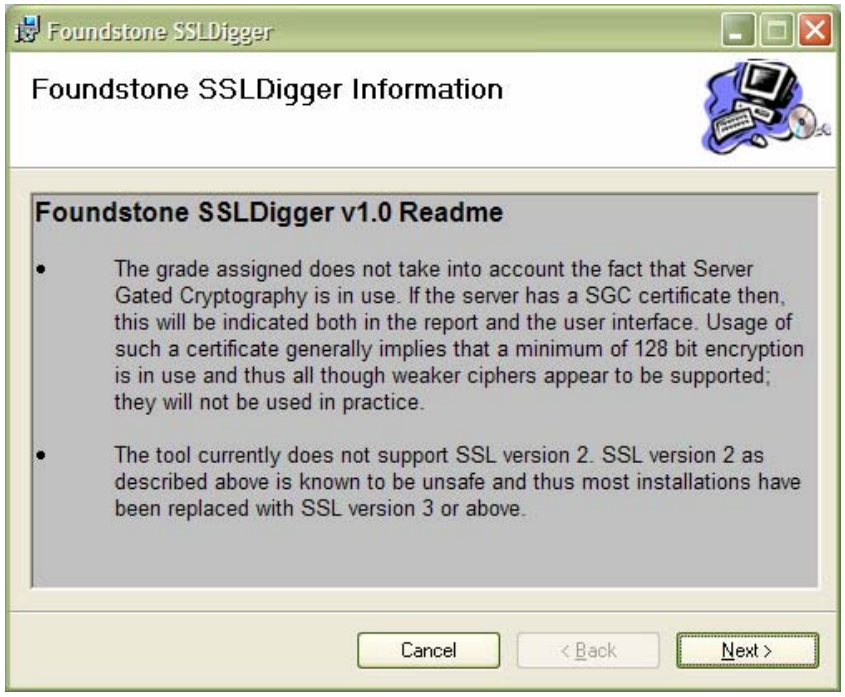

Figure 9

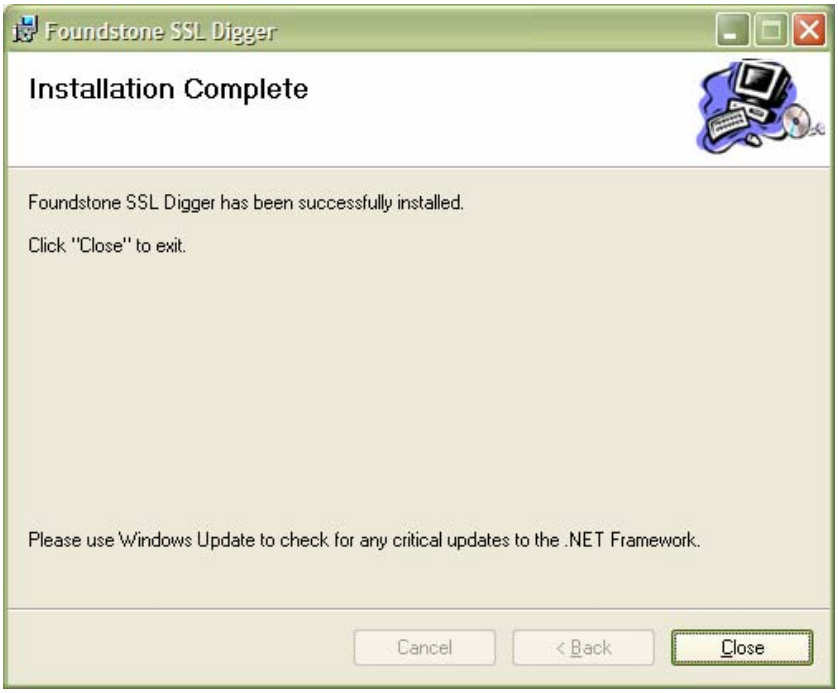

Figure 10

Foundstone

Using SSLDigger

To use the tool in single site mode, simply start it through the start menu and use the browser built into the tool. If SSL is enabled on the server and a certificate is received, its contents will be displayed in the lower pane. The "Test SSL" button can then be used to launch the cipher strength testing against the website. The Results pane displays in a sortable grid, the results of the testing as well as other information such as if a given cipher is exportable under the export control legislation or not. A progress bar and the grade for the website under test are also displayed. When testing is complete, the user is provided with the option to Save/View the final report.

In batch mode, select the input file and the familiar results dialog described will appear. Only in this case, multiple tabs will be shown with results for each of the servers being tested.

SSLDigger accepts addresses in the following formats:

- http://www.google.com
- https://www.google.com
- www.google.com
- google.com
- 66.102.7.104

Foundstone

Known Issues

- The grade assigned does not take into account the fact that Server Gated Cryptography is in use. If the server has a SGC certificate then, this will be indicated both in the report and the user interface. Usage of such a certificate generally implies that a minimum of 128 bit encryption is in use and thus all though weaker ciphers appear to be supported; they will not be used in practice.
- The tool currently does not support SSL version 2. SSL version 2 as described above is known to be unsafe and thus most installations have been replaced with SSL version 3 or above.

Acknowledgements

The tool uses the Mentalis .NET Security Library[11]. This library has attempted to port much of the functionality present in the Microsoft CryptoAPI (CAPI) to the .NET managed world.

Mark Curphey, Director of Consulting at Foundstone, Inc. is primarily responsible for the conceptualization of this tool and for creating the initial specifications. Chris Prosise (Vice President of Professional Services, Foundstone, Inc.), Eric Heitzman, Todd McBride and the rest of the "Con" group at Foundstone provided significant help along the way especially with usability issues, testing and the whitepaper.

[11] http://www.mentalis.org/soft/projects/seclib/

Foundstone

Appendix A: Minimum Key Lengths Based on Lenstra & Verheul Analysis

Year	Minimum Symmetric Key Length	Minimum Asymmetric Key Length
1982	56	417
1987	60	539
1992	64	682
1997	68	844
2000	70	952
2001	71	990
2002	72	1028
2003	73	1068
2004	73	1108
2005	74	1149
2006	75	1191
2007	76	1235
2008	76	1279
2009	77	1323
2010	78	1369
2011	79	1416
2012	80	1464
2013	80	1513
2014	81	1562
2015	82	1613
2016	83	1664
2017	83	1717
2018	84	1771
2019	85	1825
2020	86	1881
2021	86	1937
2022	87	1995
2023	88	2054
2024	89	2113
2025	89	2174
2026	90	2236
2027	91	2299
2028	92	2362
2029	93	2427
2030	93	2493

Appedix B: Ciphers Used in Testing

Cipher Suite[12]	Cipher Strength	Description
NULL-MD5	No Security	This cipher uses no encryption and simply uses the MD5 algorithm for hashing. Thus, the data being transmitted is provided with no confidentiality.
NULL-SHA	No Security	Like the cipher above, this one too uses no encryption. The only difference being the SHA1 algorithm is used for hashing instead of MD5.
EXP-DES-CBC-SHA	Weak Security	The EXP in the name of the cipher denotes that this is an exportable algorithm. It is marked as a weak security cipher since it uses 512 bit RSA keys for the key exchange and a 40 bit DES key for encryption.
EXP-RC2-CBC-MD5	Weak Security	This cipher differs in the encryption algorithm used which is 40 bit RC2. The key exchange algorithm still remains 512 bit RSA. MD5 is used instead of SHA1 as the MAC algorithm.
EXP-RC4-MD5	Weak Security	Another exportable cipher, this one uses the stream cipher RC4 with a 40 bit key for encryption.
EXP1024-DHE-DSS-DES-CBC-SHA	Weak Security	This cipher uses a 1024 bit ephemeral Diffie Hellman algorithm for key exchange and the digital signature standard (DSS) for authentication. Ephemeral DH differs from

[12] http://www.openssl.org/docs/apps/ciphers.html

		fixed DH since it uses one time keys. This in turn ensures that a different secret key is setup each time between the same server and client.
EXP1024-DHE-DSS-RC4-SHA	Weak Security	This cipher is similar to the one above and is also exportable using a 56 bit RC4 encryption algorithm.
EXP1024-DES-CBC-SHA	Weak Security	Using a 56 bit DES key for encryption, this cipher also uses a 1024 bit RSA key for key exchange and the SHA1 algorithm for hashing.
EXP1024-RC4-SHA	Weak Security	This is the last of the exportable ciphers and uses a 56 bit RC4 key.
DES-CBC-SHA	Weak Security	While this cipher is not exportable , it still uses a 56 bit DES key. The key exchange is done using non-exportable 1024 bit RSA.
ADH-AES128-SHA	Weak Security	The next two ciphers are marked as weak security even though they use 256 bit long AES keys because they utilize the anonymous Diffie Hellman algorithm to facilitate the key exhchange. Anonymous DH involves no authentication with each side sending its public DH parameters to the other. This approach is therefore susceptible to man-in-the-middle attacks in which the attacker conducts ADH with both parties.
ADH-AES256-SHA	Weak Security	For the same reasons described above, this cipher is marked to be weak security.
DH-DSS-AES128-SHA	Strong Security	This cipher uses fixed Diffie Hellman for key exchange which in turn results in a fixed secret key between two peers, based on the DH calculation using the two fixed DH public

		keys.
DH-RSA-AES128-SHA	Strong Security	Differing only in the choice of authentication algorithms, this cipher and one above have the same issue with a fixed secret key.
DHE-DSS-RC4-SHA	Strong Security	The next two ciphers are similar to the two above with the main difference being the use of ephemeral Diffie Hellman which is widely regarded to be the most secure of the three DH variations especially since a temporary authenticated key is created every time.
DHE-DSS-AES128-SHA	Strong Security	Like the cipher above this one uses ephemeral Diffie Hellman as well.
DHE-RSA-AES128-SHA	Strong Security	Again using ephemeral Diffie Hellman, this cipher differs from the above in the choice of authentication algorithms.
RC4-MD5	Strong Security	This is perhaps one of the most common ciphers in use on the Internet together with the next three ciphers below. It uses a 1024 bit RSA key exchange and uses RSA for authentication as well. A 128 bit RC4 key is used for encryption.
RC4-SHA	Strong Security	This cipher is identical to the one above except in the choice of MAC algorithms.
AES128-SHA	Strong Security	One of the newer AES based ciphers, this one also uses RSA for both key exchange and authentication.
DES-CBC3-SHA	Strong Security	Using Triple DES which was a stop-gap improvement on DES, this cipher thus relies on a 168 bit key for encryption.

DH-DSS-AES256-SHA	Excellent Security	All of the excellent security ciphers utilize 256 bit AES keys for encryption. This cipher uses fixed Diffie Hellman for key exchange and DSS for authentication.
DH-RSA-AES256-SHA	Excellent Security	Similar to the one above, this one uses RSA for authentication.
DHE-DSS-AES256-SHA	Excellent Security	The next two ciphers are similar to the previous two respectively differing only in their use of ephemeral Diffie Hellman for key exchange which for reasons explained above is considered to be more secure.
DHE-RSA-AES256-SHA	Excellent Security	Using ephemeral Diffie Hellman for key exchange and RSA for authentication, this cipher is similar the one above.
AES256-SHA	Excellent Security	The standard excellent security cipher uses a 256 bit AES encryption key and RSA for both key exchange and authentication.

Foundstone

About Foundstone

Foundstone® Inc., experts in strategic security, offers a unique combination of software, services, and education to help organizations continuously and measurably protect the most important assets from the most critical threats. Through a strategic approach to security, Foundstone identifies and implements the right balance of technology, people, and process to manage digital risk and leverage security investments more effectively.

Foundstone Strategic Secure Software Initiative S3i (TM) helps clients define, design, develop, deploy and maintain reliable and secure software. By understanding and managing inherent risk and measurably improving the software development life cycle, Foundstone helps its clients reduce development costs and improve performance.

The company has one of the most dominant security talent pools ever assembled, and has authored ten books, including the best seller *Hacking Exposed*. Foundstone is headquartered in Orange County, CA, and has offices in New York, Washington, D.C., and Seattle. For more information about Foundstone and Foundstone Enterprise Risk Solutions, visit www.foundstone.com, or call 877.91.FOUND within the U.S, and 949.297.5600 outside the U.S.

