Cryptographic SSL/TLS Current and Recent Practices

Amit Sharma*
Department of Computer Science
SKIT Jaipur
Rajasthan,IN
sharma.amit1100@gmail.com

ABSTRACT

The Secure Socket Layer (SSL) and its variant, Transport Layer Security (TLS), are used toward ensuring server security. In this paper, we characterize the cryptographic strength of public servers running SSL/TLS. We present a tool developed for this purpose, the Probing SSL Security Tool (PSST), and evaluate over 19,000 servers. We expose the great diversity in the levels of cryptographic strength that is supported on the Internet. Some of our discouraging results show that most sites still support the insecure SSL 2.0, weak export-level grades of encryption ciphers, or weak RSA key strengths. We also observe encouraging behavior such as sensible default choices by servers when presented with multiple options, the quick adoption of AES (more than half the servers support strong key AES as their default choice), and the use of strong RSA key sizes of 1024 bits and above. Comparing results of running our tool over the last two years points to a positive trend that is moving in the right direction, though perhaps not as quickly as it should.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Measurement, Security

Keywords

SSL, Network Security, Servers

*This research was supported in part by the New York Software Industry Association under grant NYS-CU0232901.

IMC'07, October 24-26, 2007, San Diego, California, USA.

1. INTRODUCTION

Cryptography is an essential component of modern electronic commerce. With the explosion of transactions being conducted over the Internet, ensuring the security of data transfer is critically important. Considerable amounts of money are being exchanged over the network, either through e-commerce sites (*e.g.,* Amazon, Buy.com), auction sites (*e.g.,* eBay), on-line banking (*e.g.,* Citibank, Chase), stock trading (*e.g.,* Schwab), and even government (*e.g.,* irs.gov). Communication with these sites is secured by the Secure Sockets Layer (SSL) or its variant, Transport Layer Security (TLS), which are used to provide authentication, privacy, and integrity. A key component of the security of SSL/TLS is the cryptographic strength of the underlying algorithms used by the protocol. It is crucial to ensure that servers using the SSL protocol have employed it properly. For example, it should be determined whether site administrators are using the best practices, are aware of their sites' vulnerabilities if they haven't already been addressed, and are promptly reacting to CERT advisories. Poor use of cryptography may be an indicator of poorly-administered security. Experience in related areas of patch management and virus/worm propagation is not encouraging. The recent interest in SSL-based VPNs only increases the need to study SSL.

One key feature of SSL/TLS is that it allows negotiation between two peers. Different implementations will not necessarily support the same cryptographic algorithms. Thus SSL allows two peers to determine a subset of common cryptographic routines. This allows for the interoperability and extensibility of the protocol. For example, SSL allows different algorithms to be used for authentication (*e.g.,* RSA, DSS), key exchange (RSA, EDH), encryption (RC2, RC4, DES, 3-DES, AES), and integrity (MD5, SHA-1). This flexibility allows for new, stronger algorithms to be added over time (such as AES) and reduces dependence on any one algorithm, in case that algorithm is broken or succumbs to brute-force exhaustive search techniques (as with DES).

While this flexibility improves interoperability, it may also compromise security. For example, server administrators may wish to support as wide a range of protocols as possible in order to maximize the number of clients that can access a site. However, they may be lax in removing features that have compromises in security. For example, if a site supports a weak form of encryption, a client may choose to use that algorithm for performance or power consumption reasons (*e.g.,* on a wireless PDA), without recognizing the dan-

gers. This could lead to a session being broken, a customer's password being cracked, and an empty bank account. While this could be considered simply a case of clients suffering the consequences of their actions, there are reasons to prevent this from happening. This type of experience could alienate a customer, damage the reputation of a business, and perhaps even lead to legal action. More importantly, experience shows that clients do not understand security, and thus steps should be taken to minimize opportunities for clients to make the wrong decision. For these reasons, we believe the bulk of the burden for ensuring security falls on the provider, or server in this case. This then raises the question: do servers deployed in the Internet adhere to current best practices by employing strong cryptography?

This paper characterizes the cryptographic strength of public servers in the Internet running SSL/TLS. We evaluate over 19,000 servers, and present a tool developed for this purpose, the Probing SSL Security Tool (PSST). We use PSST to evaluate which protocols and cryptographic options are supported by the servers, and which are chosen by the servers as a default when presented with several options. We show that a great variety of behavior can be found in the network, with both encouraging and discouraging results. Examples include:

- 85 percent of SSL sites support the SSL 2.0 protocol, even though it has significant security problems. Moreover, a small number of sites support *only* the SSL2 protocol.
- 93 percent of servers support (single) DES, despite the fact that DES is considered susceptible to exhaustive search.
- Many servers support the old export-grade encryption levels, even though US law has changed and these algorithms are considered susceptible to brute-force attacks.
- 765 (almost 4 percent) of the sites use RSA-based authentication with only 512-bit keys, even though RSA has announced that this level of security is insufficient. On the other hand, over 1200 sites use 2048 bits or greater.
- AES is already supported in over 57 percent of sites we probed. Out of these, about 94 percent default to AES when presented with all options (and the vast majority of them use a strong 256 bit key).

We have also run our tool periodically over the last two years, in order to study the evolution of SSL use (and misuse) over time. The overall trend we discovered is a steady, though a little slow, improvement in cryptographic strength of SSL/TLS servers. For example, within the past two years:

- Support of the weak SSL2 protocol has been reduced by over 9 percentage points.
- Support of AES has grown by nearly 16 percentage points.
- Support of weak public key sizes has gone down by nearly 2 percentage points.
- Support of very strong public key sizes has gone up by nearly 2 percentage points.

Thus, our results show that most servers (though not all) support both weak cryptography and strong cryptography, while making the correct choice by default, if given the option. This is in sharp contrast to the situation several years ago, where 20-30 percent of the servers used only weak cryptography (see [27] and our discussion in Section 5). As a concrete example, in 2000 25 percent of servers probed by Murray [27] supported a very weak server key size of at most 512 bits, compared to about 4 percent today.

Our results are also useful in highlighting the most prevalent supported choices among the available options, thereby allowing future efforts on improving performance and enhancing security to focus on the most relevant set of cryptographic algorithms.

Finally, our tool can be useful for regular security compliance testing, especially by large organizations that own multiple servers.

Organization.

The remainder of this paper is organized as follows. In Section 2, we provide some background on the design and history of SSL/TLS. In Section 3 we describe the design of our PSST tool and our probing methodology. Section 4 presents our results in detail, and Section 5 discusses relevant related work. Finally, we conclude and describe some possibilities for future work in Section 6.

2. SSL AND TLS

In this section we provide a brief overview of the mechanisms and history of SSL/TLS.

SSL is designed to facilitate a communication channel between two peers, providing mechanisms for secure key exchange, authentication, encryption, and integrity. It aims to be resilient to man-in-the-middle attacks, eavesdropping, replay attacks, and statistical attacks. While SSL can authenticate two sides of the conversation, in practice it is typically only the server that authenticates itself. The goals for SSL/TLS not only include security, but also interoperability, extensibility, and relative efficiency.

SSL/TLS is composed of two layers: the record layer and the handshake layer. The record layer takes data provided by a higher-layer application, fragments the data into manageable blocks, and performs compression, symmetric-key encryption, and MAC digest generation. The handshake layer performs session establishment and option negotiation, determining the per-session symmetric keys which are used in bulk by the record layer.

SSL/TLS runs "above" the Transmission Control Protocol/Internet Protocol (TCP/IP), which governs the transport and routing of data over the Internet, and "below" higher-level protocols such as the Hypertext Transport Protocol (HTTP), which use TCP/IP to support typical application tasks such as downloading Web pages. SSL lets an SSL-enabled server to authenticate itself to an SSL-enabled client and vice-versa, and allows both machines to establish an encrypted connection. All the while, SSL uses TCP/IP on behalf of the higher-level protocols.

SSL version 2.0 [22] was introduced by Netscape in 1994 for transmitting private data through the Internet (the first version of SSL was never deployed). SSL version 2.0 quickly became the de facto standard for the cryptographic protection of Web traffic. Unfortunately, SSL 2.0 was quickly shown to have several flaws [44]:

- SSL 2.0 is vulnerable to "man-in-the-middle" attacks in which an active attacker can force both the client and the server to use 40-bit encryption.
- SSL 2.0 exclusively uses the MD5 hash function (the insecurity of which will be discussed in Section 4.4).
- SSL 2.0 uses a weak message authentication code (MAC).
- SSL 2.0 feeds padding bytes into the MAC in block cipher modes while leaving the padding-length field unauthenticated. This could allow active attackers to delete bytes from the end of messages.
- SSL 2.0 uses the same key for authentication and encryption, which could lead to problems for certain ciphers.

SSL version 3.0 [20] was introduced in 1996, to improve both the security and the functionality of SSL 2.0. SSL 3.0 not only fixes the security flaws mentioned above, but also reduces the number of network round-trips, lets the server choose the ciphers, supports more complete key exchange and cipher algorithms, and uses separate authentication and encryption keys. Around the same time, Microsoft introduced its own version of SSL, Privacy Communication Technology (PCT), but this never gained as much popularity as SSL 3.0.

The Internet Engineering Task Force (IETF) established the Transport Layer Security (TLS) Working Group in 1996 to come up with a standardized version of SSL and PCT. The standardized protocol [14], imaginatively named TLS version 1.0, is very similar to SSL 3.0. So much so that TLS 1.0 is sometimes referred to as SSL 3.1. The most important difference between the two is that TLS uses the keyed-Hashing for Message Authentication Code (HMAC) algorithm [8] instead of the SSL Message Authentication Code (MAC) algorithm; HMAC produces more secure hashes than MAC. There are other minor differences such as the exclusion of the Fortezza algorithms (which are not open for public review), and the allowance of certificates to go back to an intermediary certificate authority (CA) instead of going all the way back to the root CA. Because of these improvements, the TLS and SSL 3.0 protocols don't fully interoperate, but TLS 1.0 has a mode to fall back to SSL 3.0.

Many Web sites now use SSL to obtain confidential user information, such as credit card numbers and social security numbers. All major Web browsers (Mozilla Firefox, Netscape Navigator, Internet Explorer, Safari, and Opera) support SSL. SSL 3.0 and TLS are believed to be reasonably secure if used properly. SSL 2.0 has fundamental design problems and should not be used for sensitive information.

3. METHODOLOGY

PSST is based on HTTPERF [26], a tool for measuring Web server performance, and utilizes the OpenSSL library [4] for SSL support. HTTPERF normally establishes an SSL connection advertising the entire available suite of protocols from the SSL library, allowing the server to choose which suite to use. We modified HTTPERF's core HTTP engine to make an SSL connection with the target server advertising only *one* cipher suite. If the server supports that suite, the SSL handshake is completed successfully; otherwise, an error code is returned. By connecting to the server multiple times and iterating through the set of available suites, we can determine which suites are supported and which ones are not. Each connection is aborted immediately after the initial SSL handshake, reducing overhead on the network, the SSL server probed, and the PSST client machine itself. We also included an option where all possible protocols and cipher suites were presented so that what choice the server defaults to could be observed.

To maximize our statistical confidence, we clearly wished to probe as many SSL servers as possible. In gathering our list of SSL/TLS servers, we aimed to gather as many addresses as possible while making sure that the largest and most popular sites were included as well. Since a few Web sites attract a significant portion of all the Web traffic, we used the rating sites Alexa [1] and Web100 [5] to gather a list of leading SSL/TLS servers. For breadth we used the list of target Web servers in Padhye and Floyd's study of TCP behavior [34], which in turn came from IRCache, the NLANR Web Caching Project [2]. This resulted in a list of 19,429 SSL servers that we probed with PSST. The PSST software used OpenSSL version 0.9.7b and the Linux operating system version 2.4.24.

Our tests were conducted on February 2005, August 2005, June 2006, and November 2006 from Columbia University in New York City. All our numbers are from the November 2006 test, up to Section 4.7, where we consider the other results in a longitudinal discussion. We discounted any servers that did not respond in all four surveys to keep the data comparable. There were also a handful of IP addresses that provided inconsistent data on SSL/TLS support which are excluded from our analysis. We speculate that these sites use network load balancers (i.e., L4/L7 switches) to distribute traffic across multiple servers with different SSL configurations.

Interestingly, even though our tool could be perceived to be an attack, and we made no effort to conceal our identity, we received no complaints or requests to desist. This testifies to the level of hostile traffic that now pervades the Internet; our probes were considered noise in comparison.

4. RESULTS

In this section we present our results in detail.

4.1 SSL/TLS Protocol Versions

SSL Type	Number	Percentage
SSL 2.0	16587	85.37 %
SSL 3.0	19025	97.92 %
TLS1	19111	98.36 %

Table 1: Overall SSL/TLS Support

We first examine the distribution of the different versions of SSL/TLS supported by the sampled servers. This is shown in Table 1. Most servers support all three versions (while the support of SSL2 should arguably be dropped, given its serious security problems outlined in Section 2). Moreover, a small percentage support *only* SSL 2.0, despite the known problems. The data from Table 1 is broken out in more detail in Table 2, showing that surprising permutations (in fact, all possible permutations) of the available support exist. It seems particularly odd that 0.87 percent of

SSL 2.0	SSL 3.0	TLS	Number	Percentage
Yes	No	No	24	0.12 %
No	Yes	No	146	0.75 %
Yes	Yes	No	148	0.76 %
No	No	Yes	211	1.09 %
Yes	No	Yes	169	0.87 %
No	Yes	Yes	2485	12.79 %
Yes	Yes	Yes	16246	83.62 %

Table 2: Breakdown of SSL/TLS Support

servers support SSL 2.0 and TLS, yet do not support SSL 3.0.

4.2 Key Exchange and Authentication

SSL uses two common protocols for key exchange and two common protocols for authentication, used in the following three combinations. One is to use the Ephemeral Diffie-Hellman (EDH) key exchange algorithm with the Digital Signature Standard (DSS) for authentication. Another is to use the EDH key exchange algorithm with RSA for authentication. The other popular option is to use RSA for both key exchange and authentication.

The security of the EDH key exchange algorithm is equivalent to the intractability of the Decision Diffie-Hellman problem, which is in turn based on (though not known to be equivalent to) the intractability of the discrete logarithm problem. RSA security is equivalent to the intractability of the RSA problem [39], which is in turn based on (though not known to be equivalent to) the intractability of factoring. While solving the discrete logarithm problem will solve the factoring problem, this has no bearing on the relative security of the EDH or RSA key exchange algorithms. Both problems have been subject to intense scrutiny, and are assumed to be hard for now. Based on the best attacks known, equal key sizes for EDH, DSS, and RSA give comparable levels of security [47].

KeyEx + Auth	Number	Percentage
EDH + DSS	4	0.02 %
EDH + RSA	11185	57.57 %
RSA + RSA	19401	99.86 %

Table 3: Key Exchange & Authentication Support

Table 3 shows how these algorithms are used in practice. We see that the RSA key exchange is supported by the vast majority of servers, while the EDH key exchange algorithm is supported by slightly more than half the servers. We can also see that RSA has become the de facto standard for authentication, with only a few servers supporting DSS. It is interesting to note that the 3 of the 4 servers that support DSS use DSS exclusively.

4.3 Key Size

We next turn our attention to the size of the keys used for authentication. The current industry standard is to use 1024-bit key sizes for RSA keys. Note that a 512-bit key was factored in 1999, and that this key size is now considered unsafe [41]. The 1024-bit key size provides comparable strength to an 80-bit symmetric key, and the National Institute of Standards and Technology (NIST) suggests that the 80-bit security level is appropriate for protecting data through the year 2015 [32]. They also suggest that the 112-bit level is appropriate through the year 2035. RSA Laboratories makes a similar recommendation [23, 40]. Specifically, RSA Laboratories currently recommends key sizes of 1024 bits for corporate use, and 2048 bits for extremely valuable keys such as the root key pair used by a certifying authority. They expect that the 80-bit level (corresponding to 1024-bit RSA keys) should be sufficient at least until 2010, and that extremely sensitive information should be encrypted at the 112-bit level (corresponding to 2048-bit RSA key size), which should protect data at least until 2030. Finally, we note that in recent NESSIE recommendations, a minimum of 1536 bits is suggested for RSA signature keys [11], though, according to RSA Laboratories [23], 2048 bits is a better choice.

Key Size	Number	Percentage
512	765	3.94 %
768	275	1.42 %
1024	17166	88.35 %
1280	1	0.01 %
2048	1192	6.14 %
4096	36	0.19 %

Table 4: Public Key Size Support

Table 4 presents the distribution of public key sizes supported by the servers we probed (all but 7 of which only support one size). We can see that over 5 percent of the servers only support weak key lengths that are not considered secure. We note that several odd key sizes appear, such as 1280, and 1568 in previous runs (as shown in Table 18 of Section 4.7), although these are relatively rare.

Until December of 1998, US export laws banned the overseas sales of US software using encryption with keys over 512 bits. This may explain the fact that we still see some servers supporting weak 512-bit key sizes. However, since all but 7 of the servers only support one key size, and thus the servers that support 512-bit keys also tend to have a *maximum* key size of 512 bits.

4.4 Hash Functions

Cryptographic hash functions are vital for the security of SSL/TLS. For a hash function H, given an input x the output $H(x)$ should be easy to compute, but given $H(x)$, finding an inverse should be computationally infeasible to compute. Also, given x and $H(x)$, it should be computationally infeasible to compute a y such that $H(y) = H(x)$. Such hash functions are said to be weakly collision free. If it is computationally infeasible to find any such pair x, y with $H(x) = H(y)$ (as opposed to finding a collision for a particular x), then the hash function is said to be strongly collision free.

The two most widely used hash functions are MD5 [38] and SHA-1 [30]. MD5 was created by Rivest, and has been attacked repeatedly over the years [13, 16, 17]. The fatal blow came in a work-in-progress session at the Crypto 2004 conference, where Wang announced that there existed a family of collisions in MD5 [45]. Even before this announcement, NIST recommended that only SHA-1 (whose output is 160 bits) and the related SHA-256, SHA-384, and SHA-512 al-

gorithms be used. Then in February 2005, Wang, Yin, and Yu announced that they were able to significantly reduce the search space for SHA-1 [46]. Fortunately, SSL 3.0 and TLS use both SHA-1 and MD5 in a redundant fashion for its authentication. They may use MD5 or SHA-1 alone in negotiated cipher suites, but then they are used within the HMAC construction. Since the HMAC construction works by nesting two applications of the hash function with two different keys, breaking it would require finding two related collisions. Thus, the announced attacks cannot be directly applied to HMAC at this stage. On the other hand, SSL 2.0 only uses MD5 without HMAC or SHA-1, and is thus very vulnerable as MD5 is still considered to be much weaker than SHA-1.

MAC	Number	Percentage
MD5	19201	98.83 %
SHA-1	19326	99.47 %

Table 5: Symmetric Key MAC Support

Table 5 shows the MAC algorithm support discovered by PSST. Most servers we surveyed supported both. While some of the servers that only supported MD5 did so because they only ran SSL 2.0, there were 79 sites that used SSL 3.0 and TLS that still only supported MD5 cipher suites. This brings the total number of servers supporting only MD5 to 103.

4.5 Symmetric Encryption

Symmetric encryption is the backbone of secure communication. Once secret keys have been established, the transmitted data is encrypted using a symmetric encryption algorithm. In this section, we examine both the algorithms employed and the key sizes used.

Cipher	Number	Percentage
AES	11107	57.17 %
DES	19168	98.66 %
RC2	17931	92.29 %
RC4	19241	99.03 %

Table 6: Symmetric Key Cipher Support

Table 6 presents an overview of the symmetric key cipher support in the servers we examined (here, the DES category includes 3-DES). As can be seen, nearly all the servers we surveyed support DES, RC2, and RC4. RC4 appears to be the most frequently supported cipher. The Advanced Encryption Standard (AES), which is a fairly new cipher, is already supported by over 57 percent of the servers. We now look at each cipher in more detail.

The Data Encryption Standard [29], invented by IBM at the request of the National Bureau of Standards, has been in use since 1976. DES uses 56-bit keys and has some peculiarities that make it undesirable, but in particular, its short key length has made it vulnerable to brute-force attacks. When the US export laws were in effect, the export key length was artificially reduced to 40 bits, making DES even less secure.

Table 7 shows the DES usage in detail. While the old US export regulations no longer apply, almost 67 percent of the servers surveyed still support these weak keys, and most

Cipher	Number	Percentage
DES-40	12930	66.55 %
DES-56	12102	62.29 %
DES-64	18162	93.48 %
3-DES	18943	97.50 %

Table 7: DES Support

Cipher	Number	Percentage
DES-40	25	0.13 %
DES-56	35	0.18 %
DES-64	165	0.85 %
3-DES	18943	97.50 %

Table 8: Maximum DES Key Strength

servers support DES-64, which is only slightly stronger. Fortunately, fewer than 2 percent of the servers support these weak keys exclusively, as can be seen in Table 8, showing the maximum DES key strengths supported by the servers. For example, Table 8 shows that 165 servers support DES with *at most* 64 bit keys. Prudence would indicate that support of such weak keys should be removed altogether.

Triple DES (3-DES) is a variant of DES where the DES algorithm is applied three times using three different keys. The effective key length is thus 168 bits, which protects against brute force attacks, but 3-DES is slower than other ciphers and is slowly being phased out [33]. Most of the servers we surveyed support 3-DES, using cipher block chaining (CBC).

Cipher	Number	Percentage
RC2-40	17546	90.31 %
RC2-56	7360	37.88 %
RC2-128	16278	83.78 %

Table 9: RC2 Support

Another block cipher is RC2, which was developed by Rivest in 1987. RC2 was originally developed for a 40-bit key, and is a bit slower than other ciphers. For appropriate key lengths (128 bits or greater) it is still considered secure [25].

Table 9 shows the key strengths supported by servers employing RC2. Most servers support 128 bit keys, but a significant number of servers also support the weak 56 and 40 bit key sizes. Table 10 shows the maximum key strength used by servers that support RC2. Over 8 percent of the servers exclusively support weak RC2 keys.

The most commonly supported symmetric key cipher is RC4, which was developed by Rivest in 1987. RC4 is a stream cipher, which encrypts one bit at time. A number of attacks on RC4 have been published, and there are ways of implementing RC4 within a cryptosystem that are completely insecure [19]. Fortunately, it is used correctly in SSL/TLS. As with DES, 40, 56, and 64-bit keys are insecure, and 128-bit keys are considered secure.

Table 11 shows the key strengths supported by servers utilizing RC4. As can be seen, nearly all servers support 128 bit keys, and very many support 40 bit keys. Table

Cipher	Number	Percentage
RC2-40	790	4.07 %
RC2-56	863	4.44 %
RC2-128	16278	83.78 %

Table 10: Maximum RC2 Key Strength

Cipher	Number	Percentage
RC4-40	17827	91.75 %
RC4-56	12173	62.65 %
RC4-64	11030	56.77 %
RC4-128	19154	98.58 %

Table 11: RC4 Support

Cipher	Number	Percentage
RC4-40	48	0.25 %
RC4-56	38	0.20 %
RC4-64	1	0.01 %
RC4-128	19154	98.58 %

Table 12: Maximum RC4 Key Strength

AES-128	AES-256	Number	Percentage
Yes	No	154	0.79 %
No	Yes	10709	55.12 %
Yes	Yes	244	1.26 %

Table 13: AES Support

12 shows the maximum key strength employed by servers supporting RC4. A small fraction support *only* weak key sizes.

The newly favored symmetric encryption algorithm is the Advanced Encryption Standard (AES) [31], since it is the new government standard intended to replace DES. AES was the winning proposal among 15 considered in a 2001 competition. AES is generally considered secure, although there are some concerns about potential attacks [12].

Table 13 shows the distribution of key size support for servers using AES. Even though AES has only been defined for a few years, it is encouraging to see that over 56 percent of the servers support AES-256. Oddly enough, there are only a small number of sites supporting AES with both 128 or 256 bits.

4.6 Default Choices

While most of the servers we surveyed support weak cryptography, we suspect that they do so to accept the largest number of connections, despite the security vulnerabilities. Thus, we probed the servers specifying the full range of cipher suites to see what they chose by default (as there is no excuse for choosing a weaker option than the server supports). Unfortunately, 552 of the 19,429 servers did not respond properly, thus the following numbers are for the remaining 18,887 servers.

Only three servers chose a weaker protocol when a stronger one was available. In particular, they chose SSL 2.0 when SSL 3.0 was available. All servers that supported TLS, chose to use TLS. Similarly, only four (different) servers chose to use a weaker public key size than they supported.

When it came to choosing between the two possible hash functions, 315 servers only supported one of MD5 and SHA-1, and thus had no choice. Of those that supported both, 5469 (29 percent of all servers) chose MD5.

As shown in Table 14, none of the servers chose RC2 from the available symmetric encryption algorithms. However, 657 servers that support AES (namely about 6 percent of those that support AES) did not choose AES by default.

The combined cipher suite choices are shown in Table 15. AES-256 with SHA-1 was the most popular choice, while RC4-128 with MD5 and 3-DES with SHA-1 were second and third.

4.7 Changes in Support over Time

In order to study the evolution of SSL use (and misuse) over time, we conducted tests on February and August of 2005 and June and November of 2006.

As seen in Table 16, there are promising trends in SSL version support. During this time period, the number of servers supporting SSL 2.0 was reduced by over 9 percentage points.

We also found similarly heartening news with the change in symmetric cipher support (Table 17) as the number of servers supporting AES grew by nearly 16 percentage points.

Finally, we also note that small public key sizes are slowly losing support (Table 18), while larger ones, in particular 2048-bit keys, are supported by a couple more servers. However, while this is a change in the right direction, it is too small for comfort: The fraction of servers supporting weak key sizes is still significant at about 5 percent (compared to about 7 percent in our earlier runs).

Overall, while the trend is certainly positive, it may be too slow. For example, continuing at the same rate, it will take quite a few years until the support of SSL2 is mostly abolished.

5. RELATED WORK

In 2001, Murray presented a survey of SSL servers [27]. Murray's survey generally covered similar issues as in this paper, though in less detail. In addition, it also considered whether or not a server's certificate was expired or self-signed.

Murray defined weak servers to be those that supported at least one of the following flaws: 1) only supports SSL 2.0; 2) only supports symmetric encryption using keys with at most 56 bits; 3) only supports certificate key sizes of at most 512 bits; 4) uses an expired or self-signed certificate. Murray defined strong servers to be those that supported all of the following properties: 1) supports SSL 3.0 or TLS (can support SSL 2.0); 2) supports symmetric encryption using keys with at least 64 bits (can support 40-bit keys); 3) supports certificate key sizes of at least 1024 bits (can support smaller certificate keys). Finally, the rest of the servers are considered to have a medium level of security. Note that the levels of security defined by Murray would not be considered valid today, as new developments in security have rendered several cryptosystems insecure that were previously considered secure. Moreover, servers that support both strong and weak cryptography are called "strong" here, while today to provide strong security a server should

Cipher	Servers	Percentage
RC2	0	0.00 %
RC4	5882	31.16 %
DES	2854	15.12 %
AES	10141	53.72 %

Table 14: Default Symmetric Encryption

Cipher suite	Servers	Percentage
AES-256 SHA-1	10135	53.69 %
RC4-128 MD5	5611	29.72 %
3-DES SHA-1	2837	15.02 %
RC4-128 SHA-1	259	1.37 %
3-DES MD5	12	0.06 %
RC4-40 MD5	9	0.05 %
AES-128 SHA-1	6	0.03 %
RC4-56 SHA-1	3	0.02 %
DES-64 SHA-1	3	0.02 %
DES-56 SHA-1	2	0.01 %

Table 15: Default Cipher Suites

arguably support only reasonably strong cryptography (and not support any weak options), as previously discussed.

Table 19 shows a comparison between Murray's 2000 and 2001 surveys and our 2006 survey, using Murray's definitions to allow for comparison. The numbers are not completely comparable, as Murray surveyed 8081 servers while we examined 19,429. Another difference is that we did not consider expired or self-signed certificates. The dramatic trend, however, is the reduction in the percentage of weak servers, and the rise in the percentage of strong servers. This trend is exhibited when we compare several specific aspects directly (Table 20). This is consistent with our results (Section 4.7) indicating the same positive trend (though less dramatically) over the last 2 years.

Scanning Approaches to Security

Scanning large numbers of servers is a common approach to testing server security. Nmap [3] is a well-known security scanning tool. Provos and Honeyman [35] scanned about 2300 SSH servers to determine which version of SSH was being used. Similarly, Rescorla [37] probed 891 SSL servers that used the OpenSSL library to see if they were vulnerable to a bug in OpenSSL, and whether that vulnerability was being addressed. Our study differs from the above in that it focuses on cryptographic security rather than software vulnerabilities.

SSL/TLS Performance Studies

Researchers have also recently studied the performance of SSL servers [6, 10, 36]. In these studies, an SSL site is set up in a lab and evaluated. Each of these studies made certain choices about the algorithms used and the key sizes employed. For example, Apostolopoulos et al. [6] varied the RSA key size between 512 bits and 1024 bits, and studied the DES, 3-DES, RC4, MD5 and SHA-1 algorithms. Coarfa et al. [10] used RSA with a key size of 1024, RC4 with 128-bit keys and MD5. Rescorla varied RSA key sizes from 512 bits to 2048 bits, while studying RC4, DES, 3-DES, SHA-1, and MD5. While these studies were appropriate for their times, our results suggest that any future performance studies should likely look at stronger key sizes (e.g., 2048 for RSA) and current best practice algorithms (e.g., AES and SHA-1).

SSL type	02/2005	08/2005	06/2006	11/2006
SSL2	94.49 %	93.23 %	87.95 %	85.37 %
SSL3	97.96 %	98.30 %	98.16 %	97.92 %
TLS1	97.51 %	98.32 %	98.28 %	98.36 %

Table 16: Changes in SSL Support Over Time

Cipher	02/2005	08/2005	06/2006	11/2006
AES	41.26 %	48.29 %	55.18 %	57.17 %
DES	99.13 %	99.28 %	98.81 %	98.66 %
Weak DES	97.32 %	97.00 %	94.63 %	93.48 %
RC2	96.52 %	96.20 %	93.63 %	92.29 %
RC4	99.50 %	99.57 %	99.18 %	99.03 %

Table 17: Change in Cipher Support Over Time

Attacks on SSL/TLS

It should be noted that there has been recent work ([9, 24, 7, 43, 28]) attacking SSL/TLS as well. Most of these attacks, however, are not directed at the various cryptographic algorithms used by SSL/TLS, but rather in their implementation or against the peculiarities of SSL/TLS itself. These attacks can often be prevented by applying particular patches to the SSL/TLS software, but we do not have a reliable way of measuring these preventive measures yet.

6. CONCLUSIONS AND FUTURE WORK

We have presented the probing SSL security tool (PSST), and used it to analyze server security on the Internet, evaluating over 19,000 servers. Given the volume of sensitive Internet-based transactions which utilize SSL/TLS servers, understanding what security measures are in place today to protect confidential data, and to what degree, is extremely important. This is particularly true given the rapid pace at which our understanding of cryptography evolves, and vulnerabilities are discovered in protocols previously believed to be secure. For example, SHA-1 was considered to be secure and was even recommended by NIST when we first ran our experiments in February 2005. By the time we ran our experiments in August 2005, SHA-1 was found to have serious flaws. Our results show that there is great variability in the level of cryptographic support in Internet servers running the SSL/TLS protocol, including a significant portion of the servers which support weak, broken, or outdated cryptography.

In addition to its utility as a security tool, the results generated using PSST would be useful for SSL/TLS server performance benchmarking. Any effort to improve the performance of the cryptographic components of SSL/TLS could be focused on the most commonly used algorithms.

We see several areas for future work:

- *Expired and self-signed certificates.* Murray's study from 2001 tested for expired and self-signed certificates; we plan to extend PSST to test those properties on our set of servers.

Key Size	02/2005	08/2005	06/2006	11/2006
512	5.01 %	5.32 %	4.17 %	3.94 %
768	1.93 %	1.84 %	1.54 %	1.42 %
1024	88.46 %	87.80 %	88.33 %	88.35 %
1048	0.01 %	0.01 %	0.00 %	0.00 %
1280	0.00 %	0.00 %	0.01 %	0.01 %
1536	0.01 %	0.00 %	0.00 %	0.00 %
1568	0.01 %	0.01 %	0.01 %	0.00 %
2048	4.51 %	4.96 %	5.91 %	6.14 %
4096	0.12 %	0.15 %	0.17 %	0.19 %

Table 18: Change in Public Key Size Support

	2000	2001	11/2006
Weak Servers	31 %	23 %	4.07 %
Medium Servers	10 %	5 %	1.51 %
"Strong" Servers	57 %	51 %	94.42 %

Table 19: Overall Server Security

- *Measuring even larger numbers of servers.* While our list of servers is large, CPU power has increased enough that a large fraction of the available IPV4 address space could be probed by randomly choosing IP addresses to test.
- *Measuring SSL VPNs.* Since the use of SSL for Virtual Private Networks has been increasing recently, this is also an area for further investigation.
- *SSH Servers.* While authentication is handled differently in SSH than in SSL, many of the cipher suites utilized are the same. Since SSH is used as the entry point into many organizations, its cryptographic strength should be ensured.
- *Client-Side Studies.* This work has examined the available cryptographic strength on the server side. Clearly, end-to-end security depends on the support of both sides of the conversation. It would thus be interesting to evaluate what is being used by clients on the Internet. One way to do this would be to monitor the packets into and out of a high-volume SSL server (or cluster of servers).

We believe PSST can be extended in a relatively straightforward way to study several of these issues.

Acknowledgments

We thank Noel Codella for participating in early stages of this work. Thanks to Ran Canetti, Angelos Keromytis, and Patrick McDaniel for discussions and constructive comments on earlier drafts of this paper. Thanks also to Tomer Malkin Nahum for participating in several brainstorming sessions.

7. REFERENCES

[1] Alexa Web Search - Top 500. `http://www.alexa.com/site/ds/top_500`.
[2] IRCache. `http://www.ircache.net`.
[3] Nmap. `http://www.insecure.org/nmap/`.
[4] The OpenSSL project. `http://www.openssl.org`.

	2000	2001	11/2006
Server Key $\leq$ 512 bits	25.0 %	17.0 %	3.94 %
SSL 2.0 only	0.4 %	1.4 %	0.12 %
$\leq$ 56-bit sym. enc. only	9.0 %	6.0 %	0.13 %

Table 20: Weak Server Features

[5] Web100. `http://www.web100.com`.
[6] George Apostolopoulos, Vinod Peris, and Debanjan Saha. Transport layer security: How much does it really cost? In *IEEE InfoCom*, New York, NY, March 1999.
[7] Gregory V. Bard. The vulnerability of SSL to chosen plaintext attack. Cryptology ePrint Archive, Report 2004/111, 2004. `http://eprint.iacr.org/`.
[8] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In N. Koblitz, editor, *Advances in Cryptology — CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 534–545. Springer-Verlag, 1996.
[9] Dan Boneh and David Brumley. Remote timing attacks are practical. In *The 12th USENIX Security Symposium*, August 2003.
[10] Cristian Coarfa, Peter Druschel, and Dan S. Wallach. Performance analysis of TLS Web servers. *ACM Transactions on Computer Systems*, 24(1), February 2006.
[11] NESSIE Consortium. Portfolio of recommended cryptographic primitives. Internet draft, February 2003. `http://www.cryptonessie.org/`.
[12] Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In Yuliang Zheng, editor, *Advances in Cryptology — ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer-Verlag, 2002.
[13] B. den Boer and A. Bosselaers. Collisions for the compression function of MD5. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT 1993*, volume 470 of *Lecture Notes in Computer Science*, pages 293–304. Springer-Verlag, 1994.
[14] T. Dierks and C. Allen. The TLS protocol, version 1.0, January 1999. RFC-2246.
[15] Tim Dierks and Eric Rescorla. The TLS protocol, version 1.1, June 2005. Internet Draft, `http://www.ietf.org/internet-drafts/draft-ietf-tls-rfc2246-bis-13.txt`, expires December 2005.
[16] Hans Dobbertin. Cryptanalysis of MD5 compress. In *Fast Software Encryption*, pages 53–69, 1996.
[17] Hans Dobbertin. The status of MD5 after a recent attack. *CryptoBytes*, 2(2), 1996.
[18] N. Ferguson and B. Schneier. *Practical Cryptography*. Wiley Publishing, Inc., 2003.
[19] Scott Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the key scheduling algorithm of RC4. In *Selected Areas in Cryptography*, volume 2259 of *Lecture Notes in Computer Science*, pages 1–24, 2001.
[20] Alan O. Freier, Philip Karlton, and Paul C. Kocher.

The SSL protocol version 3.0. Internet draft, Netscape Communications, November 1996. http://wp.netscape.com/eng/ssl3/ssl-toc.html.

[21] Eu-Jin Goh. SSL sniffer. http://crypto.stanford.edu/~eujin/sslsniffer/index.html.

[22] Kipp E. B. Hickman. The SSL protocol. Internet draft, Netscape Communications, February 1995. http://wp.netscape.com/eng/security/SSL_2.html.

[23] Burt Kaliski. TWIRL and RSA key size. Internet draft, RSA Laboratories, May 2003. http://www.rsasecurity.com/rsalabs/node.asp?id=2004.

[24] Vlastimil Klima, Ondrej Pokorny, and Tomas Rosa. Attacking RSA-based sessions in SSL/TLS. Cryptology ePrint Archive, Report 2003/052, 2003. http://eprint.iacr.org/.

[25] Lars R. Knudsen, Vincent Rijmen, Ronald L. Rivest, and M. J. B. Robshaw. On the design and security of RC2. In *FSE '98: Proceedings of the 5th International Workshop on Fast Software Encryption*, pages 206–221. Springer-Verlag, 1998.

[26] D. Mosberger and T. Jin. httperf – a tool for measuring Webserver performance. In *Proceedings of the ACM SIGMETRICS Workshop on Internet Server Performance (WISP)*, pages 69–67, Madison, WI, June 1998.

[27] Eric Murray. Changes in deployment of cryptography. Invited talk, USENIX Security Symposium 2001. http://www.usenix.org/events/sec01/murray/index.htm, July 2001.

[28] Netcraft News. Vulnerable versions of OpenSSL apparently still widely deployed on commerce sites. http://news.netcraft.com/archives/2003/11/03/vulnerable_versions_of_ope%nssl_apparently_still_widely_deployed_on_commerce_sites.html.

[29] NIST. Data encryption standard DES, December 1993. http://www.itl.nist.gov/fipspubs/fip46-2.htm.

[30] NIST. Secure hash standard, federal information processing standards publication 180-1, April 1995. http://www.itl.nist.gov/fipspubs/fip180-1.htm.

[31] NIST. Advanced encryption standard (AES), federal information processing standards publication 197, November 2001. http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[32] NIST. Special publication 800-57: Recommendation for key management. part 1: General guideline, January 2003. http://csrc.nist.gov/CryptoToolkit/kms/guideline-1-Jan03.pdf.

[33] NIST. Announcing proposed withdrawal of federal information processing standard (FIPS) for the data encryption standard (DES) and request for comments, July 2004. http://edocket.access.gpo.gov/2004/04-16894.htm.

[34] Jitendra Padhye and Sally Floyd. On inferring TCP behavior. In *ACM SIGCOMM Symposium on Communications Architectures and Protocols*, San Diego, CA, August 2002.

[35] Niels Provos and Peter Honeyman. ScanSSH: Scanning the Internet for SSH servers. In *USENIX Large Installation System Administration Conference (LISA)*, pages 25–30, 2001.

[36] Eric Rescorla. *SSL and TLS*. Addison Wesley, 2000.

[37] Eric Rescorla. Security holes... who cares? In *Proceedings of the 12th USENIX Security Symposium*, pages 75–90, August 2003.

[38] Ron Rivest. The MD5 message digest algorithm, April 1992. RFC-1321.

[39] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[40] RSA Laboratories. How large a key should be used in the RSA cryptosystem? Internet draft, RSA Crypto FAQ. http://www.rsasecurity.com/rsalabs/node.asp?id=2218.

[41] RSA Laboratories. RSA crypto challenge sets new security benchmark - 512-bit public key factored by international team of researchers, August 1999.

[42] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, 1994.

[43] S. Vaudenay. Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS, ... In *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 534–545. Springer-Verlag, 2002.

[44] David Wagner and Bruce Schneier. Analysis of the SSL 3.0 protocol. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce*, pages 29–40, Oakland, CA, November 1996. http://www.cs.berkeley.edu/~daw/papers/ssl3.0.ps.

[45] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu. Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD, 2004. Manuscript. Available from eprint.iacr.org.

[46] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In *Advances in Cryptology — CRYPTO 2005*, Lecture Notes in Computer Science. Springer-Verlag, 2005.

[47] Michael J. Wiener. Performance comparison of public-key cryptosystems. *CryptoBytes*, 4(1), 1998. http://www.rsasecurity.com/rsalabs/node.asp?id=2004.