SECURE COMMUNICATIONS

White Paper

high security remote access

# Debunking the Myths of SSL VPN Security

By Rainer Enders & Clint Stewart

As of July 2011

**N**etwork
**C**ommunications
**P**roducts engineering

## USA:

NCP engineering, Inc.
444 Castro Street, Suite 711
Mountain View, CA 94041
Tel.: +1 (650) 316-6273
Fax: +1 (650) 251-4155

## Germany:

NCP engineering GmbH
Dombuehler Str. 2
D-90449 Nuremberg
Tel.: +49 (911) 9968-0
Fax: +49 (911) 9968-299

## Internet

http://www.ncp-e.com

## Email

info@ncp-e.com

## Support

NCP offers support for all international users by means of Fax and Email.

## Email Addresses

| | |
|---|---|
| helpdesk@ncp-e.com | (English) |
| support@ncp-e.com | (German) |

## Fax

| | |
|---|---|
| +1 (650) 251-4155 | (USA) |
| +49 (911) 9968-458 | (Europe) |

When submitting a support request, please include the following information:

- exact product name
- serial number
- version number
- an accurate description of your problem
- any error message(s)

## Contents

# Debunking the Myths of SSL VPN Security

By Rainer Enders & Clint Stewart

This paper outlines a clear analysis of the flaws, issues and security problems of secure sockets layer (SSL) virtual private networks (VPNs), beginning at the source of the technology, and detailing 10 prevailing myths of security assurance promised by vendors and assumed as safe by VPN users. The intent is to detail the security myths and dispel dangerous market hype, which is leading to over-reliance on the protocol alone. Both authors have spent decades developing and using VPN technologies, and cast their experiences in this paper in an attempt to help companies avoid damaging network security miss-steps. Experience has taught the authors that a mixed VPN environment is ideal and one-size does not fit all.

## Relying on a Popular Target

Today's users of VPNs continue to rely on vendor solutions using the OpenSSL Toolkit, which has been hacked 44 times over an 8-year period since 2002 [1]. That's an average of 5.5 times per year, or more than once per quarter. The Internet connection via web browser is statistically one of the most vulnerable points of attack as indicated by a recent study from Qualys, Inc.  [24]. A sample web browser exploit is provided in Appendix A to show how one of the most critical flaws in the most popular web browser is downplayed by the vendor.
Some of the highest profile security breaches occur using the SSL VPN technology. Why is that? Do users implement the technology incorrectly? Or, is it simply not as good as all the marketing hype makes it out to be? CIOs and network security architects should ask themselves: "Am I trying to force blood from a stone because management of SSL is simpler than other VPN protocols?"

## A Note on Mobile Devices

While mobile phones and other handheld devices are mistakenly considered relatively safe, this misnomer does not qualify as an SSL myth. It does, however, require addressing, as the consumerization of IT forces CIOs and network security architects to integrate these devices into the VPN structure.
Beyond the recent consumer-oriented, high profile hacks to celebrity address books, the danger to enterprises is being laid bare in a more subtle manor.  In May 2011, Juniper Networks published a study that found risks to mobile phone security at an all time high, and cited a 400% rise in malware against the Android, for example. Interestingly, one of the key recommendations of the study offers the use of SSL VPNs as a solution [22]. However, in 2008, critical mobile SSL VPN vulnerabilities were discovered by Christophe Vandeplas [23], provided in Appendix B, as a laboratory example of the man-in-the-middle (MITM) exploit.
In mid-March 2011, after Comodo issued nine fraudulent certificates affecting several domains, Microsoft issued updates for its PC platforms to fix the vulnerabilities; however, the company is still working on a patch for Windows Phone 7 at the time of this paper's writing [21]. More details surrounding this attack are outlined in Myth 6. Clearly, the priority is not currently on the mobile platform, and this is a danger.

## Considering Skype's Use of SSL

Employee's use of Skype - whether for personal or business use - is exploding. The service reported an average of 145 million connected users per month in the fourth quarter of 2010, *before* the Facebook rollout of Skype-powered group video chat service to 750 million users worldwide by August 2011 [31], or the Verizon 4G LTE mobile broadband network deal to integrate Skype on all phones takes effect this year [28].

Skype uses SSL and Advanced Encryption Standard (AES) hashed with the RSA security algorithm for its public key cryptography [27]. The details of how this combination is dismantled as a security model are explained in Myth 8 and Myth 2. Suffice it to say that Skype is not nearly as secure as people think. As we will learn in Myth 5, the public key cryptography is susceptible to the infamous MITM attack. As a result of these revelations, Skype and Facebook users need to be very concerned about what they disclose in their personal and business conversations.

The net effect of attacks against the trust model for mobile certificates and use of Skype should leave CIOs and network security architects uneasy about SSL and using it to secure mobile devices and Skype within their network ecosystems. Employees are using them, and policies restricting mobile devices and Skype use are no longer effective or logical.

## Leading Assumptions and Myths

**Myth 1: SSL VPN is clientless.**
Clientless SSL VPN products from multiple vendors operate in a way that breaks fundamental browser security mechanisms, according to a warning from the U.S. Computer Emergency Response Team (US-CERT). This security problem, discussed first in 2006, let an attacker use these devices to bypass authentication or conduct other web-based attacks. Clientless VPN products from Juniper Networks, Cisco Systems, SonicWall and SafeNet have been confirmed as vulnerable, according the US-CERT.

By convincing a user to view a specially crafted web page, a remote attacker is able to obtain VPN session tokens and read or modify content, including cookies, script or HTML content, from any site accessed through the clientless SSL VPN. This effectively eliminates same origin policy restrictions in all browsers. For example, the attacker is able to capture keystrokes, while a user is interacting with a web page.

Because all content runs at the privilege level of the web VPN domain, mechanisms to provide domain-based content restrictions, such as Internet Explorer security zones and the Firefox add-on NoScript, can be bypassed. Clientless VPNs offer browser-based access to corporate intranets, but US-CERT warns that they subvert the same-origin policy that prevents certain active content (JavaScript, etc.) from accessing to changing another site's data.

According to the advisory, "many clientless SSL VPN products retrieve content from different sites, then present that content as coming from the SSL VPN, effectively circumventing browser same-origin restrictions." In one attack scenario, a malicious hacker can create a web page that obfuscates the document.cookie element in such a way as to avoid being rewritten by the web VPN, and then the document.cookie object in the returned page will represent all of the user's cookies for the web VPN domain.

US-CERT reported this document.cookie typically included the web VPN session ID cookie itself and all globally unique cookies set by sites requested through the web VPN. Continued the report, "the attacker may then use these cookies to hijack the user's VPN session and all other sessions accessed through the web VPN that rely on cookies for session identification."

Additionally, an attacker could construct a page with two frames: one hidden and one that displays a legitimate intranet site. The hidden frame could log all keys pressed in the second, benign frame and submit these key presses as parameters to an XMLHttpRequest GET to the attacker's site, rewritten in web VPN syntax. The bigger problem here is, that there is no solution to this even in mid-2011. Depending on their specific configuration and location in the network, these devices may be impossible to operate securely, the group said [Naraine, 2009].

The most important problem is that web-based VPNs break the customary browser security model that relies on domain name separation for the purpose of restricting access to cookies and other objects. Browsers generally allow foo.com to interact with its own cookies or windows, but prevent the site from accessing resources related to bar.com. Yet, through SSL VPN, they all may look the same:

```
https://webvpn.foocorp.com/http/0/foo.com/serious_work
https://webvpn.foocorp.com/http/0/bar.com/fun_and_games
```

Because of this design, all pages displayed through a clientless web VPN interface are lumped together. Whenever a page or just an HTML fragment that can be controlled by the attacker is displayed by *any* of the applications behind clientless web VPNs, JavaScript can access:

- Clientless VPN session cookies, which can then be passed to the attacker; this is equivalent to the attacker obtaining access to all protected systems and compromising clientless VPNs altogether. Associating the cookie with clients' IP addresses could mitigate the threat, but such an approach is not always implemented, and is impractical with AOL and other ISPs.

- Cookies set by other applications, if passed to the browser (as some clientless SSL VPNs do), are separated by the use of the "path" parameter alone, which does not necessarily establish a browser security domain boundary. This is equivalent to the attacker obtaining user credentials to these other web applications [Zalewski, 2006].

Another clientless VPN study shows that, as early as 2003, researchers at Dartmouth College presented the attack phenomenon of Keyjacking. In theory, Public Key Infrastructure (PKI) can provide a flexible and strong way to authenticate users in distributed information systems. In practice, much is being invested in realizing this vision via tools such as client-side SSL and browser-based key stores. Exploring this vision, the researchers demonstrated that browsers use personal certificates to authenticate requests that the user neither knew of, nor approved. They found that, in some scenarios, direct migration from password-based systems to clientless SSL actually made things worse.

The researchers also demonstrated the easy permeability of these key stores, including new attack vectors on medium- and high-security browser and operating system keys. They suggested some short-term countermeasures in the study, but their findings concluded that against this background, it is not clear that the current clientless SSL infrastructure can achieve the PKI vision. They suggested that a fundamental rethinking of the trust, usage and storage model might result in more effective tools for building a PKI [12].

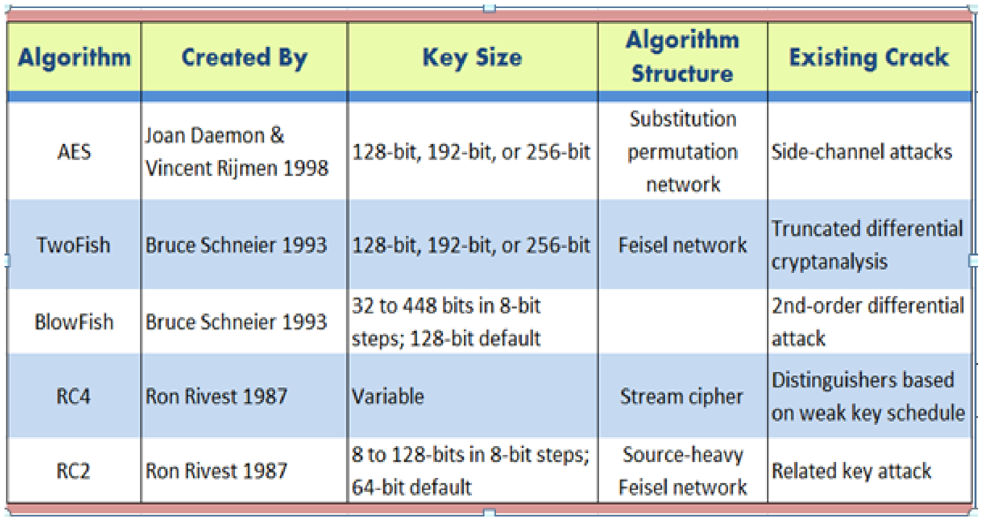
**Myth 2: Online banking via SSL sessions is secure.**
A very prevalent use of SSL occurs at the web browser, often deployed by companies who need to secure sensitive information transfer from customers or partners. A recent attack targeted Citi-Group's 21 million customers and resulted in a 1% success rate—210,000 users. This exploit focused on weaknesses in the SSL connection and issues with the web browser itself [32].

AES, developed by Defense Advanced Research Projects Agency (DARPA), is widely accepted as the strongest encryption on the planet. Recently, Swiss researchers published a memo describing a way to gather information about the data transmitted over an SSL channel using vulnerability in SSL implementations of block ciphers, such as AES.

In certain circumstances, it is possible to use this method to decrypt some of the data in the messages, including encrypted passwords. This vulnerability is due to the way error handling is implemented in applications that use the cipher-block chaining mode, such as AES in SSL. The method works by measuring timing differences in the way an application reacts to new messages created by an attacker. At the time of this finding, the implementation of SSL in Mozilla's network security services version 2.8 libraries was susceptible to this method of attack against AES. Security Issues persist with the Network Security Services (NSS), a set of libraries designed to support cross-platform development of security-enabled client and server applications, including x.509 digital certificates used in many web browser applications [5].
Mozilla's mitigating strategy for this flaw is to use the RC4 cipher algorithm, developed by RSA Security. However, this strategy is hollow because the RC4 algorithm was cracked 10 years ago using the secure shell protocol and two file formats from Microsoft [i.e., .doc and .xls] [13, 14]. Fundamental lessons can be gained observing this quote from Bruce Schneier, a renowned master cryptanalyst who worked for the CIA over many years:

"One of the most important rules of stream ciphers is to never use the same key stream to encrypt two different documents. If someone does, you can break the encryption by XORing the two cipher text streams together. The key stream drops out, and you end up with plaintext XORed with plaintext—and you can easily recover the two plaintexts using letter frequency analysis and other basic techniques" [Schneier, 2005].

Comparison of Most Popular Ciphers



| Algorithm | Created By | Key Size | Algorithm Structure | Existing Crack |
|---|---|---|---|---|
| AES | Joan Daemon & Vincent Rijmen 1998 | 128-bit, 192-bit, or 256-bit | Substitution permutation network | Side-channel attacks |
| TwoFish | Bruce Schneier 1993 | 128-bit, 192-bit, or 256-bit | Feisel network | Truncated differential cryptanalysis |
| BlowFish | Bruce Schneier 1993 | 32 to 448 bits in 8-bit steps; 128-bit default | | 2nd-order differential attack |
| RC4 | Ron Rivest 1987 | Variable | Stream cipher | Distinguishers based on weak key schedule |
| RC2 | Ron Rivest 1987 | 8 to 128-bits in 8-bit steps; 64-bit default | Source-heavy Feisel network | Related key attack |

**Source: Journal of Security Engineering - 2009**

There are numerous ways an attacker can mount a successful attack against the web browser—too many to enumerate in this paper. If you care for details, the Open Web Application Security Project (OWASP) is a good resource.

**Myth 3: HTTPS is a secure pipe.**

As mentioned earlier, in the case of Myth 2, the financial transactions of hundreds of thousands of CitiGroup customers were not secure. The assumption is that this was a secure pipe because the user is connected directly to the Internet through a user's Internet Service Provider's (ISP) network. The ISP provides its customers with a private network address, for example:

```
Windows IP Configuration
Wireless LAN adapter Wireless Network Connection 2:
  Connection-specific DNS Suffix  . : domain.actdsltmp
  Link-local IPv6 Address . . . . . : fe80::4054:f335:33ff:8e46%12
  IPv4 Address. . . . . . . . . . . : 192.168.0.4
  Subnet Mask . . . . . . . . . . . : 255.255.255.0
  Default Gateway . . . . . . . . . : 192.168.0.1
Ethernet adapter Local Area Connection:
  Media State . . . . . . . . . . . : Media disconnected
  Connection-specific DNS Suffix  . :
Tunnel adapter isatap.domain.actdsltmp:
  Media State . . . . . . . . . . . : Media disconnected
  Connection-specific DNS Suffix  . : domain.actdsltmp
Tunnel adapter Teredo Tunneling Pseudo-Interface:
  Connection-specific DNS Suffix  . :
  IPv6 Address. . . . . . . . . . . : 2001:0:4137:9e76:105e:1bef:9e8f:459b
  Link-local IPv6 Address . . . . . : fe80::105e:1bef:9e8f:459b%13
  Default Gateway . . . . . . . . . : ::
Tunnel adapter isatap.{84412028-F34C-4A25-8B20-EE41BB07A738}:
  Media State . . . . . . . . . . . : Media disconnected
  Connection-specific DNS Suffix  . :
```

In this example, we see private IP addresses used (an IPv4 address entry), which is common in internal networks. Network address translator (NAT) gateways are usually used to provide Internet connectivity. Because of the Windows firewall and anti-virus software, the user believes he or she is using a secure connection (or pipe) to the Internet. Notably, the wireless DSL connection in the listing above shows IPv6 tunneling using a Teredo Tunneling Pseudo-Interface. According to the Internet Society, NAT has traditionally been used to forward incoming traffic to specific IPv4 addresses, and this has, to some extent, reduced the number of devices in a local area network accessible directly from outside.

However, IPv6 allocates addresses that are theoretically all routable through the Internet. This might constitute a security threat. The emphasis of setting up NAT correctly in the router is, therefore, now shifting to setting up the firewall correctly. It is also worth noting that IPv6 tunneling allows a path through IPv4 firewalls. Attackers might be able to penetrate a network if the IPv6 firewall is not set up properly [29]. Certainly, this particular nuance of IPv6 tunneling is not obvious to anyone lacking advanced network engineering background.

Also worth noting, when using IPv6 tunneling, is the introduction of the mobility function, which raises the complexity of verifiable security as if elevated on a non-linear scale. Mobility raises a considerable amount of concern when considering security because this function uses two types of addresses: the real address and the mobile address. The first is a typical IPv6 address contained in an extension header. The second is a temporary address contained in the IP header. Because of these characteristics—somewhat more complicated if we consider wireless mobility—the temporary component of a mobile node address could be exposed to spoofing attacks on the home agent. Mobility requires special security measures, and network administrators must be fully aware of them [30].

Web application developers in enterprise networks are especially trusting of this paradigm of secure Internet connectivity. They make use of the OpenSSL Toolkit provided by their integrated development environment (IDE) and the deployment environment for their web applications, which is typically some flavor of an application server environment, such as JBoss, Websphere, WebLogic, etc. The mainstream IDEs provide a Remote System Explorer (RSE), which is a perspective and toolkit that allows developers to connect and work with a variety of remote systems. With the predefined plug-ins, they can look at remote file systems, transfer files between hosts, do remote search, execute commands and work with processes [16]. Application developers trust that SSL is a secure communications facility that encrypts all communications between a client and a target system.

Unfortunately, the prevailing mentality that security is someone else's responsibility (see Myth 10) is in play here, and it is easier for application developers to say to themselves, "I am using a secure pipe to perform my development activities, and therefore, my end is secure," when in fact, insecurity has been passed down the line.

**Myth 4: One-way certificate authentication of a SOA web service is secure because it uses HTTPS.**

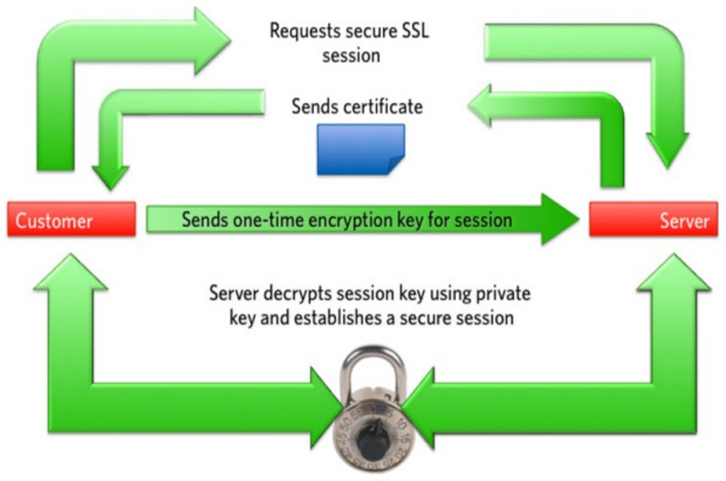
SOA’s simplicity lies in its use of descriptor-based definitions of application transactions that can be articulated directly from a business process into a service description with associated attributes in the description correlating to the procedures of the business process and sub-process threads.

Because SOA uses web-based technology, it is convenient to use SSL as the mechanism to secure user sessions. SSL can be used to tunnel any application-level protocol, which would be otherwise run on top of TCP in the communications protocol stack. The most common use nowadays is to secure the HTTP communication vis-à-vis HTTPS (see Myth 3) in which case the user’s browser is not authenticated; only the server side is authenticated by SSL. This is known as one-way SSL authentication. The MITM attack has been successful against this authentication scheme for at least 10 years [10].

**Myth 5: Two-way certificate exchange between a SOA web service and a client can always be trusted.**

SSL achieves its security by using certificates to authenticate each side of a connection made between two parties—a web server and a client (usually a web browser)—which are based on public key cryptography. The SSL protocol assumes that, if a public key can be used to decrypt a piece of information, then it's all but certain that the information was originally encrypted with the corresponding private key.

When initiating a two-way SSL session, the client will check that the SOA web service certificate is valid and signed by a trusted entity. The server running the web service publishes a certificate—a little chunk of data that includes a company name, a public key and some other bits and pieces—and when the client connects to the server, the client sends the server some information encrypted using the public key from the certificate. The server then decrypts this using its private key. Once the connection is established, all information during that session is encrypted with this information.



Requests secure SSL session
Sends certificate
Customer
Sends one-time encryption key for session
Server
Server decrypts session key using private key and establishes a secure session

Source: Ars Technica - 2011

Since only the server knows the private key—and hence, only the server can decrypt the information encrypted with the public key—this allows the client to prove that it is communicating with the rightful owner of the certificate. Herein lies the flaw.

To defeat this setup, the MITM just has to do a little bit more work; it has to create its own certificate with a private / public key pair and sit between the client and server—acting as server to the client and client to the server—and listen in on everything sent between the two [11].

**Myth 6: Using trusted certificates from a certificate authority (CA) is airtight.**
Certificates used to authenticate an SSL connection allow for the certain identification of each party and for the negotiation of an encrypted channel for communication. The certificates themselves are files whose alteration can be easily detected and whose origin are verified by a trusted certificate authority, such as Comodo or VeriSign.

The web application developers use this trusted certificates model extensively when building their applications. The problem is that the CA can be spoofed. The Electronic Frontier Foundation staff technologist Peter Eckersley has a good, in-depth analysis of the revelation that Iranian hackers acquired fraudulent SSL certificates for Google, Yahoo, Mozilla and others by spoofing Comodo [9]. CAs sell digitally signed certificates that browsers use to verify their network connections; with these spoofed certificates, the hackers could undetectably impersonate Yahoo and Google (allowing them to read email even if it was being read over a secure connection); the Mozilla certificate would allow them to slip malicious spyware onto the computer of anyone installing a Firefox plug-in [8, 9]. HTTPS and other SSL-using protocols (secure SMTP, POP, IMAP, Jabber and many, many others all build on SSL) still offer protection against casual snoopers; they'll protect against the use of Firesheep in a hipster café just fine. But the trust and security promises that are implicit in the use of SSL, and which are depended on by many—to the extent that people literally bet their life on these protections—are promises that it cannot keep. The centralized trust model doesn't work [11].

**Myth 7: Java Authentication and Authorization Services (JAAS) framework handles all protocols and mechanisms in a secure manner.**
As aforementioned, the Internet resources and SOA web pages are web applications. As such, they make use of the JAAS framework, which is a user-centric authentication and authorization collection of Eclipse plug-ins to manage authentication and authorization within an application built on the Rich Client Platform framework. The plug-ins provide an implementation of the JAAS API and can be extended by developers to support their own security needs.

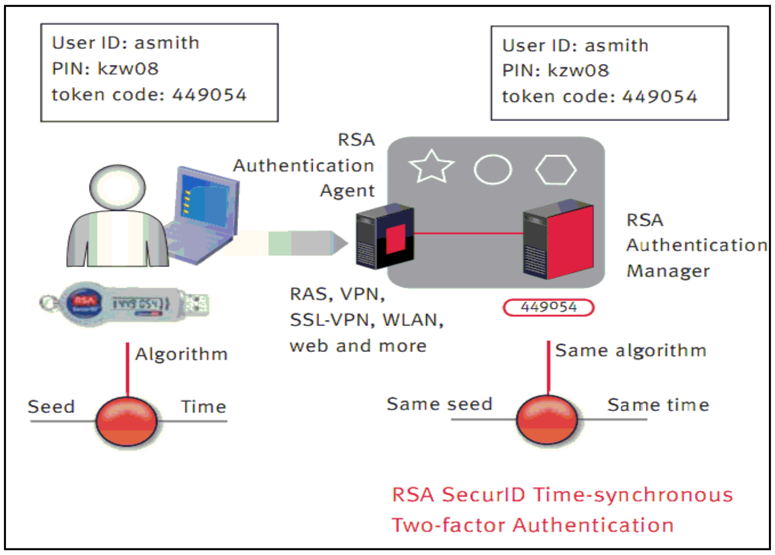
The code snippet below shows how easy it is to disable every authorization check in a system implementing JAAS.

```
public pointcut hackJAAS();
: call( * AccessController.checkPermission(..) );
void around() : hackJAAS()
{
//Do nothing. No proceed-call.
}
```

The reason this is such an easy task is that JAAS is a standardized framework. To perform an authorization check, a user must call AccessController.checkPermission. Everyone knows this—both lawful programmers and hackers. That means that if an application uses JAAS, a hacker automatically know which code they need to disable. The hacker doesn't need to see the source code, nor do they need to see any kind of documentation [10]. The Norwegian Information Security Laboratory does an excellent job of explaining the technical details of this vulnerability.

**Myth 8: RSA SecurID provides a secure connection.**
The RSA SecurID token authentication system is a two-factor authentication method, which is the most common secure access method in the U.S. with 40 million users. The RSA SecurID token authentication method uses the RSA ACE Server, which is a clock synchronization key scheme. It works on a timing frequency that changes the token keys so that they never seem to be the same. The frequency and the seed key were both found on the RSA ACE Server, which was hacked by perpetrators on March 18, 2011 [3].

Here is the way one inventor describes the scheme in his patent granted in 2008: "The pseudo-random token codes are only valid during a short time that they are displayed (e.g. 30 seconds). A hash function that generates the pseudo-random token code takes a current time and a secret key as inputs. The secret key is provided to the token by the manufacturer and then provided to the authentication server.



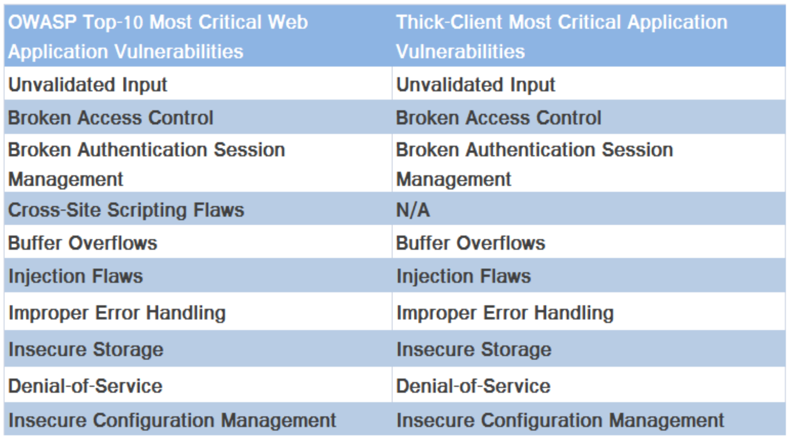
**Source: EMC Corporation**

This scheme makes the authentication system very time sensitive. If an authentication server and token have clocks that diverge, the system quickly breaks. Also, the security of the leading hash function has been called into question." The inventor is referring to a detailed cryptanalysis study by

Springer-Verlag, 2003 [4]. These researchers found that the block cipher at the heart of the RSA SecurID hash function can be broken in a few milliseconds using a 2003-vintage PC.

**Myth 9: Thick-client SSL VPNs are more secure than thin-client SSL VPNs.**
Thick client is defined as an application client that processes data in addition to rendering. An example of a thick client application can be a Visual Basic, JAVA or VB.NET application that communicates with a database.

The risks observed in thick-client applications generally include information disclosures, unauthorized access, authentication bypass, application crashes, unauthorized, high privilege transactions or privilege escalations. With the single exception of cross-site scripting, the vulnerabilities of thick clients are the same as the top 10 OWASP vulnerabilities of web applications [20].



| OWASP Top-10 Most Critical Web Application Vulnerabilities | Thick-Client Most Critical Application Vulnerabilities |
|---|---|
| Unvalidated Input | Unvalidated Input |
| Broken Access Control | Broken Access Control |
| Broken Authentication Session Management | Broken Authentication Session Management |
| Cross-Site Scripting Flaws | N/A |
| Buffer Overflows | Buffer Overflows |
| Injection Flaws | Injection Flaws |
| Improper Error Handling | Improper Error Handling |
| Insecure Storage | Insecure Storage |
| Denial-of-Service | Denial-of-Service |
| Insecure Configuration Management | Insecure Configuration Management |

**Source: Paladion Networks – 2005**

**Myth 10: Security is the responsibility of a specialist department.**
The prevailing attitude is that the responsibility for security falls in the court of someone with a job title related to security, such as application security specialist, cyber security guru or chief security officer, etc. As a result, the well-known SSL vulnerability announcements are overlooked, ignored and never mentioned by the development staff. They use SSL technology as provided by the application server vendor and the SSL technology as provided by their company's VPN client vendor to remotely log in to use sensitive company resources. Consequently, few of these users ever realize that effective security should be everyone's concern.

Of course, this mentality is not entirely the fault of employees. The companies themselves and their executive leadership are ultimately responsible for ensuring all personnel have adequate security training. Legal statutes and regulatory regimes in every industry require companies to create a culture of awareness and security knowledge through effective training programs.

When organizations lack definitive security policies, business processes driven by effective security, system development and engineering requirements that embody security precautions, as well as industry best practices showing tangible evidence of security precautions, employees are left with the broad assumption and personal disposition that the responsibility for security is someone else's job, not theirs.

## Summary

Against a backdrop of over-hyped SSL protocol use, vulnerabilities are endemic to the point where banks have their customer data stolen at an alarming rate, web application developers create a false sense of security by trusting the confidence and credibility of a protocol that is likely to fail them before they can get through a single development cycle.
And, of course, there are the private consumers (750 million unsuspecting Facebook and / or Skype users), ever trusting in their product vendors and their ISPs to steer them in the right direction for staying secure while browsing the Internet, socializing with family and friends online, etc. Real security protection won't be found in SSL VPN whether using a thick client, clientless application or SOA web services; anyone using OpenSSL for VPN and / or web services access to be the only VPN protocol relied on.
As seen by the many studies presented in this paper, many of the attack vectors used against SSL are from other technologies using SSL or that SSL makes use of—whether it is block ciphers, stream ciphers, document format, and authentication and authorization scheme, among others. Conventional wisdom would suggest that if we're serious about using a security paradigm, we get serious about testing the software before it is released to the public—having them test it for their most secure enterprise production requirements and most sensitive personal conversations or online business transactions is counter-productive.

**About the Authors**
**Rainer Enders, CTO, Americas, NCP engineering**
Mr. Enders is CTO, Americas for secure remote access and VPN solution provider NCP engineering. He has 20 years of experience in the networking and security industry. His other areas of expertise are test automation in quality assurance and the testing and verification of complex network and system architectures. Prior to joining NCP in January 2010, Rainer headed his own strategic consulting firm that focused on computer and network security and storage networking. Before that, he held a variety of technical roles at Identity Engines, Neoscale Systems, Yipes Enterprise Services and Ericsson.

**Clint Stewart, Cyber Security Architect**
Mr. Stewart is an independent consultant, and expert in security engineering holding NSA certifications in ComSec and InfoSec, CompuSec, TranSec, SigInt, NetSec, and cyrptographics systems. He is formerly a member of the DHS InfraGard contributing as the telephony security expert for number portability. He is also a member of the NSA's Information Assurance Technical Frameworks Forum with similar contributions to describe, definitize, and standardized the accreditation process for number portability systems and other areas of information assurance including the Global Information Grid (GIG). Additionally, he is a member of the DoD's Information Assurance Technical Advisory Center where he advises on a wide range of information assurance challenges requiring operational solutions for software and systems security. He is an active member of the NERC and NIST standards development teams working on SCADA/IED security and smart grid cyber security.

## Research References

1 - OpenSSL.org official security vulnerabilities listing from 2001 to 2011.
http://www.openssl.org/news/vulnerabilities.html
2 - Kelly Jackson Higgins, 2008. United Business Media LLC. InformationWeek - Treasury Takes Security Up A Notch. http://download.entrust.com/resources/download.cfm/23445/
3 - Robert McMillan, 2011. CIO Magazine - RSA Warns SecurID Customers After Company is Hacked.
http://www.cio.com/article/677467/RSA_Warns_SecurID_Customers_After_Company_is_Hacked
4 - Springer-Verlag, 2003. Katholieke Universiteit Leuven, Belgium - Department of Electrical Engineering. Cryptanalysis of the Alleged SecurID Hash Function.
Cryptanalysis of the Alleged SecurID Hash Function (extended version)*
5 - Terry Hayes , 2010. Mozilla.org - A Vulnerability in SSL/TLS implementations of ciphersuites that use block ciphers. http://www.mozilla.org/projects/security/pki/nss/news/vaudenay-cbc.html
6 – Arindam Mandal, 2005. Security Docs - Thick Client Application Secuirty. Taken here...
http://www.securitydocs.com/library/2885/
7 - Remote System Explorer User’s Guide. IBM Corporation 2000, 2011.
http://help.eclipse.org/helios/index.jsp?topic=/org.eclipse.rse.doc.user/tasks/tworkssl.html
8 – Cory Doctorow, 2011. BoingBoig Blog - Understanding the SSL security breach, preparing for the next one. http://www.boingboing.net/2011/03/24/understanding-the-ss.html
9 - Peter Eckersley, 2011. Electronic Frontier Foundation Blog. Iranian hackers obtain fraudulent HTTPS certificates: How close to a Web security meltdown did we get?
https://www.eff.org/deeplinks/2011/03/iranian-hackers-obtain-fraudulent-https
10 - Teemu Koponen, Juha Mynttinen, 2001. University of Houston, School of Science and Computer Engineering. Secure Sockets Layer (SSL) Man-in-the-middle attack.
http://sce.uhcl.edu/yang/teaching/csci5931webSecuritySpr04/secure%20Sockets%20Layer%20%28SSL%29%20Man-in-the-middle%20Attack.htm
11 - By Peter Bright, 2011. Ars Technica - How the Comodo certificate fraud calls CA trust into question. http://arstechnica.com/security/news/2011/03/how-the-comodo-certificate-fraud-calls-ca-trust-into-question.ars
12 - John Marchesini, S.W. Smith, Meiyuan Zhao, Department of Computer Science - Dartmouth College, 2003.Keyjacking: Risks of the Current Client-side Infrastructure.
http://middleware.internet2.edu/pki03/presentations/11.pdf
13 - Department of Homeland Security, US-CERT Division, 2001. Vulnerability Note VU#565052 - Passwords sent via SSH encrypted with RC4 can be easily cracked.
https://www.kb.cert.org/vuls/id/565052
14 - Hongjun Wu, 1998. The Misuse of RC4 in Microsoft Word and Excel published by Institute for Infocomm Research, Singapore. http://eprint.iacr.org/2005/007.pdf
15 - Bruce Schneier, 2005. Schneier on Security Blog.
http://www.schneier.com/blog/archives/2005/01/microsoft_rc4_f.html
16 – Remote System Explorer User’s Guide. IBM Corporation 2000, 2011.
http://help.eclipse.org/helios/index.jsp?topic=/org.eclipse.rse.doc.user/tasks/tworkssl.html
17 - Hågen Hasle, 2002 – Norwegian Information Security Laboratory. ASPECT-ORIENTED PROGRAMMING AND SECURITY: A comparison between implementing JAAS with AOP and OOP, 2002.

18 –M. Zalewski, 2006. SSL VPNs and Security, June 8, 2006.
http://seclists.org/fulldisclosure/2006/Jun/238
19 – R. Naraine, 2009 - Clientless SSL VPNs expose corporate users to attacks, December 1, 2009.
http://www.zdnet.com/blog/security/clientless-ssl-vpns-expose-corporate-users-to-attacks/5000
20 – A. Mandal, 2005 - Thick Client Application Security.
http://www.infosecwriters.com/text_resources/pdf/Thick_Client_Application_Security.pdf
21 - Anton D. Nagy, 2011. Corey Online Media, LLC - Microsoft Working On WP7 Patch To Fix SSL Vulnerability? http://pocketnow.com/windows-phone/microsoft-working-on-wp7-patch-to-fix-ssl-vulnerability
22 - Juniper Networks, 2011. AT RISK: GLOBAL MOBILE THREAT STUDY FINDS SECURITY VULNERABILITIES AT ALL TIME HIGHS FOR MOBILE DEVICES.
http://www.juniper.net/us/en/company/press-center/press-releases/2011/pr_2011_05_10-09_00.html
23 - Christophe Vandeplas, 2008. Watchguard Fireware SSL-VPN Vulnerability.
http://christophe.vandeplas.com/2009/08/watchguard-fireware-ssl-vpn_02.html
24 - Wolfgang Kandek, 2011. Qualys, Inc. - The Inconvenient Truth About the State of Browser Security.
http://laws.qualys.com/SPO1-204_Kandek.pdf
25 - Mark Wodrich, 2011. Microsoft Corporation - Notes on exploitability of the recent Windows BROWSER protocol issue. http://blogs.technet.com/b/srd/archive/2011/02/16/notes-on-exploitability-of-the-recent-windows-browser-protocol-issue.aspx
26 - Skype Limited, 2011.
http://about.skype.com/
27 - Salman A. Baset and Henning Schulzrinne, 2004. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. http://arxiv.org/ftp/cs/papers/0412/0412017.pdf
28 – Courtney Boyd Myers, 2011. The Next Web. Skype signs on over 27 million users in one day.
http://thenextweb.com/apps/2011/01/10/skype-signs-up-over-1-million-users-in-one-day/
29 – Internet Society, 2011. Frequently Asked Questions - Is IPv6 likely to introduce new security vulnerabilities? http://wiki.chapters.isoc.org/tiki-index.php?page=IPv6+FAQ#C1_Is_IPv6_likely_to_introduce_new_security_vulnerabilities_
30 – Samuel Sotillo, 2006. IPv6 Security Issues.
http://www.infosecwriters.com/text_resources/pdf/IPv6_SSotillo.pdf
31 – Jennifer LeClaire, 2011. Yahoo News - Facebook Rolling Out Skype Multi-User Video Chat.
http://news.yahoo.com/facebook-rolling-skype-multi-user-video-chat-232221556.html
32 – Peter Bright, 2011. Ars Technica - Citigroup latest bank to disclose hack: 200k accounts compromised.
http://arstechnica.com/security/news/2011/06/citigroup-latest-bank-to-disclose-hack-200k-accounts-compromised.ars

## Appendix – A: Notes on exploitability of the recent Microsoft Windows BROWSER protocol issue

16 Feb 2011 3:47 PM

Earlier this week a PoC exploit for a vulnerability in the BROWSER protocol was released on Full Disclosure. There has been some discussion regarding whether this issue can result in Remote Code Execution (RCE) or is only a Denial of Service (DoS). This blog post provides details on the exploitability based on Microsoft's internal analysis.

Which systems are vulnerable?
All versions of Windows are vulnerable, although the issue is more likely to affect server systems running as the Primary Domain Controller (PDC). In environments following best practices, the BROWSER protocol should be blocked at the edge firewalls thus limiting attacks to the local network.

The BROWSER protocol operates on top of SMB and is used to discover machines and resources on the network. It is implemented as a kernel driver (mrxsmb.sys or bowser.sys, depending on the version of Windows). This vulnerability affects Windows machines that have been configured to (A) use the BROWSER network protocol and (B) that then become Master Browser on the local network. The BROWSER protocol uses an election process to determine which system will act as the "master" in terms of data collection and response handling.

In normal enterprise networks the Primary Domain Controller (PDC) will become Master Browser, but depending on the network configuration, other computers on the network can become Master Browser, and therefore be vulnerable. A single system will be Master Browser at any point in time. (See the [MS-BRWS] protocol specification for more details).

Root cause of the vulnerability
A malformed BROWSER message would cause the Master Browser to hit the vulnerable code below and trigger the vulnerability.

The vulnerability is due to an integer underflow where a 32-bit length value becomes -1. This is then used in a memcpy call as follows:

```
if ( Length > 0 ) {
   RtlCopyMemory(StringOffset, InsertionString, Length*sizeof(WCHAR));
```

The copy length will therefore be -2 (0xFFFFFFFE) on 32-bit systems, and 0x1FFFFFFFE on 64-bit systems. This leads to a kernel pool buffer overrun in the context of a thread running at the PASSIVE IRQ level on Windows XP and Windows Server 2003. On Vista and later platforms, the thread will be running at the DISPATCH IRQ level. Threads running at PASSIVE IRQ level can be pre-empted by the operating system, making exploitation slightly more likely.

Effects of the buffer overrun

The copy operation will cause a bugcheck when invalid memory is referenced either from the source or destination address. This bugcheck is almost guaranteed to happen, since roughly 4GB of contiguous memory will be referenced during the copy on 32-bit systems. (On 64-bit systems, roughly 8GB of contiguous memory will be referenced)

It is impossible to have 4GB of contiguous virtual address space mapped at both the source and destination on 32-bit systems. Having 8GB of contiguous virtual address space mapped may be possible on 64-bit systems with sufficient physical memory.

RCE may also be possible if the corrupted memory is used before the RtlCopyMemory triggers a bugcheck, and in a way that can be used to change code execution. For instance, a thread running in another processor may reference the corrupted memory in parallel (while the RtlCopyMemory operation is in progress). On platforms where the copy operation is done at the PASSIVE IRQ level, the thread could be pre-empted by a higher-priority thread or hardware interrupt. Should this happen, the higer-priority thread may reference corrupted memory in a way that can lead to RCE. Microsoft feels that triggering these timing conditions reliably will be very difficult.

Exploitability for RCE
Based on the conditions outlined above, while RCE is theoretically possible, Microsoft feels it is not likely in practice. DoS is much more likely.

Looking at the Exploitability Index (XI) rating scale, we would assign this an XI rating of ‘3’ – Functioning exploit code unlikely on Windows Vista and higher. On Windows XP and Windows Server 2003, the XI rating would be ‘2’ – Inconsistent exploit code likely.

Thanks to Brian Cavenah and Matt Miller for their input.  The Microsoft Malware Protection Center (MMPC) blog also has some notes on this issue that you can find here:
http://blogs.technet.com/b/mmpc/archive/2011/02/16/my-sweet-valentine-the-cifs-browser-protocol-heap-corruption-vulnerability.aspx

- Mark Wodrich, MSRC Engineering [25]

## Appendix – B: Critical Security Advisory for WatchGuard Mobile VPN

```
- - - - - - - - - - - - - - - - - - - - - - - - - -
Security Advisory
- - - - - - - - - - - - - - - - - - - - - - - - - -
  Severity: High
  Title: Watchguard Fireware SSL-VPN MiTM Multiple Vulnerabilities
     Date: November 29, 2008
- - - - - - - - - - - - - - - - - - - - - - - - - -


 * Project: WatchGuard Firewall SSL-VPN
 * Version affected:
 WatchGuard Fireware 10.0 up to 10.2.2
 WatchGuard Mobile VPN with SSL 10.0 for Macintosh
 WatchGuard Mobile VPN with SSL 10.0.2 for Macintosh
 WatchGuard Mobile VPN with SSL 10.0 for Windows

 * Discovered:  April 06, 2008
 * Reported  :  April 08, 2008
 * Fixed     :  October 07, 2008 (only for Windows)
 * Advisory  :  November 29, 2008

 * Not Fixed :  Mobile VPN with SSL for Macintosh

 * Security risk: High Severity
 * Vulnerability: MiTM Multiple Vulnerabilities with Abritrary Code Execution

 * Discovered by: Christophe Vandeplas <christophe@vandeplas.com>

-------- SHORT DESCRIPTION --------
Due to bad design of the 'WatchGuard Mobile VPN with SSL Client'
it is vulnerable to a MiTM attack resulting in multiple consequences:
- Username and password gathering
- OpenVPN configuration poisoning
- Upload of malware on the victims machine
- Redirection to another VPN server
- Full transparent MiTM for the complete VPN tunnel
- Arbitrary code execution on the victims machine

This vulnerability should be classed as high severity.

-------- REMEDIATION --------

For Windows computers update ASAP your software.
```

Stop using the software on Macintosh computers until Watchguard releases a fixed Macintosh version of the client.

-------- FULL DESCRIPTION --------

-- [ How the Watchguard Fireware SSL-VPN Works

The Watchguard SSL-VPN system consists of four parts:
- OpenVPN Client
- WatchGuard Mobile VPN with SSL GUI Client (or Watchguard GUI Client)
- OpenVPN Server
- Webserver where the client-configuration resides (port 4100)

The OpenVPN Client and Server take care of the full VPN tunnel. As authentication the 'auth-user-pass', 'server-certificate' and 'client-cert' mechanisms are used. These are good security practices.
The Watchguard GUI takes care of two different tasks. It takes care as management interface for entering the OpenVPN credentials and it downloads the client configuration from the webserver on port 4100.
The webserver for the client-configuration runs on port 4100 and uses SSL for encryption. The certificate is self signed.

-- [ The Flow:

The Watchguard GUI Client needs the Firebox IP, username and password. When the user clicks on 'connect' the GUI Client connects on the webserver on ip:4100 using SSL encryption where it downloads the 'client.wgssl' file with the following HTTP GET:

```
GET /?action=sslvpn_download&username=testuser&password=testpass&filename=client.wgssl
```

A file called 'client.wgssl' is then downloaded on the machine. This file is a TGZ and contains the following files:

```
MD5SUM      - Checksums of the different files from the wgssl package
VERSION     - File with version info from Watchguard
ca.crt      - Certificate authority public key
client.crt  - Client public key
client.ovpn - OpenVPN configuration file
client.pem  - Private key for the client
```

The Watchguard GUI Client extracts the files and then starts up OpenVPN with this configuration.
OpenVPN then takes care of setting up the VPN tunnel. When the username and password are required it communicates with the Watchguard GUI Client using the OpenVPN management interface. The VPN tunnel is started.

-- [The problem:

The problem resides in the way GUI Client downloads the configuration. The webserver:4100 works with a self-signed certificate. The validity of this certificate is never checked correctly. The Client GUI does check the strings in the following fields:

```
Common Name (CN)        = Fireware Web Server
Organization (O)        = Watchguard
Organizational Unit (OU) = Fireware
```

These checks are insufficient. A full certificate check should be performed using either the manually imported certificate before connection or using an imported CA-cert.

This results in:
1) Anyone can generate a certificate with these values. An attacker could run a webserver and impersonate the original SSL-VPN server.
2) The password is stored in clear-text in the GET method.
3) The checks on the content of the client.wgssl are almost nihil. It is possible to completely replace the existing client configuration and to add extra files in the configuration directory of the client machine.
4) It is possible to redirect the client to another VPN server. As we have complete control the full VPN tunnel could be MiTM-ed without the user noticing this.
5) Combined with the 'up cmd' option of the OpenVPN configuration file arbitrary code could be executed on the victims machine.

-------- PROOF OF CONCEPT --------

--[ Main Setup

Take an original 'client.wgssl' file, rename it to 'client.wgssl.tgz' and extract it.
  Change the 'remote' value to another IP (1.1.1.1) in the 'client.ovpn' file and recalculate the checksum.
  Copy a 'malware.exe' in the same diractory. (this could be childporn or a virus)
  Compress all files again in a new .tgz package with the right filename.

An Apache webserver has been configured to run on port 4100 using the SSLEngine. The SSL certificate was generated with the following values:

```
  Common Name (CN)        = Fireware Web Server
  Organization (O)        = Watchguard
  Organizational Unit (OU) = Fireware
```

In the webroot of the webserver we uploaded a small script that saves the parametes into a database and returns a client.wgssl file as datastream.

-- [ Connect

Now open the Watchguard GUI Client and connect to the IP of the webserver. (192.168.1.110)
The Client downloads the new 'client.wgssl' file and extracts it.

-- [ Results:

1) The Watchguard GUI accepts the fake certificate
2) When checking the logfile we see a successful download of the configuration file:

```
  192.168.1.120 - - [08/Apr/2008:19:06:14 +0200] "GET
/?action=sslvpn_download&username=testuser&password=testpass&filename=client.wgssl HTTP/1.1" 200
12 "-" "-"
```

These usernames and passwords could be stored in a database using a php script. (see PHP Sample Script)

3a) The client configuration is completely rewritten with our own configuration file.
3b) Check the Watchguard GUI Client directory for the 'malware.exe' file.

4a) Check the logging, you will notice that the client doesn't connect to 192.168.1.110 but to 1.1.1.1
4b) Out of scope of this POC
5) Out of scope of this POC

-------- Technical SOLUTIONS for Watchguard --------

The WatchGuard Mobile VPN with SSL Client should correctly check the validity of the SSL Certificate using the well-defined standards when connecting to the configuration-website on port 4100.

The Watchguard firewall should enable the user to generate and upload certificates and to link these certificates to the webservers.
An even better solution would be to run both OpenVPN and the configuration-website on the same port and use the same certificates.
OpenVPN starting from version 2.1 supports a feature called port-sharing where OpenVPN can share a port with a webserver or other service.

-------- PHP Sample Script --------

```php
<?php
// Do whatever we want with the variables
$user=$_GET['username'];
$pass=$_GET['password'];

// Return the client.wgssl configuration file
header("Content-Type: application/octet-stream");
header("content-disposition: attachment; filename=\"client.wgssl\"");
```

```
$file=file_get_contents("client.wgssl");
echo $file;
?>
```