Microsoft, the NSA, and You

Here is the press release; for the full details, look **here**.

A sample program which replaces the NSA's key is **here**, at the bottom of the page.

NEW Have a question? A ***Frequently Asked Questions*** *(FAQ)* guide is now available! Please check the *FAQ* before emailing anyone. Thanks!

FOR IMMEDIATE RELEASE

Microsoft Installs US Spy Agency with Windows

Research Triangle Park, NC - 31 August 1999 - Between Hotmail hacks and browser bugs, Microsoft has a dismal track record in computer security. Most of us accept these minor security flaws and go on with life. But how is an IT manager to feel when they learn that in every copy of Windows sold, Microsoft may have installed a 'back door' for the National Security Agency (NSA - the USA's spy agency) making it orders of magnitude easier for the US government to access their computers?

While investigating the security subsystems of WindowsNT4, Cryptonym's Chief Scientist Andrew Fernandes discovered exactly that - a back door for the NSA in every copy of Win95/98/NT4 and Windows2000. Building on the work of Nicko van Someren (NCipher), and Adi Shamir (the 'S' in 'RSA'), Andrew was investigating Microsoft's "CryptoAPI" architecture for security flaws. Since the CryptoAPI is the fundamental building block of cryptographic security in Windows, any flaw in it would open Windows to electronic attack.

Normally, Windows components are stripped of identifying information. If the computer is calculating "number_of_hours = 24 * number_of_days", the only thing a human can understand is that the computer is multiplying "a = 24 * b". Without the symbols "number_of_hours" and "number_of_days", we may have no idea what 'a' and 'b' stand for, or even that they calculate units of time.

In the CryptoAPI system, it was well known that Windows used special numbers called "cryptographic public keys" to verify the integrity of a CryptoAPI component before using that component's services. In other words, programmers already knew that windows performed the calculation "component_validity = crypto_verify(23479237498234...,crypto_component)", but no-one knew exactly what the cryptographic key "23479237498234..." meant semantically.

Then came WindowsNT4's Service Pack 5. In this service release of software from Microsoft, the company crucially forgot to remove the symbolic information identifying the security components. It turns out that there are really two keys used by Windows; the first belongs to Microsoft, and it allows them to securely load CryptoAPI services; the second belongs to the NSA. That means that the NSA can also securely load CryptoAPI services... on your machine, and without your authorization.

The result is that it is tremendously easier for the NSA to load unauthorized security services on all copies of Microsoft Windows, and once these security services are loaded, they can effectively compromise your entire operating system. For non-American IT managers relying on WinNT to operate highly secure data centers, this find is worrying. The US government is currently making it as difficult as possible for "strong" crypto to be used outside of the US; that they have also installed a cryptographic back-door in the world's most abundant operating system should send a strong message to foreign IT managers.

There is good news among the bad, however. It turns out that there is a flaw in the way the "crypto_verify" function is implemented. Because of the way the crypto verification occurs, users can easily eliminate or replace the NSA key from the operating system without modifying any of Microsoft's original components. Since the NSA key is easily replaced, it means that non-US companies are free to install "strong" crypto services into Windows, without Microsoft's or the NSA's approval. Thus the NSA has effectively removed export

```
control of "strong" crypto from Windows. A demonstration program that replaces
the NSA key can be found on Cryptonym's website.

Cryptonym: Bringing you the Next Generation of Internet Security,
using cryptography, risk management, and public key infrastructure.

Interview Contact:
   Andrew Fernandes
   Telephone: +1 919 469 4714
   email: andrew@cryptonym.com
   Fax: +1 919 469 8708

Cryptonym Corporation
1695 Lincolnshire Boulevard
Mississauga, Ontario
Canada  L5E 2T2

http://www.cryptonym.com

# # #
```

The Full Details

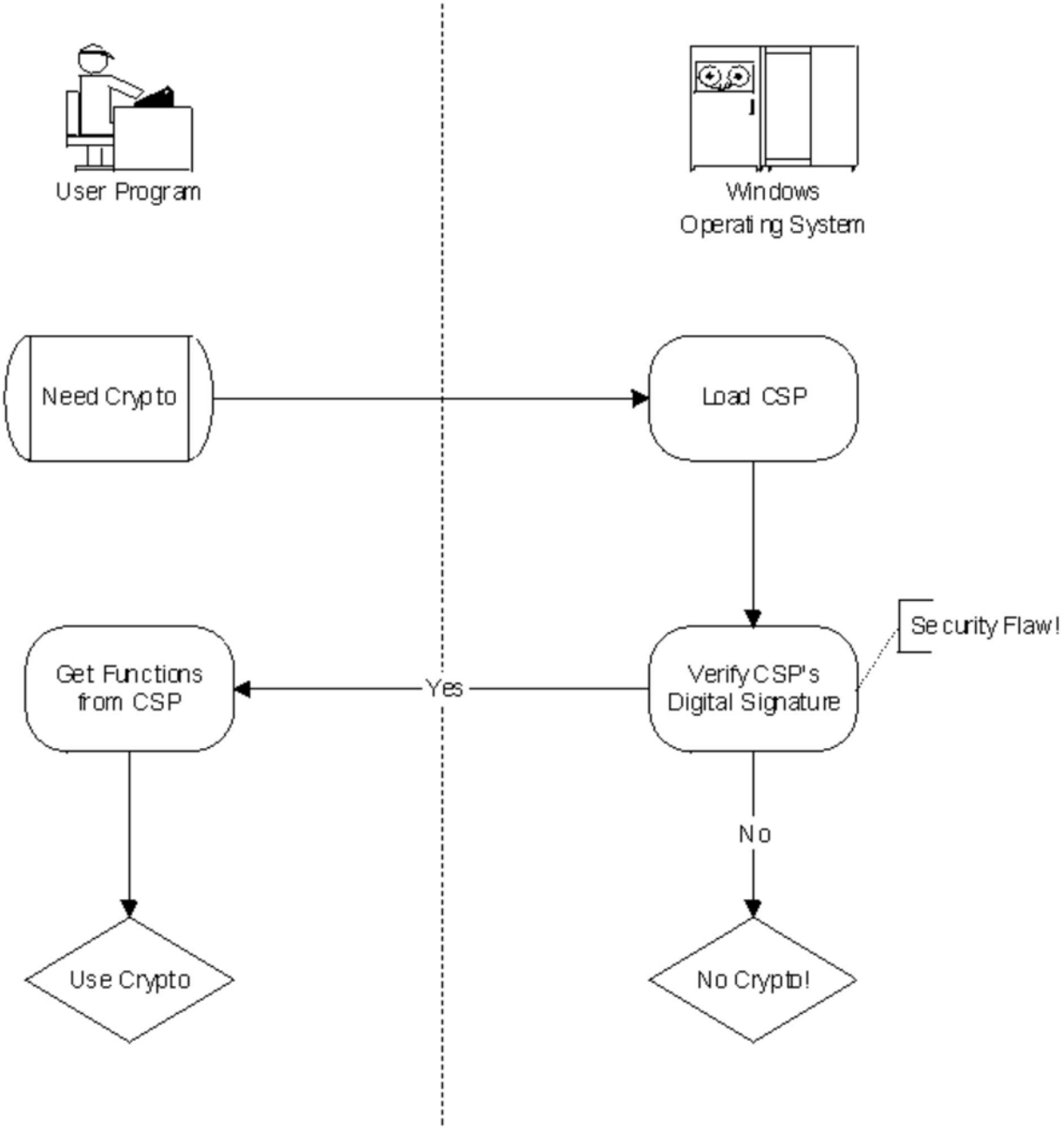
These details are essentially the contents of the "Rump Session" talk that Andrew Fernandes gave at the **Crypto'99** Conference, on 15 August 1999, in Santa Barbara, California.

Note 1: many people have written us and assumed that we "reverse engineered" Microsoft's code. This is not true; we did not reverse engineer Microsoft code at any time. In fact, the debugging symbols were found using standard Microsoft-purchased programmer's tools, completely by accident, when debugging one of our own programs.

Note 2: many reporters have stated that Andrew studied computer science at the **University of Waterloo** and was a classmate of Ian Goldberg of **Zero Knowlege Systems**. In fact, Andrew studied biochemistry and mathematics at Waterloo for his undergraduate, and mathematics at **McGill** for his graduate work. He and Ian graduated in the same year, but really did not know each other at the time.

An Overview of the Microsoft's CryptoAPI

Microsoft's CryptoAPI allows independent software vendors (ISVs) to dynamically load Cryptographic Serivce Providers (CSPs) as in the following diagram:



This arrangement of having Windows verify the CSP signature is what allows Microsoft to add cryptographic functionality to Windows. They will not digitally sign a CSP unless you first agree to abide by US export rules. Translation: Microsoft will not allow non-US companies to add strong crypto functions to Windows.

Fortunately, the verification of the CSP's digital signature opens up a security flaw in this picture.

Observations

Using NT4 Server, SP5 (domestic, 128-bit encryption version), and Visual C++ 6, SP3. These same results have been found in Win95osr2, Win98, Win98gold, WinNT4 (all versions), and Win2000 (up to and including build 2072, RC1).

Many people have emailed us to say that these debugging symbols are actually present in NT4-Workstation, and are in the original CD's debugging symbols! Thanks, people!

Before CSP loading		**in ADVAPI32.DLL**
Address 0x77DF5530	->	A9 F1 CB 3F DB 97 F5 ... ...
Address 0x77DF55D0	->	90 C6 5F 68 6B 9B D4 ... ...

After RC4 encryption using	**we see**

A2 17 9C 98 CA	=>	R S A 1 ... 00 01 00 01 ... (looks like an RSA public key)
A0 15 9E 9A C8	=>	R S A 1 ... 00 01 00 01 ... (looks like an RSA public key)

Looking at SP5 debugging symbols		**in "_CProvVerifyImage@8"**
Address 0x77DF5530	<-	has data tag "_KEY"
Address 0x77DF55D0	<-	has data tag "_NSAKEY"

Screenshots **One**, **Two**, **Three**, **Four**, and **Five** showing the actual debugging information.

The Flaw

An attack:

- Replace "_KEY" with your own key...
- ...but Windows will stop working since it cannot verify its own security subsystem!

An better attack:

- Replace "_NSAKEY" with your own key...
- ... Windows keeps working, since Microsoft's key is still there
- stops the NSA
- works because Windows tries to verify the CSP first using "_KEY", and then silently fails over to "_NSAKEY"

The Result:

- Windows CryptoAPI system still functional
- the NSA is kicked out
- the user can load an arbitrary CSP, not just one that Microsoft or the NSA signed!

Implications

1. What is the purpose of "_NSAKEY"? Espionage? Or do they simply not want to rely on Microsoft when installing their own CSPs?
2. Using RSA's Data Security's (now Security Dynamics) "BSafe" toolkit actually makes analysis of a program **easier**.
3. We do not need to modify the "advapi32.dll" file in order to remove the NSA key, nor do we need special privilleges on the machine.
    a. use self-modifying code
    b. needs undocumented vxd calls under Win95 and Win98
    c. needs special memory features under WinNT and Win2k
4. It is easy for any process to bypass **any** CSP and substitute its own.
5. Export control is effectively dead for Windows.
6. Note for Win2k - there appear to be **three** keys in Win2k; Microsoft's, the NSA's, and an unknown third party's. Thanks to Nicko van Someren for bringing this to our attention.

Removing the NSA

A sample program which temporarily replaces the NSA key with a test key, and leaves the rest of the CryptoAPI system intact, can be downloaded by clicking **this link** (currently only for WinNT and Win2k).

For legal reasons, source code will be provided for free, but only be available through a Nondisclosure Agreement with Cryptonym. You can download the NDA here in **Word/Wordperfect RTF** format, **PostScript PS** format, or **Acrobat PDF** format. *Make sure you initial all pages, and fax it to the phone number indicated.*

These files are provided for demonstration purposes only, and may not be redistributed or used for any purpose other than demonstration without the written authorization and license of Cryptonym Corporation. For more information, please contact:

Andrew Fernandes
email: **andrew@cryptonym.com**
Phone +1 919 469 4714
Fax +1 919 469 8708

Win95/98 Programmers: we could use help in porting the software to Win95/98. If you have a strong background in Win95/98 virtual memory management, virtual device writing, and Windows 'internals', and don't mind volunteering your time, please contact Andrew at the addresses above!

Microsoft & the "NSAKEY"
Frequently Asked Questions

This *FAQ* was last updated on 11 September 1999.
Make sure you've read the **original press release** before continuing!

If you have any questions or comments, please send them to **Andrew Fernandes**.

Answers

Again, if you have any questions or comments, please send them to **Andrew Fernandes**.

What is the danger?

Am I at risk with the "_NSAKEY"?

Almost certainly, the answer is no. The potential for a backdoor like this one should only worry organizations intending to rely on Windows for large scale, very security sensitive operations. Windows has enough known bugs that almost hacker can access and tamper with your system. For the average person, I would be more worried about *known* Windows bugs and viruses letting in hackers than I would be about the NSA spying on me.

What does the "_NSAKEY" affect? PGP? SSL?

The key affects *only* products that use Microsoft's CryptoAPI architecture. So far, that is a *very* small number of programs indeed. Most programs, such as Network Associate's **PGP** or Netscape's **SSL** do not (and probably never

will) use CryptoAPI. In fact, Microsoft themselves rarely use the CryptoAPI (or so we are told). You can look **here** for more info on Microsoft's CryptoAPI.

So who, if anyone, should be worried?

The only people who should possibly worry are those responsible for the operation of highly secure computers. Large organizations, companies, and banks may fit this description. Individuals almost certainly don't.

What is the real danger of the "_NSAKEY"?

The real danger represented by the "_NSAKEY" is that it shows how US companies must work with the NSA to gain export approval licenses. In the US, a company needs an export license from the US Federal Department of Commerce. Before granting this license, the Commerce Department requires the company pass technical review by the NSA. You can see Bruce Schneier's **Crypto-Gram** for how this can imply all sorts of nasty things. Also see [**Q12** - Why would it be an NSA key?].

What about the "removal program"?

Does the program remove the "_NSAKEY" permanently?

No. The program is a demonstration-only utility that loads a Microsoft-signed and a Cryptonym-signed Cryptographic Service Provider (CSP) into the CryptoAPI simultaneously. The CryptoAPI itself is not modified at all from its original form.

When will a Win95/98 version be available?

Whenever a programmer with lots of experience in kernel-mode virtual memory management under these operating systems donates his/her time to produce it. We've had a very few number of offers, but no serious discussions have occurred so far.

When and where can I buy the final program?

Never. First of all, by eliminating the second permanently, key we would be modifying the functionality of Microsoft provided code. This violates all sorts of copyright laws and is quite illegal. Second, we do not have (nor do we want) a valid RSA license needed to produce the digital signatures needed to run the demo. Therefore we only offer the program to be used in a research and development setting.

I get errors when running the program; why?

You need to be in very specific directories when running the various programs. Please read the "readme.txt" files in the distribution very carefully.

Why is the source available only under NDA?

We offer the source code (for free) under nondisclosure agreement to enforce our assertion that the program is to be used for research and development only.

Is Microsoft telling the truth?

What does Microsoft say?

The official Microsoft press release is **here**, as of 07 September 1999. Note that there are a few technical problems with what they have said. Specifically, Microsoft asserts:

- the "_NSAKEY" key cannot be used to start or stop security services on a computer.
  However, we have never discussed starting or stopping security services, merely loading CryptoAPI Cryptographic Service Providers that may be *already* used by *already* running security services.
- the "_NSAKEY" cannot be used by someone to weaken Windows' security.

However, we are not talking about a generic someone here; whoever owns or effectively controls the "_NSAKEY" can tamper with already running security services, and compromise Windows' security.

- the "_NSAKEY" is a merely a backup key for the first key (which is named "_KEY").
  However, that explanation doesn't make a lot of sense. For example:
  - Root keys are always (more accurately, should always) be symmetrically encrypted and cryptographically "split" just in case they are lost. Most tamper-resistant hardware works this way.
  - Having a second key operating in series with the first key allow the second key to be replaced (as per the demonstration program). In Microsoft's defense, they have written poor software suffering from the same problem like this before, such as the Authenticode framework.
  - Although conceivable, why are the keys not labeled "_KEY1" and "_KEY2", or "_KEY" and "_BACKUP_KEY"? That so-called unfortunate name "_NSAKEY" seems to be rather strong evidence that the NSA had input into the CryptoAPI system, at least on some level.

Furthermore, in the **Washington Post**, Microsoft indicated that the "_NSAKEY" was there "only a notation that the key conforms to technical standards set by the NSA". **However**, we need to point out that the NSA *has no technical standards* for publicly available cryptography. In the US, the National Institute of Standards and Technology (NIST) is responsible for cryptography standards. *If* it can be said that the NSA in some way does have crypto standards, then those are for *bad cryptography*. In other words, the NSA standards are for weak, escrowed, or back-door holes in cryptography.

Lastly, Microsoft asserts in their **press release** that losing the original "_KEY" would be devastating because the rely on it daily. However, by their own admission there are less than 50 Cryptographic Service Providers they have signed so far, in the past 2+ years that the CryptoAPI has been available. Software developers do not need to have their software signed by Microsoft until they are ready for a final release, thanks to Microsoft's own CryptoAPI Software Development Kit. Therefore it would seem that they need the key once every couple of weeks, on the average, at most.

What do other data security specialists say?

Ian Goldberg of **Zero Knowledge Systems**, agrees that the key can be used as some sort of back door, allowing "alternate" Cryptographic Service Providers to be loaded into Windows.

Bruce Schneier, of **Counterpane Systems** does not buy the arguument. He argues that there are already so many ways of compromising a Windows machine that installing yet-another-hole is just a silly idea, and that Microsoft would not be so stupid as to leave an incriminating debugging symbol around.

On the other hand, his company (along with **Peter Gutmann**, and others) has done quite a bit of work finding "stupid" cryptography and programming errors in Windows. Furthermore, his comments seem to have been made in the context of a personal computer user, where viruses and trojan-horses are a real concern. In a data center operation, the integrity of the computer components is what gives you security, not vulnerability to viruses; and the CryptoAPI is an integral component of Windows.

Lastly, Bruce concludes that either the key is just a backup key as Microsoft contends, or it was placed in to allow the NSA to produce Cryptographic Service Providers for their own internal use. We've already talked about the key being a backup key (see [**Q10** - What does Microsoft say?]). Regarding the possibility of NSA internal use, just remember that this key is in every copy of almost every operating system Microsoft sells. If it can be used for NSA internal use, why not external use?

Point in fact, Microsoft does not need the actual crypto DLL to sign it. Instead, all they need is a 16-byte "digest" of the program, called an "MD5 hash". Therefore, Microsoft could easily sign crypto modules without having access to the content of those modules themselves. So why have a separate key for the NSA?

Matt Blaze of AT&T Research said in the **New York Times** that the key is probably a backup key and nothing more. Again, see [**Q10** - What does Microsoft say?] for a response.

Why would it be an NSA key?

For an interesting read discussing An interesting link that discusses some of the previous adventures of the NSA can be found by **clicking here**. Other information on how the NSA works can be found by looking up "**Project Echelon**" as mentioned in the 14 August 1999 issue of **The Economist**.

Historically, Microsoft's behavior has been a little strange. Initially, developers were aware of one signature verification key embedded in the CryptoAPI. In August of 1998, Nicko van Someren of Ncipher (along with Adi

Shamir; the 'S' in RSA) found that there was a **second key** along with the first. Then, this August, we found the debugging symbols attached to the second key. With each development, Microsoft has said the same thing: don't worry, everything is fine, you have nothing to worry about. Yet every time, we find that they have not told us the whole story of how their security components really work, and what exactly were the terms on which they passed their NSA technical review.

Why would it not be an NSA key?

There are actually several good reasons why the second key may *not* be "owned" in some way by the NSA. We think it's inconceivable that the NSA did not have *some* sort of influence on the second key; but whether or not they "control" it is an open argument.

For instance, there are several good arguments that indicate the NSA would not need anything more than a one-time signature from Microsoft to be able to insert unauthorized Cryptographic Service Providers into a target system. Specifically, if the NSA wrote a "shim" module that inserted itself between Windows and the crypto module, and intercepted secure traffic. So, they may have no need to write multiple service providers. Then again, we don't know what they may have planned...

What should Microsoft do about this?

How can Microsoft acquit themselves?

Microsoft can come clean in this matter by telling its clients (and the world) the exact details of its deal with the NSA. If all they had to do was pass a technical review, let us see the requirements of that technical review. Let us talk to the developers and managers involved in working with the NSA. To get their export license, we already know that some sort of agreement with the NSA exists. We deserve to know the details.

Is any other operating system at risk?

Are you at risk if you use MacOS, Solaris, HP/UX, Tru64 Unix, AIX, VMS, OS/2, ... ?

Have you ever seen the source code to these operating systems? Who knows what's going on in them?

A lesson may come from a story partially retold **here**. IBM was shipping strong-crypto versions of Lotus Notes to Europe, but warned their customers that in order to get an export license, the US government had been given an escrow key to their "strong" crypto. Possibly because they didn't publicize the fact enough, the Swedish government decided to make heavy use of Notes' security... to their peril. So, in the end, IBM was really technically being honest, but possibly not quite honest enough.

Does Open-Source software, like Linux, help?

Yes and no. Having the source code to your operating system certainly helps reduce the risk of not knowing what's going on, but consider this:

- the latest Linux kernel alone is well over 1.5-million lines of source code,
- it would take a long time for some very experienced people to scan all of that code,
- you would have to rely on the judgment of all those programmers to spot and "irregularities", and
- nothing stops your hardware from being compromised if the software is secure.

These things being said, open source software definitely helps because it does give you more ways to manage the risk of relying on others. For instance, even though you would have to rely on the developers you had scanning the source code:

- lots of developers all over the world scan the code regularly, and there is a good chance an irregularity would be noticed, and
- at least you get to pick who you are going to rely on to form security judgments.

Remember, we cannot eliminate the risks that we run in life, but we can manage them.

Miscellaneous Questions...

What about that third key in Win2k?

Microsoft has stated that this is a testing key, used for internal development, and it should be removed in the final Win2k release.

This explanation is plausible and reasonable. However, since Win2k won't be shipping for at least six more months, by the time we find out, this whole issue may be dead. Furthermore, even if the key is still there, an excuse could be as simple as "oh, we forgot to remove it that time... you know, trying to make a late shipping deadline..."

Where can the debugging symbol be found?

Originally we reported that the symbol was only in NT4, sp5. However, many people have written in to report that the symbol can be found in every version of NT4 and every service pack. Unfortunately, we somehow missed them when we looked.

Wasn't this whole thing a little overly sensationalized?

Perhaps. There is a real danger in trying to explain a highly technical subject to the public that the ramifications will be either ignored or overly sensationalized.

For instance, due to the relatively soft stance taken by the team that broke the cellular telephone encryption schemes, most people today still believe that conversations on a digital cellular phone are secure from eavesdropping and cloning. In fact, the opposite is true; most modern digital cell phones are secure only against casual eavesdropping. Odds are that even this marginal security will disappear soon.

We decided to err on the side of getting the message out. Roughly, our message would be "don't trust a software or hardware vendor just because they have a good public relations department." A company that treats your security in a cavalier manner should reap what they have sown.

Do programmers use a silly names like "_NSAKEY" on purpose?

As a strictly anecdotal story, one reader wrote in to tell us that the X.509 Distinguished Name (DN) of the Lotus Notes' escrow key is "Big Brother". So yes, sometimes programmers do use silly names in their development. However silly the name "Big Brother" is, notice that it is still apt, and contains a kernel of truth about what Notes was doing.
