Microsoft's plans to stop people pirating the next version of Windows have suffered a setback.

A German computer magazine has found weaknesses in the piracy protection system built into Windows XP.

The weaknesses could mean that in up to 90% of cases users can circumvent the copy protection system.

But Microsoft said that the protection system would be much stronger and harder to defeat when the final version of XP is released later this year.

Component count

In a bid to combat piracy Microsoft is introducing a product activation system into the XP versions of its software. Activating a product involves contacting Microsoft for an identification number that is then combined with the serial numbers of the components inside your computer to create a unique identifier.

it would be possible to 'activate' nearly 90 percent of home-user machines without Microsoft knowing anything about it

Mike Hartmann, Tec Channel

Big changes to the hardware in a machine could mean that users have to contact Microsoft for a new identification number to re-activate their software.

By tying software to individual machines Microsoft hopes to stop its products being run on more machines than they are licensed for.

But now German computer magazine Tec Channel has analysed the product activation system that is being used in the test, or beta, versions of Windows XP and found that, in many cases, it can be compromised by making simple changes.

File fiddling

When Windows XP is first installed and activated it generates a file called wpa.dbl that stores information about the configuration of your machine.

Changes to any one of the ten components or serial numbers that this file watches are logged. When three changes have been made the wpa.dbl file is deleted forcing the user to contact Microsoft to reactivate the software.

But Mike Hartmann, a journalist at Tec Channel, has found that the ability of the wpa file to spot piracy can be easily compromised.

In tests Mr Hartmann installed and activated XP, then saved a version of the wpa file that was generated. He then changed components on the test machine so XP had to be re-activated. However, copying the old version of the wpa file back in the Windows system directory stopped requests for reactivation.

Piracy problems

The activation was also compromised when XP was fooled into thinking that a desktop PC was a laptop in a docking station, rather than a self-contained machine. In this configuration some components that wpa watches would be in the docking station rather than the portable computer. XP dutifully ignored any changes made to these components.

XP activation items
network card address
graphics card ID number
CPU serial number
SCSI host adapter number
IDE controller number
hard disk serial number
CPU type
Ram size
Volume ID
CD-Rom serial number

In total Mr Hartmann found a way to make the Windows XP activation technology ignore six of the ten components that it monitors. Mr Hartmann said another two can vary in only a small number of ways among all machines making it possible to create a "universal" wpa file that should activate XP on most PCs.

"With some smart tools that do automatic matching of hardware and activation-files it would be possible to 'activate' nearly 90 percent of home-user machines without Microsoft knowing anything about it," Mr Hartmann told BBC News Online.

Mr Hartmann expects to see activation file sites springing up on the web that offer wpa files tied to PCs with particular configurations thus ruining Microsoft's chances of cutting piracy.

"Should Microsoft stick with current version of wpa they will have wasted lots of money for call-center-employees, webservers and the technology itself," he said.

But a spokeswoman for Microsoft said that the version of the activation system that is in the pre-release versions of Windows XP is weaker than that which will ship with the finished version.

"The things that have been highlighted as a way of potentially bypassing activation will not be in the final code," said the spokeswoman. "The final code is going to be very different to what we have now."

"Product activation is not completely fixed in place at this time," she added.

Hacking WindowsXP Product Activation

Basic Issues

The file wpa.dbl in the directory system32 contains information on the system at the time of the Activation. If more than three hardware components are changed, Windows XP will notice it and delete wpa.dbl. With that the user shall be forced to activate XP anew. You do not get another 30 days of time, though, to activate again (in RC1 it is a fortnight). Instead XP takes the date of the installation as a basis. That means you have to activate immediately to run XP again, if the installation took place 30 days ago.

Volume serial number of the system volumes (displayed with dir-command)
MAC address of the network card (displayed with netstat -r -n)
Identification string of the CD ROM drive
Identification string of the displays
CPU serial number
Identification string of the system's hard disk
Identification string of the SCSI host adapter
Identification string of the IDE controller
String of the processor model

RAM size
1 = docking station, 0 = without docking station

First Tests

For a beginning we first of all saved the file wpa.dbl and then replaced the graphics card and the network card. As expected Windows XP was cooperative, so we could work without any disturbance. The first surprise showed up as we replaced the Celeron with a Pentium III: Suddenly Windows XP wanted to activate anew although we only changed three components.
The answer to the riddle is to be found in the serial number of the processor. Replacing the processor did not only change one but already changed two pieces of hardware information. For us that means to restart the computer and to switch off the serial number in the BIOS. Nonetheless XP insists on the Activation. A glance at wpa.dbl shows the reason why: Apparently XP put the file back in a non-activated condition. We again restart the computer, boot into DOS and copy the saved wpa.dbl back into the system directory of XP. With the next start of XP, the demand for Activation has disappeared. Evidently, wpa.dbl is the central authority to decide whether or not Activation already took place.

We re-install Windows XP on our computer from the ground up, using the very same product key. Nevertheless, the computer gets another product ID, as the last three digits are generated randomly. Although the product ID changed, Windows can be activated by copying the saved file wpa.dbl into the appropriate directory. Our next try brings an even bigger surprise: The Activation still works although we use a completely new product key for the installation.

Forged Hardware

These results kept in the back of our minds we try to activate Windows XP on another computer by copying the file wpa.dbl. First of all we adapt the volume ID of the new computer by means of freeware tools. The command line volumeid c: 3333-3333 changes the corresponding coefficient of the new system: The first component of Microsoft's protection is canceled.

With some network cards it is possible to adjust the MAC manually by means of the driver. The corresponding option in the register Advanced is called "Network Address" or "Locally administered Network Address".

So meanwhile we succeeded in switching off two components of the Activation by pretending another network address to the new system. The CPU serial number is switched off anyway, both computers do not have a SCSI host adapter and the memory is of the same size with both of them. With that altogether five sections of the hardware ID are identical.
Six actually, for both computers are not "to be docked". The latter gives us a bold plan...

Notebook of Eight Kilogrammes

What would happen if we tell the operating system that the computer is a notebook? This option can be toggled in the hardware profile of the device manager.

Can Microsoft be tricked that easily? Yes it can! After the next restart of the computer the analysis of the installation ID makes clear that suddenly the graphics card and the IDE/SCSI controller are no longer used to calculate the hardware ID.
So only three more differences in the configuration of the hardware remain:

Identification of the hard disk
Identification of the CPU
Identification of the CD-ROM drive

Because these three components are allowed to be different without XP insisting on a new Activation, this should be sufficient. So we copy the file wpa.dbl into the system32 directory of the second computer and start

Windows XP. In the start menu it still says "Activate Windows". But when you call it up, you get your just reward though:

Windows XP enlists ten hardware components to calculate the installation ID, but six of them can be canceled without any problems:

Component	To be canceled by
Volume ID	Adapted by means of tool
MAC address	Tuned by means of driver
Graphics card	Switch over to docking station
CPU serial number	Switch off in BIOS
SCSI host adapter	Switch over to docking station
IDE controller	Switch over to docking station

Important: A LAN does not tolerate two computers with the same MAC address.

Only four components are working almost effectively:

Component	Size of bit field
Hard disk	7
CPU type	3
CD ROM	7
RAM size	3

Two fields are coded with three bits and two with seven bits. Because in each field the coefficient 0 is impossible, 7*7*127*127=790321 possibilities remain for the file wpa.dbl. As only three components are allowed to change from the moment of Activation onwards, you can take the weakest fixed component for a "Universal Activation".

The CPU type or the RAM size present themselves here as the best solution. It is more than sufficient to only once activate a computer with 128 MBytes of RAM at Microsoft's. With its file wpa.dbl you can then "activate" all other computers of the same memory size.

Conclusion

With its technology of Activation Microsoft wants to thwart the user who occasionally copies software. Up to a certain degree this may still work. But by means of the above described steps nearly everybody can activate his own XP merely by getting a corresponding wpa.dbl file. There certainly will exist some web sites in the near future where the user can comfortably download "his"wpa.dbl.

Should the current procedure of Activation remain, then Microsoft will spend a lot of money like water for technology, web servers and call centers without any considerable success. It would be much more lucrative to drop the Activation and to lower the price for XP.

Microsoft did not comment on the weak points of the Activation until now. But probably their statement goes as follows: "In its final version WPA will look completely different. We did not implement these steps in the RC1 for only one reason, that is not to annoy the testers."

But it definitely is a fact that in-between the Release Candidates and the real Release normally only bugs are rectified. May sharp tongues call the WPA itself a bug, in our opinion it is nothing more but an example of bad programming.

Inside Windows Product Activation

A Fully Licensed Paper

July 2001

Fully Licensed GmbH, Rudower Chaussee 29, 12489 Berlin, Germany

ht*p://www.licenturion.com

>> INTRODUCTION

The current public discussion of Windows Product Activation (WPA) is characterized by uncertainty and speculation. In this paper we supply the technical details of WPA - as implemented in Windows XP - that Microsoft should have published long ago.

While we strongly believe that every software vendor has the right to enforce the licensing terms governing the use of a piece of licensed software by technical means, we also do believe that each individual has the right to detailed knowledge about the full implications of the employed means and possible limitations imposed by it on software usage.

In this paper we answer what we think are currently the two most important open questions related to Windows Product Activation.

* Exactly what information is transmitted during activation?

* How do hardware modifications affect an already activated installation of Windows XP?

Our answers to these questions are based on Windows XP Release Candidate 1 (build 2505). Later builds as well as the final version of Windows XP might differ from build 2505, e.g. in the employed cryptographic keys or the layout of some of the data structures.

However, beyond such minor modifications we expect Microsoft to cling to the general architecture of their activation mechanism. Thus, we are convinced that the answers provided by this paper will still be useful when the final version of Windows XP ships.

This paper supplies in-depth technical information about the inner workings of WPA. Still, the discussion is a little vague at some points in order not to facilitate the task of an attacker attempting to circumvent the license enforcement supplied by the activation mechanism.

XPDec, a command line utility suitable for verifying the presented information, can be obtained from http://www.licenturion.com/xp/. It implements the algorithms presented in this paper. Reading its source code, which is available from the same location, is highly recommended.

We have removed an important cryptographic key from the XPDec source
code. Recompiling the source code will thus fail to produce a working
executable. The XPDec executable on our website, however, contains
this key and is fully functional.

So, download the source code to learn about the inner workings of WPA,
but obtain the executable to experiment with your installation of
Windows XP.

We expect the reader to be familiar with the general procedure of
Windows Product Activation.

>> INSIDE THE INSTALLATION ID

We focused our research on product activation via telephone. We did
so, because we expected this variant of activation to be the most
straight-forward to analyze.

The first step in activating Windows XP via telephone is supplying the
call-center agent with the Installation ID displayed by msoobe.exe,
the application that guides a user through the activation process. The
Installation ID is a number consisting of 50 decimal digits that are
divided into groups of six digits each, as in

002666-077894-484890-114573-XXXXXX-XXXXXX-XXXXXX-XXXXXX-XX

In this authentic Installation ID we have substituted digits that we
prefer not to disclose by 'X' characters.

If msoobe.exe is invoked more than once, it provides a different
Installation ID each time.

In return, the call-center agent provides a Confirmation ID matching
the given Installation ID. Entering the Confirmation ID completes the
activation process.

Since the Installation ID is the only piece of information revealed
during activation, the above question concerning the information
transmitted during the activation process is equivalent to the
question

'How is the Installation ID generated?'

To find an answer to this question, we trace back each digit of the
Installation ID to its origins.

>>> Check digits

The rightmost digit in each of the groups is a check digit to guard
against simple errors such as the call center agent's mistyping of one
of the digits read to him or her. The value of the check digit is
calculated by adding the other five digits in the group, adding the

digits at even positions a second time, and dividing the sum by seven. The remainder of the division is the value of the check digit. In the above example the check digit for the first group (6) is calculated as follows.

```
1 | 2 | 3 | 4 | 5 <- position
---+---+---+---+---
0 | 0 | 2 | 6 | 6 <- digits

0 + 0 + 2 + 6 + 6 = 14 (step 1: add all digits)
0 + 6 + 14 = 20 (step 2: add even digits again)

step 3: division
20 / 7 = 2, remainder is 20 - (2 * 7) = 6

=> check digit is 6
```

Adding the even digits twice is probably intended to guard against the relatively frequent error of accidentally swapping two digits while typing, as in 00626 vs. 00266, which yield different check digits.

>>> Decoding

Removing the check digits results in a 41-digit decimal number. A decimal number of this length roughly corresponds to a 136-bit binary number. In fact, the 41-digit number is just the decimal encoding of such a 136-bit multi-precision integer, which is stored in little endian byte order as a byte array. Hence, the above Installation ID can also be represented as a sequence of 17 bytes as in

```
0xXX 0xXX 0xXX 0xXX 0xXX 0xXX 0xXX 0xXX
0x94 0xAA 0x46 0xD6 0x0F 0xBD 0x2C 0xC8
0x00
```

In this representation of the above Installation ID 'X' characters again substitute the digits that we prefer not to disclose. The '0x' prefix denotes hex notation throughout this paper.

>>> Decryption

When decoding arbitrary Installation IDs it can be noticed that the most significant byte always seems to be 0x00 or 0x01, whereas the other bytes look random. The reason for this is that the lower 16 bytes of the Installation ID are encrypted, whereas the most significant byte is kept in plaintext.

The cryptographic algorithm employed to encrypt the Installation ID is a proprietary four-round Feistel cipher. Since the block of input bytes passed to a Feistel cipher is divided into two blocks of equal size, this class of ciphers is typically applied to input blocks consisting of an even number of bytes - in this case the lower 16 of the 17 input bytes. The round function of the cipher is the SHA-1 message digest algorithm keyed with a four-byte sequence.

Let + denote the concatenation of two byte sequences, ^ the XOR operation, L and R the left and right eight-byte input half for one round, L' and R' the output halves of said round, and First-8() a function that returns the first eight bytes of an SHA-1 message digest. Then one round of decryption looks as follows.

```
L' = R ^ First-8(SHA-1(L + Key))
R' = L
```

The result of the decryption is 16 bytes of plaintext, which are - together with the 17th unencrypted byte - from now on interpreted as four double words in little endian byte order followed by a single byte as in

name	size	offset
H1	double word	0
H2	double word	4
P1	double word	8
P2	double word	12
P3	byte	16

H1 and H2 specify the hardware configuration that the Installation ID is linked to. P1 and P2 as well as the remaining byte P3 contain the Product ID associated with the Installation ID.

>>> Product ID

The Product ID consists of five groups of decimal digits, as in

AAAAA-BBB-CCCCCCC-DDEEE

If you search your registry for a value named 'ProductID', you will discover the ID that applies to your installation. The 'About' window of Internet Explorer should also yield your Product ID.

>>>> Decoding

The mapping between the Product ID in decimal representation and its binary encoding in the double words P1 and P2 and the byte P3 is summarized in the following table.

digits	length	encoding
AAAAA	17 bits	bit 0 to bit 16 of P1
BBB	10 bits	bit 17 to bit 26 of P1
CCCCCCC	28 bits	bit 27 to bit 31 of P1 (lower 5 bits)
		bit 0 to bit 22 of P2 (upper 23 bits)
DDEEE	17 bits	bit 23 to bit 31 of P2 (lower 9 bits)
		bit 0 to bit 7 of P3 (upper 8 bits)

The meaning of each of the five groups of digits is documented in the next table.

digits	meaning
AAAAA	apparently always 55034 (in Windows XP RC1)
BBB	most significant three digits of Raw Product Key
	(see below)
CCCCCCC	least significant six digits of Raw Product Key
	plus check digit (see below)
DD	index of the public key used to verify the
	Product Key (see below)
EEE	random value

As can be seen, the (Raw) Product Key plays an important role in generating the Product ID.

>>>> Product Key

The Raw Product Key is buried inside the Product Key that is printed on the sticker distributed with each Windows XP CD. It consists of five alphanumeric strings separated by '-' characters, where each string is composed of five characters, as in

FFFFF-GGGGG-HHHHH-JJJJJ-KKKKK

Each character is one of the following 24 letters and digits:

B C D F G H J K M P Q R T V W X Y 2 3 4 6 7 8 9

Very similar to the decimal encoding of the Installation ID the 25 characters of the Product Key form a base-24 encoding of the binary representation of the Product Key. Decoding the Product Key yields a multi-precision integer of roughly 115 bits, which is stored - again in little endian byte order - in an array of 15 bytes. Decoding the above Product Key results in the following byte sequence.

```
0x6F 0xFA 0x95 0x45 0xFC 0x75 0xB5 0x52
0xBB 0xEF 0xB1 0x17 0xDA 0xCD 0x00
```

Of these 15 bytes the least significant four bytes contain the Raw Product Key in little endian byte order. The least significant bit is removed by shifting this 32-bit value (0x4595FA6F - remember the little endian byte order) to the left by one bit position, resulting in a Raw Product Key of 0x22CAFD37, or

583728439

in decimal notation.

The eleven remaining bytes form a digital signature, allowing verification of the authenticity of the Product Key by means of a hard-coded public key.

>>>> Product Key -> Product ID

The three most significant digits, i.e. 583, of the Raw Product Key's

nine-digit decimal representation directly map to the BBB component of
the Product ID described above.

To obtain the CCCCCCC component, a check digit is appended to the
remaining six digits 728439. The check digit is chosen such that the
sum of all digits - including the check digit - is divisible by
seven. In the given case, the sum of the six digits is

7 + 2 + 8 + 4 + 3 + 9 = 33

which results in a check digit of 2, since

7 + 2 + 8 + 4 + 3 + 9 + 2 = 33 + 2 = 35

which is divisible by seven. The CCCCCCC component of the Product ID
is therefore 7284392.

For verifying a Product Key, more than one public key is available. If
verification with the first public key fails, the second is tried,
etc. The DD component of the Product ID specifies which of the public
keys in this sequence was successfully used to verify the Product Key.

This mechanism might be intended to support several different parties
generating valid Product Keys with different individual private keys.

However, the different private keys might also represent different
versions of a product. A Product Key for the 'professional' release
could then be signed with a different key than a Product Key for the
'server' release. The DD component would then represent the product
version.

Finally, a valid Product ID derived from our example Product Key might
be

55034-583-7284392-00123

which indicates that the first public key (DD = index = 0) matched and
123 was chosen as the random number EEE.

The randomly selected EEE component is the reason for msoobe.exe
presenting a different Installation ID at each invocation. Because of
the applied encryption this small change results in a completely
different Installation ID.

So, the Product ID transmitted during activation will most probably
differ in the last three digits from your Product ID as displayed by
Internet Explorer or as stored in the registry.

>>> Hardware Information

As discussed above, the hardware configuration linked to the
Installation ID is represented by the two double words H1 and H2.

>>>> Bit-fields

For this purpose, the double words are divided into twelve
bit-fields. The relationship between the computer hardware and the
bit-fields is given in the following table.

double word	offset	length	bit-field value based on
H1	0	10	volume serial number string
			of system volume
H1	10	10	network adapter MAC address
			string
H1	20	7	CD-ROM drive hardware
			identification string
H1	27	5	graphics adapter hardware
			identification string
H2	0	3	unused, set to 001
H2	3	6	CPU serial number string
H2	9	7	harddrive hardware
			identification string
H2	16	5	SCSI host adapter hardware
			identification string
H2	21	4	IDE controller hardware
			identification string
H2	25	3	processor model string
H2	28	3	RAM size
H2	31	1	1 = dockable
			0 = not dockable

Bit 31 of H2 specifies, whether the bit-fields represent a notebook
computer that supports a docking station. If docking is possible, the
activation mechanism will be more tolerant with respect to future
hardware modifications. Here, the idea is that plugging a notebook
into its docking station possibly results in changes to its hardware
configuration, e.g. a SCSI host adapter built into the docking station
may become available.

Bits 2 through 0 of H2 are unused and always set to 001.

If the hardware component corresponding to one of the remaining ten
bit-fields is present, the respective bit-field contains a non-zero
value describing the component. A value of zero marks the hardware
component as not present.

All hardware components are identified by a hardware identification
string obtained from the registry. Hashing this string provides the
value for the corresponding bit-field.

>>>> Hashing

The hash result is obtained by feeding the hardware identification
string into the MD5 message digest algorithm and picking the number of
bits required for a bit-field from predetermined locations in the
resulting message digest. Different predetermined locations are used
for different bit-fields. In addition, a hash result of zero is

avoided by calculating

Hash = (Hash % BitFieldMax) + 1

where BitFieldMax is the maximal value that may be stored in the bit-field in question, e.g. 1023 for a 10-bit bit-field, and 'x % y' denotes the remainder of the division of x by y. This results in values between 1 and BitFieldMax. The obtained value is then stored in the respective bit-field.

>>>> RAM bit-field

The bit-field related to the amount of RAM available to the operating system is calculated differently. The seven valid values specify the approximate amount of available RAM as documented in the following table.

value	amount of RAM available
0	(bit-field unused)
1	below 32 MB
2	between 32 MB and 63 MB
3	between 64 MB and 127 MB
4	between 128 MB and 255 MB
5	between 256 MB and 511 MB
6	between 512 MB and 1023 MB
7	above 1023 MB

It is important to note that the amount of RAM is retrieved by calling the GlobalMemoryStatus() function, which reports a few hundred kilobytes less than the amount of RAM physically installed. So, 128 MB of RAM would typically be classified as "between 64 MB and 127 MB".

>>>> Real-world example

Let us have a look at a real-world example. On one of our test systems the hardware information consists of the following eight bytes.

0xC5 0x95 0x12 0xAC 0x01 0x6E 0x2C 0x32

Converting the bytes into H1 and H2, we obtain

H1 = 0xAC1295C5 and H2 = 0x322C6E01

Splitting H1 and H2 yields the next table in which we give the value of each of the bit-fields and the information from which each value is derived.

dw &		
offset	value	derived from
H1 0	0x1C5	'1234-ABCD'
H1 10	0x0A5	'00C0DF089E44'
H1 20	0x37	'SCSI\CDROMPLEXTOR_CD-ROM_PX-32TS__1.01'

```
H1 27 | 0x15 | 'PCI\VEN_102B&DEV_0519&SUBSYS_00000000&REV_01'
H2 0 | 0x1 | (unused, always 0x1)
H2 3 | 0x00 | (CPU serial number not present)
H2 9 | 0x37 | 'SCSI\DISKIBM_____DCAS-34330______S65A'
H2 16 | 0x0C | 'PCI\VEN_9004&DEV_7178&SUBSYS_00000000&REV_03'
H2 21 | 0x1 | 'PCI\VEN_8086&DEV_7111&SUBSYS_00000000&REV_01'
H2 25 | 0x1 | 'GenuineIntel Family 6 Model 3'
H2 28 | 0x3 | (system has 128 MB of RAM)
H2 31 | 0x0 | (system is not dockable)
```

>>> Using XPDec

XPDec is a utility to be run from the command prompt. It may be
invoked with one of four command line options to carry out one of four
tasks.

>>>> XPDec -i

This option enables you to access the information hidden in an
Installation ID. It decodes the Installation ID, decrypts it, and
displays the values of the hardware bit-fields as well as the Product
ID of your product. Keep in mind that the last three digits of the
Product ID contained in the Installation ID are randomly selected and
differ from the Product ID displayed by Internet Explorer.

The only argument needed for the '-i' option is the Installation ID,
as in

```
XPDec -i 002666-077894-484890-114573-XXXXXX-XXXXXX-XXXXXX-XXXXXX-XX
```

>>>> XPDec -p

To help you trace the origin of your Product ID, this option decodes a
Product Key and displays the Raw Product Key as it would be used in a
Product ID.

The only argument needed for the '-p' option is the Product Key, as in

```
XPDec -p FFFFF-GGGGG-HHHHH-JJJJJ-KKKKK
```

Note that this option does not verify the digital signature of the
Product Key.

>>>> XPDec -v

This option calculates the hash of a given volume serial number. It
was implemented to illustrate our description of string hashing. First
use '-i' to display the hardware bit-fields. Then use this option to
verify our claims concerning the volume serial number hash.

The only argument needed for the '-v' option is the volume serial
number of your system volume, as in

```
XPDec -v 1234-ABCD
```

(The volume serial number is part of the 'dir' command's output.)

>>>> XPDec -m

This option calculates the network adapter bit-field value
corresponding to the given MAC address. Similar to '-v' this option
was implemented as a proof of concept.

The only argument needed for the '-m' option is the MAC address of
your network adapter, as in

XPDec -m 00-C0-DF-08-9E-44

(Use the 'route print' command to obtain the MAC address of your
network adapter.)

>> HARDWARE MODIFICATIONS

When looking at the effects of hardware modifications on an already
activated installation of Windows XP, the file 'wpa.dbl' in the
'system32' directory plays a central role. It is a simple
RC4-encrypted database that stores, among other things like expiration
information and the Confirmation ID of an activated installation,

a) the bit-field values representing the current hardware
configuration,

and

the bit-field values representing the hardware configuration
at the time of product activation.

While a) is automatically updated each time the hardware configuration
is modified in order to reflect the changes, remains fixed. Hence,
can be thought of as a snapshot of the hardware configuration at
the time of product activation.

This snapshot does not exist in the database before product activation
and if we compare the size of 'wpa.dbl' before and after activation,
we will notice an increased file size. This is because the snapshot is
added to the database.

When judging whether re-activation is necessary, the bit-field values
of a) are compared to the bit-field values of , i.e. the current
hardware configuration is compared to the hardware configuration at
the time of activation.

>>> Non-dockable computer

Typically all bit-fields with the exception of the unused field and
the 'dockable' field are compared. If more than three of these ten
bit-fields have changed in a) since product activation, re-activation
is required.

This means, for example, that in our above real-world example, we
could replace the harddrive and the CD-ROM drive and substantially
upgrade our RAM without having to re-activate our Windows XP
installation.

However, if we completely re-installed Windows XP, the information in
would be lost and we would have to re-activate our installation,
even if we had not changed our hardware.

>>> Dockable computer

If bit 31 of H2 indicates that our computer supports a docking
station, however, only seven of the ten bit-fields mentioned above are
compared. The bit-fields corresponding to the SCSI host adapter, the
IDE controller, and the graphics board are omitted. But again, of
these remaining seven bit-fields, only up to three may change without
requiring re-activation.

>> CONCLUSIONS

In this paper we have given a technical overview of Windows Product
Activation as implemented in Windows XP. We have shown what
information the data transmitted during product activation is derived
from and how hardware upgrades affect an already activated
installation.

Looking at the technical details of WPA, we do not think that it is as
problematic as many people have expected. We think so, because WPA is
tolerant with respect to hardware modifications. In addition, it is
likely that more than one hardware component map to a certain value
for a given bit-field. From the above real-world example we know that
the PX-32TS maps to the value 0x37 = 55. But there are probably many
other CD-ROM drives that map to the same value. Hence, it is
impossible to tell from the bit-field value whether it is a PX-32TS
that we are using or one of the other drives that map to the same
value.

In contrast to many critics of Windows Product Activation, we think
that WPA does not prevent typical hardware modifications and,
moreover, respects the user's right to privacy.

>> ABOUT THE AUTHORS

Fully Licensed GmbH is a start-up company focusing on novel approaches
to online software licensing and distribution. Have a look at their
website at

http://www.licenturion.com

for more information.

Their research branch every now and then analyzes licensing solutions
implemented by other companies.

>>
>> Frequently asked questions and their answers
>> concerning the Fully Licensed WPA paper
>>
>> Fully Licensed GmbH, July 10, 2001
>>

>> 1. Was Microsoft involved in the creation of the paper?

Microsoft was not involved in the creation of the paper in any
way. However, we made a draft version available to Microsoft to give
them a head-start. We consider it to be good etiquette to inform a
vendor of a pending publication related to one his or her products, so
that the vendor is able to prepare an official response.

>> 2. Why should we believe you?

We do not expect you to believe us. That's why we have provided our
complete knowledge about WPA and the XPDec utility. Combine both to
verify our claims.

>> 3. But Thomas Lopatic, one of your managing directors was born in
Unterschleissheim, Germany, which is the town near Munich in
which Microsoft's European headquarters are located.

This is a nice coincidence. It is in a way understandable - and at the
same time highly amusing to us :-) - that this has given rise to
rumors about the whole paper being a cleverly planned Microsoft
conspiracy.

Thomas was actually born in Karlsruhe, Germany. However, he was living
in Unterschleissheim from the 1970s - i.e. long before Microsoft moved
there - until recently, when he moved to Berlin. That's why some
records still list Unterschleissheim as the place where he
lives. Incorrectly interpreting these records led to the rumor that
Thomas was born in Unterschleissheim.

>> 4. Does Microsoft downplay the paper?

No, most definitely not. The paper really IS harmless. It does not
provide any information that would help a pirate circumvent WPA.

>> 5. Why did you release details on Windows Product Activation?

We felt that there is a need for facts in the debate about Windows Product Activation. Many people suspected that WPA could be abused to spy on end-users. Our paper, however, shows that insensitive information is transmitted during product activation. From this, it can be seen that the facts that we provide really are a necessary contribution to the ongoing discussion about WPA.

We think that license enforcement mechanisms will be an important part of the future of software distribution via the Internet. Thus, we do think that public discussion of technology of this kind must be free from bias and it must be based on facts and openness.

We hope that the information that we provide positively affects the current debate. The debate is necessary, but it should be based on facts and full disclosure of information relevant to the privacy question.

>> 6. Do you know how to circumvent Windows Product Activation?

No. We provide insight into which information is transmitted to Microsoft during activation. Our paper is important to help people understand the impact of WPA on their work and their privacy. We do not believe that our paper helps in any way to circumvent the license enforcement provided by WPA.

>> 7. Your paper says that Microsoft will err on the user's side.

What our paper shows is that a) no sensitive information is transferred to Microsoft and typical hardware upgrades do not negatively affect an already activated installation of Windows XP.

But, if you either completely re-install Windows XP or modify your hardware beyond what is tolerated by product activation, you have to re-activate Windows XP.

The important question now is: How often will Microsoft let you re-activate? Erring on the user's side would mean that they allow you to re-activate as often as you like, which seems to be what Microsoft says they will do.

It is, however, impossible to confirm this policy by means of a technical analysis.

>> 8. Why doesn't Microsoft know which hardware I use?

Let us consider the case of IDE controllers. In the installation ID transmitted to Microsoft they are represented by a 4-bit value. The 4 bits are obtained by applying the MD5 message digest algorithm to a string that uniquely identifies the vendor and model of the IDE controller, e.g.

```
'PCI\VEN_8086&DEV_7111&SUBSYS_00000000&REV_01'
```

and picking 4 bits from fixed locations in the resulting 128-bit message digest.

With 4 bits, we can represent 16 different values at maximum. However, there are far more than 16 different models of IDE controllers out there. So, since there are more models than 4-bit values, the above hashing procedure must yield the same 4 bits for more than one model. The more models there are, the more models will map to a given 4-bit value.

In contrast to what Microsoft says, the privacy that WPA provides is not based on the assumption that it is impossible to invert the employed message digest algorithm, i.e. MD5. If we used all 128 bits of the message digest derived from a hardware component's identification string, this 128-bit value would most probably uniquely identify the hardware component. If we used 128 bits, each hardware component on earth would probably map to a different value.

What an attacker would then do is build a list of all hardware components on this planet and calculate the corresponding 128-bit values, which are probably all different. Then finding the hardware component that corresponds to a certain 128-bit value is just a table lookup away.

Privacy is based on the fact that only a few bits of the resulting 128-bit message digest are considered. Obviously this leads to lots of collisions, i.e. lots of hardware components mapping to a given value. If there were 160 different models of IDE controllers, we could on average expect 160 / 16 = 10 models to map to the same 4-bit value.

Let us, as another example, consider the MAC address of an ethernet adapter. The discussion is technically not 100% accurate, but it illustrates the point. The MAC address is a 48-bit value, which means that it can theoretically be one of 281,474,976,710,656 different values. However, its 10-bit representation in the Installation ID is obtained by picking 10 bits from the MD5 hash over an ASCII string comprised of the 12 hex digits of the 48-bit value. Picking 10 bits leads to 1,024 different results at maximum.

So, on average, we expect

281,474,976,710,656 / 1,024 = 274,877,906,944

MAC addresses to map to the same 10-bit value. Because of this, nobody will be able to obtain the actual MAC address from the 10-bit value, since there are 274,877,906,944 candidate MAC addresses from which the 10-bit value could have been derived.

It is interesting to see that the bit-field that represents the MAC address is 10 bits in size, while the bit-field representing the IDE controller only consists of 4 bits. Microsoft probably have assigned a longer bit-field to a component if they expect more diversity in the

identification string of this component. The number of different IDE
controller models is smaller by orders of magnitude than the number of
different MAC addresses. So, to produce sufficient collisions, they
decided to use a relatively small bit-field for IDE controllers but
could still afford to chose a 10-bit bit-field in the case of MAC
addresses.

>> 9. What are the implications of re-activating after hardware
changes?

This is an interesting issue which is not covered in our paper. We
simply did not think of it. Our mistake. It was brought to our
attention by an article by Greg Falcon <veloso@verylowsodium.com> on
www.slashdot.org: If you have to re-activate your installation of
Windows XP because of hardware modifications, your new hardware
configuration is embedded in the Installation ID in the form discussed
above. While this does not enable anyone to find out which components
you have, it is trivial to find out which components you have
changed. Just examine which bit-fields have changed their value since
the original activation.

"Windows XP enlists ten hardware components to calculate the installation
ID, but six of them can be canceled without any problems:

Volume ID ---------- Adapted by means of tool
MAC address -------- Tuned by means of driver
Graphics card -------Switch over to docking station
CPU serial number - Switch off in BIOS
SCSI host adapter -- Switch over to docking station
IDE controller ------- Switch over to docking station

Important: A LAN does not tolerate two computers with the same MAC address.
"

(Switching to 'Docking Station' in Device Manager / Performance / File
System settings doesn't mean you actually have a docking station of any kind
so can be used for non-notebook computers that cannot even USE a docking
station!)

If you want WindowsXP on a network, you're stuck (until someone finds
another route around Activation - juding by the speed of this one, that
won't be long!) But then why would anyone want more than one copy of Windoze
XP on a LAN anyway!

"It is more than sufficient to only once activate a computer with 128 MBytes
of RAM. With its file wpa.dbl you can then "activate" all other computers of
the same memory size." [no matter what other hardware is installed as long
as it's set as a notebook and the volume tag etc is set to match].

"Can Microsoft be tricked that easily? Yes it can! After the next restart of
the computer [after changing to docking station] the analysis of the
installation ID makes clear that suddenly the graphics card and the IDE/SCSI

controller are no longer used to calculate the hardware ID. In computers
that can be docked, XP ignores the identification of the graphics card, the
SCSI host adapter and the IDE controller.

So only three more differences in the configuration of the hardware remain:

Identification of the hard disk
Identification of the CPU
Identification of the CD-ROM drive

Because these three components are allowed to be different without XP
insisting on a new Activation, this should be sufficient. So we copy the
file wpa.dbl into the system32 directory of the second computer and start
Windows XP. In the start menu it still says "Activate Windows". But when you
call it up, you get your just reward though:
"Windows Product Activation: Windows is already activated. Click OK to
exit."

"So first of all Tecchannel saved the file then started changing hardware.
Two items OK, but replacing a third - the CPU - triggered the deletion.
Although you'd think the CPU is only one component, it's actually tallied up
as two. Switching off the CPU serial number in the bios and therefore
knocking it down to one doesn't get the earlier wpa.dbl back - this has been
restored in a non-activated state.

Copy the saved version back? That surely shouldn't work - but it does. Next,
Tecchannel tried a completely new installation using the same product key.
This produces a new product ID, but nevertheless copying the wpa.dbl file
back again works.

They also use this file on another computer, altering the computer's volume
ID first, which is easily enough done. They can also use forged network
cards MAC addresses, so now they've taken two parts of the hardware ID out
of the picture. Next, use the hardware profile to tell the computer it's a
notebook with a docking station. This works, and tells WPA to stop counting
the IDE/SCSI controller and the graphics card.

That gets the differences counted down to three, hard disk, CPU and CDROM
ID, which is within the limit, so WPA is effectively toast.

What does this mean? Tecchannel's investigation shows that, at the very
least, you can use the same wpa.dbl file to activate as many computers as
you like, provided the RAM size is the same. A 'universal' file that didn't
even require the same RAM might be a possibility, but it's more likely that
people will simply swap files to get one appropriate for their hardware. "

Hope it enlights some of us to create a unique keygen just to get the confirmation ID generated after putting
the Installation ID on the same.

Enjoy!!