

## Known issue Internet Explorer en smartcard communicatie

**Versie** : 1.0  
**Status** : Definitief  
**Datum** : 22 oktober 2007

---

### Inleiding

In de WDH pilot in Enschede is een hardnekkig technisch probleem opgetreden dat veel tijd heeft gekost. Het is meestal omschreven als het 'wegvallen van de verbinding met de UZI-pas'. Dit werd veroorzaakt door het niet correct beëindigen van lopende communicatie met de UZI-pas bij het afsluiten van Microsoft Internet Explorer (IE) of het afsluiten van de actuele pagina. Dit memo beschrijft de volgende zaken:

- use case waarbij het optreedt en waarmee het reproduceerbaar is;
- symptomen;
- omgevingen waar het optreedt;
- analyse en standpunten leveranciers;
- oplossingsrichtingen;
- conclusies.

De doelgroep van dit memo zijn XIS applicatie ontwikkelaars. Sommige tekstfragmenten zijn in het Engels om te voorkomen dat vertaling misverstanden introduceert.

### Use case waarbij het optreedt en reproduceerbaar is

De volgende use case is gebruikt om het probleem te reproduceren:

- Voer de pas in de kaartlezer;
- Ga naar een website die met SSL beveiligd is (met client authenticatie);
- Selecteer het certificaat;
- Voer de PIN-code in;
- Sluit IE af zodra het LED<sup>1</sup>-je knippert.

Afhankelijk van kaartlezer, werkstation en smartcard zijn meerdere pogingen nodig omdat timing een rol speelt.

### Symptomen

Aan de volgende symptomen is herkenbaar dat het probleem is opgetreden:

- Het IE window verdwijnt maar in de Process Explorer is de IE instance nog steeds zichtbaar;
- Na opstarten van het Token Beheer Programma, staat de 'Token status' op 'aanwezig'. De status wijzigt niet in 'operationeel'. Na verwijderen en opnieuw invoeren van de smartcard wijzigt de 'Token status' in 'operationeel';
- Vanuit geen enkele applicatie is nog communicatie mogelijk met de pas. Zowel herinvoer van de pas als het 'killen' van de hangende IE instance geven de pas weer vrij voor communicatie.

---

<sup>1</sup> Dit is uiteraard alleen mogelijk als de kaartlezer via een LED signaleert dat er communicatie is. Dit is het geval bij de kaartlezer die standaard meegeleverd wordt: OmniKey Cardman 3121 USB.

### Omgevingen waar het optreedt

Het probleem doet zich voor bij Internet Explorer en is daadwerkelijk aangetoond met IE6 en IE7. Bij IE7 is het zowel reproduceerbaar op Windows XP (SP2) als Windows Vista.

De use case is daarnaast getest met:

- diverse middleware producten in combinatie met smartcards van verschillend fabrikaat. Deze testen zijn uitgevoerd met AET SafeSign Identity Client 2.3.2 en 2.0.14, GemPlus GemSafe 4.2.0 SP3 en Charismatics CSSI 4.0 Admin Edition. Met alle soorten middleware was de kern van het probleem te reproduceren, hoewel de symptomen enigszins verschilden. Van de drie geteste producten (AET SafeSign IC, GemPlus, Charismatics) heeft SafeSign IC nog het meest wenselijke c.q. begrijpelijke gedrag: het hangende IE proces wordt vrijgegeven na uitnemen en herinvoeren van de smartcard. Bij GemPlus en Charismatics is het opnieuw invoeren van de smartcard en herstarten van IE niet voldoende om weer een werkende configuratie te krijgen.
- AET SafeSign IC is ook getest met kaartlezers van Omnikey (3121 USB, 4000 PCMCIA) en Gemplus (GemPC Twin USB). Hier waren er alleen kleine verschillen in de kans van reproduceren.

Conclusie: in alle gevallen kan het issue optreden als Internet Explorer wordt afgesloten tijdens communicatie met de smartcard. Dezelfde use case leidt dus tot problemen bij gebruik van Internet Explorer.

Hoewel het issue niet specifiek is voor de UZI-pas, was het tot nu toe onbekend in de supportorganisatie van AET en Microsoft. Omdat het een timing aspect kent (afsluiten IE tijdens communicatie met smartcard) zijn er ook andere factoren die grote invloed hebben op de kans dat het issue optreedt:

- Applicatiegedrag en gebruikersgedrag. De omschreven Use Case is ongebruikelijk voor een gemiddelde gebruiker<sup>2</sup> die met een UZI-pas inlogt op een web-applicatie. Wanneer IE echter (embedded) opgestart en afgesloten wordt vanuit een andere applicatie kan de eindgebruiker zich niet bewust zijn van het feit dat er communicatie is met de smartcard of dat IE afgesloten wordt;
- Als er vanuit een applet communicatie is met de UZI-pas kan het laden van een andere web pagina eveneens de communicatie onderbreken;
- Communicatie frequentie. Als er vanuit een applet communicatie is met de UZI-pas kan er ook veel vaker gecommuniceerd worden (bijv. monitoring) dan strikt noodzakelijk is voor het opzetten van een geauthenticeerde SSL sessie.
- Snelheid van smartcard en kaartlezer. Als de pas niet snel communiceert, vergroot dit vanzelfsprekend de kans dat de communicatie onderbroken kan worden. Bij de UZI-passen die vanaf medio 2007 in productie uitgegeven worden, is de communicatie snelheid globaal een factor 2 hoger waardoor de kans kleiner is dat het issue optreedt.

Bovenstaande factoren verklaren waarom dit probleem niet eerder geconstateerd is en veel gebruikers probleemloos werken met IE, SafeSign IC en de UZI-pas.

In de WDH pilot was sprake van de volgende applicatie architectuur: IE werd (embedded) opgestart en afgesloten vanuit een call-management applicatie. Er was vanuit een applet actieve communicatie met de UZI-pas. Voor de gebruiker was niet duidelijk dat bepaalde gebruikersacties in de applicatie tot gevolg hadden dat IE werd afgesloten en de lopende communicatie met de UZI-pas werd onderbroken. Het leek alsof op willekeurige momenten

---

<sup>2</sup> Intuïtief zal een gebruiker een randapparaat niet verwijderen als er communicatie is. Een USB stick verwijderen bij het wegschrijven van bestanden wordt ook afgeraden.

geen communicatie meer mogelijk was met de UZI-pas vandaar de omschrijving 'wegvallen van de verbinding met de UZI-pas'.

### Analyse

Microsoft geeft de volgende analyse van wat er gebeurt:

*When we close IE in the middle of a transaction the following happens:*

- 1. We destroy the IE window*
- 2. The destruction of the window calls the onDestroy event, which some add-ons wait on to unload. As they unload they end their threads.*
- 3. Sets APCs (Asynchronous Procedure Calls) to end all the threads in the process – This is from the call to ExitProcess*
- 4. Calls LdrShutdownProcess that calls the entry points of the loaded dlls so that they can clean up – this is the second thing that ExitProcess initiates.*
- 5. As the entry point of AETPKSS1 is called in the main thread as a result of LdrShutdownProcess, it calls SCardDisconnect, which tries to acquire a lock on the smart card. However the smart card is still transmitting on another thread.*
- 6. AETPKSS1 kills three of its threads using terminatethread*
- 7. The thread transmitting runs the APC code and exits without calling SCardEndTransaction.*
- 8. The main thread continues to run waiting on the smart card.*

In een andere analyse geeft Microsoft het volgende aan:

*From the dump, we can see that IE is closing (kernel32!ExitProcess) and as it unloads aetpkss1.dll, this module then calls the winscard!SCardDisconnect with parameters hCard = 0xea010000, unsigned long dwDisposition = 0. dwDisposition = 0 is SCARD\_LEAVE\_CARD, meaning don't do anything special to the card (like reset it). This attempts to obtain a lock on the smartcard to disconnect (as we might do with a file - we need exclusive access). Since the smartcard is busy, we have to wait, so we call winscard!CSCardSubcontext::WaitForAvailable and the IE process remains in memory.*

AET heeft een vergelijkbare analyse over wat er gebeurt:

*When the problem happens the thread 0xC40 (containing the PKCS#11 functionality and communicating with the smartcard) is terminated in the middle of a PC/SC transaction. This causes the thread with threadid 0x884 (thread used by the application dll unloading mechanism) to become in a deadlock when disconnecting from the PC/SC subsystem.*

AET geeft daarbij nog de volgende interpretatie:

*This is expected functionality because the entire PC/SC system is designed in such a way that it is impossible to disconnect from a resource when a transaction is in progress. When this was not the case it would become possible to destroy a smart card by interrupting a PC/SC transaction. An application (in this case IE) should respect the PC/SC transactional system when shutting down. Not doing so may cause deadlocks. It must be stated that this can happen to all mechanisms using resource locking. In this situation it happens to the PC/SC subsystem but when using the design as it is chosen to implement IE it may also happen when using Databases, Files or other resources. A solution would be to use a shell around IE that controls its shutdown behaviour and only permits shutdown when no smart card transaction is in progress. The real problem is in the used architecture and not in the used technology, hence the solution has to come out of changing the used architecture.*

Microsoft volgt vanuit IE een andere redenering:

*IE on its own is not smartcard-aware and there is no smartcard code in the IE code. IE is, however, CryptoAPI aware and it is here that we interface the smartcard. Defining where the fix should occur is difficult especially in terms of what is meant by application. IE calls ExitProcess and leaves the shutdown to the system, which tells the threads to close then detaches the DLLs. That is perfectly ok – that is what the API is there for. However, here it causes the issue since the transmission is not in the main thread.*

*IE is made smartcard-aware by the loading of the AET modules. It would there seem reasonable that it should then be responsible for making sure the thread dealing with the transmission is kept alive until the transmission completes. This could be done by hooking exitprocess, hooking the wm\_close windows message or using a plugin to delay calling exitprocess until the card is unlocked. Once exitprocess is called IE has no control over thread closure.*

## Oplossingsrichtingen

Technisch zijn er 3 oplossingen: de applicatie, de browser en de middleware.

### 1. De applicatie

In de PC/SC standard bevat deel 7 de *Application Domain and Developer Design Considerations*. Paragraaf 3.3 gaat over *Device Sharing and Control*. Ook als er op hoog abstractieniveau ontwikkeld wordt –door gebruik te maken van third party libraries voor bijvoorbeeld SSL- is het van groot belang om de onderliggende PC/SC architectuur goed te begrijpen. Ook dient men te verifiëren dat de applicatierichtlijnen uit deze paragraaf ingevuld zijn, samengevat:

- Communiceer alleen met de pas als het nodig is;
- Geef geen reset naar de smartcard;
- Claim geen exclusieve toegang tot de smartcard behalve als dat noodzakelijk is;
- Beëindig connecties bij afsluiten van de applicatie.

Een mogelijkheid om het probleem vanuit een webapplicatie te voorkomen is met behulp van de Javascript onUnload Event Handler. De onUnload Event Handler maakt het mogelijk om bepaalde acties uit te voeren bij het verlaten van een webpagina of afsluiten van de browser. Zie: [http://www.codetoad.com/javascript/miscellaneous/onunload\\_event.asp](http://www.codetoad.com/javascript/miscellaneous/onunload_event.asp). Op die manier kan je bij het afsluiten van IE zorgen dat lopende communicatie eerst wordt afgerond.

### 2. De browser

In Firefox (getest 2.0.0.6) treedt het probleem niet op<sup>3</sup>.

Er zijn ook manieren om het afsluitgedrag van Internet Explorer aan te passen. Microsoft geeft de volgende mogelijkheden aan om het afsluitgedrag van IE te wijzigen:

*Another possibility is to hook or wait on the event of the window closing using a plugin or a Browser Helper Object to run code that will wait on the thread to complete before running ExitProcess. In this way it would mimic Firefox's behaviour.*

Zie voor Browser Helper Objects: The Browser the Way You Want It

<http://msdn2.microsoft.com/en-us/library/bb250436.aspx>

---

<sup>3</sup> Dit is ook bevestigd door analyse van Microsoft die het afsluiten van FireFox als volgt omschrijft:

*In firefox, the closure is dealt with in a different way.*

1. *Some threads begin to end. Can't see what they are doing, but they are being closed through the C function \_endthread.*
2. *The transmission thread (communication with smartcard) resumes and completes the transaction. It appears that a semaphore is used to block execution further on the main thread. The all important SCardEndTransaction is called.*
3. *The rest of the threads now begin to exit including the thread that was doing the transmit with the smartcard.*
4. *ScardDisconnect is called in the main thread.*
5. *All threads are gone, now we call ExitProcess.*

### 3. De middleware

Microsoft geeft aan dat in de BaseCryptoServiceProvider die zij zelf ontwikkelen het probleem niet optreedt onder andere omdat alle communicatie in de 'main thread' van de BaseCSP module plaatsvindt.

AET heeft onderzocht of de middleware is aan te roepen via een Windows service. Deze service zou de communicatie met de UZI-pas gecontroleerd afronden ook als IE wordt afgesloten tijdens een PC/SC transactie. Technisch is dit mogelijk, maar vanuit het UZI-register is besloten om dit niet verder uit te werken. Dit zou maatwerk introduceren waarvan de impact niet in verhouding staat tot de omvang van het probleem.

### **Conclusies**

Na een uitvoerige analyse en discussie met de betrokken leveranciers trekt het UZI-register de volgende conclusies:

- De betrokken partijen zijn het eens over wat er gebeurt.
- Het probleem is zeldzaam omdat het sterk samenhangt met gebruikers- en applicatiegedrag.
- Als applicatiebouwers zich bewust zijn van dit issue en de PC/SC richtlijnen in acht nemen zal dat meestal afdoende zijn. Mocht het toch optreden dan zijn er voldoende suggesties gedaan om een workaround te creëren.
- Zelfs binnen de PC/SC architectuur ([www.pcscworkgroup.com](http://www.pcscworkgroup.com)) blijft het lastig om exact te bepalen welke component verantwoordelijk is om dit te voorkomen. De standaard spreekt van 'ICC (smartcard) awareness' van een applicatie, maar IE is op zichzelf niet smartcard aware. Ook de precieze afbakening van een applicatie is lastig als deze bestaat uit bijvoorbeeld een java applet, JVM, browser en middleware. Het is daarom onwaarschijnlijk dat dit opgelost gaat worden in een update van IE of in SafeSign Identity Client.