Zugriffsschutz auf Funktionen der Bürgerkarten-Umgebung		Konvention 1.2.1 Empfehlung
Kurzbeschreibung	Dieses Dokument beschreibt die Art und Weise wie Funktionen der Bürgerkarten-Umgebung vor unbefugtem Zugriff geschützt werden können: • Schutzfunktionen beim Zugriff durch Aufruf der Bindungen TCP und TLS • Schutzfunktionen beim Zugriff durch Aufruf der Bindungen HTTP und HTTPS • Schutzfunktionen beim Zugriff über die Befehlskaskadierung der Bindungen HTTP und HTTPS	
Autoren:	Arno Hollosi Gregor Karlinger Thomas Rössler Martin Centner et al.	Projektteam/Arbeitsgruppe AG Bürgerkarte
Datum:	1.3.2005	

Inhaltsverzeichnis

1. Allgemeines

Dieses Dokument beschreibt die Art und Weise wie Funktionen der *Bürgerkarten-Umgebung* vor unbefugtem Zugriff geschützt werden können:

- Schutzfunktionen beim Zugriff durch Aufruf der Bindungen TCP und TLS
- Schutzfunktionen beim Zugriff durch Aufruf der Bindungen HTTP und HTTPS
- Schutzfunktionen beim Zugriff über die Befehlskaskadierung der Bindungen HTTP und HTTPS

1.1. Namenskonventionen

Zur besseren Lesbarkeit wurde in diesem Dokument auf geschlechtsneutrale Formulierungen verzichtet. Die verwendeten Formulierungen richten sich jedoch ausdrücklich an beide Geschlechter.

Folgende Namenraum-Präfixe werden in dieser Spezifikation zur Kennzeichnung der Namenräume von XML-Elementen verwendet:

Präfix	Namenraum	Erläuterung
`sl`	`http://www.buergerkarte.at/namespaces/securitylayer/1.2#`	Elemente der Applikationsschnittstelle Security-Layer

1.2. Schlüsselwörter

Dieses Dokument verwendet die Schlüsselwörter MUSS, DARF NICHT, ERFORDERLICH, SOLLTE, SOLLTE NICHT, EMPFOHLEN, DARF, und OPTIONAL zur Kategorisierung der Anforderungen. Diese Schlüsselwörter sind analog zu ihren englischsprachigen Entsprechungen MUST, MUST NOT, REQUIRED, SHOULD, SHOULD NOT, RECOMMENDED, MAY, und OPTIONAL zu handhaben, deren Interpretation in [Keywords] festgelegt ist.

2. Klassifizierung des Zugriffs

2.1. Authentisierungsklassen

Ein Zugriff auf einen Befehl der *Bürgerkarte* kann hinsichtlich der Authentisierung in eine der vier Klassen *anonym*, *pseudoanonym*, *certified* und *certifiedGovAgency* eingeteilt werden. Zur Vornahme der Einteilung in diese Klassen sind folgende Fragen zu beantworten:

Kann der Ursprung der Befehlsanfrage festgestellt werden?

> Zur Beantwortung dieser Frage werden Parameter aus den Bindungsprotokollen wie z.B. die Quell-IP-Adresse, der HTTP-Header `Referer`, der Parameter `DataURL` der HTTP-Bindung, oder Informationen aus verwendeten Client- oder Serverzertifikaten verwendet.

Kann das Ziel der Befehlsantwort festgestellt werden?

> Zur Beantwortung dieser Frage werden ebenfalls Parameter aus den Bindungsprotokollen wie z.B. Quell-IP-Adresse, der Parameter `DataURL` der HTTP-Bindung, oder Informationen aus verwendeten Client- oder Serverzertifikaten verwendet.

Die vier Authentisierungsklassen lassen sich vereinfacht wie folgt charakterisieren:

anonym

> Die *Bürgerkarten-Umgebung* hat weder Informationen über den Ursprung der Befehlsanfrage noch über das Ziel der Befehlsantwort.

pseudoanonym

> Die *Bürgerkarten-Umgebung* besitzt zwar Informationen über den Ursprung der Befehlsanfrage und/oder über das Ziel der Befehlsantwort, diese Informationen basieren jedoch auf nicht gesicherten Parametern der Bindungen.

certified

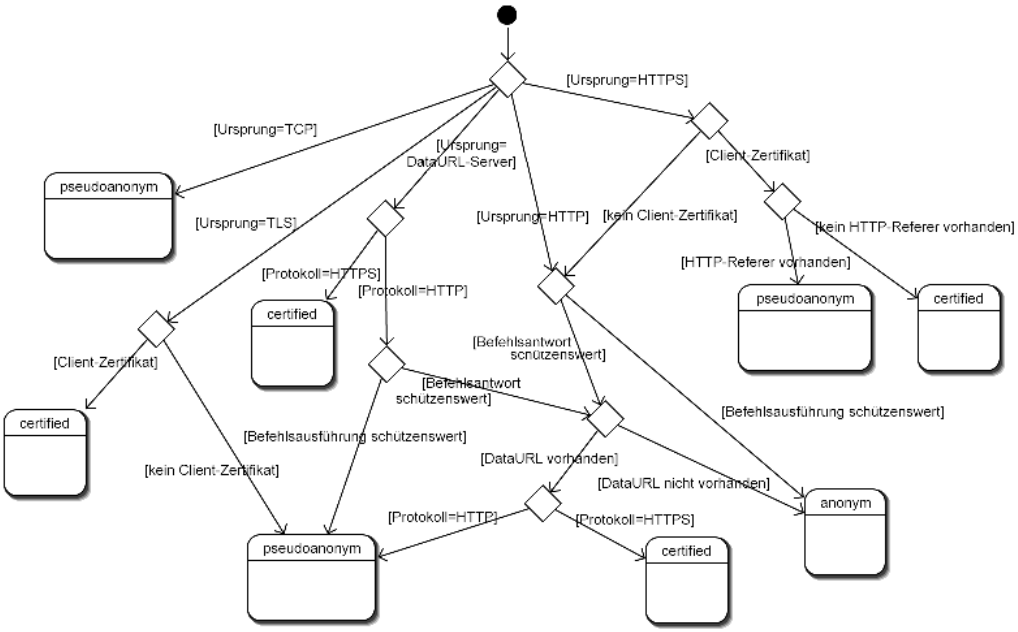
> Die *Bürgerkarten-Umgebung* hat gesicherte (zertifikatsbasierende) Informationen über den Ursprung der Befehlsanfrage und/oder das Ziel der Befehlsantwort.

certifiedGovAgency

> Die *Bürgerkarten-Umgebung* hat gesicherte (zertifikatsbasierende) Informationen über den Ursprung der Befehlsanfrage und/oder das Ziel der Befehlsantwort. Aus diesen Informationen geht hervor, dass es sich bei Ursprung bzw. Ziel um eine Behörde oder einen Dienstleister der Behörde handelt, d.h. der Domainname, auf den das Zertifikat ausgestellt ist, matched das Pattern `*.gv.at`, oder das Zertifikat enthält entweder die Zertifikatserweiterung *Verwaltungseigenschaft* oder die Zertifikatserweiterung *Dienstleistereigenschaft* [VerwEig].

Die nachfolgende Grafik enthält den Entscheidungsbaum zur Zuordnung eines Zugriffs auf einen Befehl der *Bürgerkarten-Umgebung* in eine dieser vier Authentisierungsklassen. Aus Gründen der Übersichtlichkeit wurde in in der Grafik auf die weitere Unterscheidung zwischen *certified* und *certifiedGovAgency* verzichtet.

Abbildung 1. Authentisierungsklassen



Zunächst ist nach der Herkunft der Befehlsanfrage zu unterscheiden. Eine Befehlsanfrage kann aus einer der vier Bindungen TCP, TLS, HTTP und HTTPS oder vom `DataURL`-Server (Server, der mit dem Parameter `DataURL` der HTTP-Bindung bezeichnet wird, und von dem im Zuge der Befehlskaskadierung weitere Befehlsanfragen an die *Bürgerkarten-Umgebung* übermittelt werden können) eintreffen.

Kommt eine Befehlsanfrage aus der Bindung TCP, fällt der Zugriff in die Klasse *pseudoanonym*, als Identifikations-URL wird die Source-IP-Adresse der TCP-Verbindung vermerkt.

Entspringt eine Befehlsanfrage aus der Bindung TLS, ist in einem weiteren Schritt zu prüfen, ob sich in der TLS-Verbindung zur *Bürgerkarten-Umgebung* der Anfrager mittels Client-Zertifikat identifiziert hat. Falls ja, fällt der Zugriff in die Klasse *certified*, falls nein, fällt er (analog zur Bindung TCP) in die Klasse *pseudoanonym*. In beiden Fällen wird als Identifikations-URL die Source-IP-Adresse der TLS-Verbindung vermerkt.

Stammt eine Befehlsanfrage aus der Bindung HTTP, ist in einem weiteren Schritt zu prüfen, ob bereits die Ausführung des Befehls schützenswert ist oder lediglich die Befehlsantwort (vergleiche dazu Abschnitt 2.3, „Einstufung der Befehle“). Ist bereits die Befehlsausführung schützenswert, ist der Zugriff der Klasse *anonym* zuzuordnen, da diesfalls für die Zuordnung des Zugriffs der Ursprung der Befehlsanfrage maßgeblich ist. Die IP-Adresse der Befehlsanfrage ist zwar bekannt, jedoch muss in den allermeisten Fällen davon ausgegangen werden, dass die Befehlsanfrage über die Bindung HTTP vom Browser des *Bürgers* kommt, und dadurch der Browser sozusagen als Proxy agiert und die tatsächliche Herkunft der Befehlsanfrage überdeckt. Als Identifikations-URL wird die Source-IP-Adresse der HTTP-Verbindung vermerkt. Ist hingegen die Befehlsantwort schützenswert, ist das Ziel der Befehlsantwort maßgeblich für die Zuordnung des Zugriffs. Es ist daher in einem nächsten Schritt zu prüfen, ob der Parameter `DataURL` dieser Bindung verwendet wird. Falls nein, liegt keine Information über das Ziel der Befehlsantwort vor; der Zugriff fällt in die Klasse *anonym*, als Identifikations-URL wird die Source-IP-Adresse der HTTP-Verbindung vermerkt. Falls ja, ist in einem abschließenden Schritt das Protokoll des Parameters `DataURL` zu analysieren. Handelt es sich dabei um `http`, fällt der Zugriff in die Klasse *pseudoanonym*, handelt es sich hingegen um `https`, ist der Zugriff der Klasse *certified* zuzuordnen, da die Information über das Ziel der Befehlsantwort über das Zertifikat des Servers hinter der `DataURL` gesichert ist. In beiden Fällen wird als Identifikations-URL die `DataURL` vermerkt.

Erreicht die Befehlsanfrage die *Bürgerkarten-Umgebung* über die Bindung HTTPS, ist in einem weiteren Schritt zu prüfen, ob sich in der darunterliegenden TLS-Verbindung der Anfrager mittels Client-Zertifikat identifiziert hat. Falls nein, sind beginnend mit der Prüfung, ob die Befehlsausführung oder die Befehlsantwort schützenswert sind, die selben weiteren Prüfungen wie bei der Bindung HTTP anzuwenden. Wurde hingegen das Client-Zertifikat verwendet, ist der HTTP-Header `Referer` zur Feststellung des eigentlichen Ursprungs der Befehlsanfrage auszuwerten. Ist dieser Header vorhanden, ist der Zugriff der Klasse *pseudoanonym* zuzuordnen, wobei als Identifikations-URL der Wert des Headers festgehalten wird. Fehlt der Header, ist der Zugriff der Klasse *certified* zuzuordnen, wobei dann als Identifikations-URL die Source-IP-Adresse der HTTPS-Verbindung vermerkt wird.

Wird die Befehlsanfrage über die Befehlskaskadierung der Bindungen HTTP bzw. HTTPS vom `DataURL`-Server an die *Bürgerkarten-Umgebung* gerichtet, ist in einem weiteren Schritt das Protokoll jener URL zu analysieren, über welche die *Bürgerkarten-Umgebung* den `DataURL`-Server zuvor kontaktiert hat. Handelt es sich dabei um `https`, fällt der Zugriff in die Klasse *certified*, da dann über das Zertifikat des `DataURL`-Servers gesichert der Ursprung der Befehlsanfrage sichergestellt werden kann. Als Identifikations-URL wird die `DataURL` vermerkt. Handelt es sich hingegen um `http`, ist in einem weiteren Schritt zu prüfen, ob bereits die Ausführung des Befehls schützenswert ist oder lediglich die Befehlsantwort (vergleiche dazu Abschnitt 2.3, „Einstufung der Befehle“). Ist bereits die Befehlsausführung schützenswert, ist der Zugriff der Klasse *pseudoanonym* zuzuordnen, da diesfalls für die Zuordnung des Zugriffs der Ursprung der Befehlsanfrage maßgeblich ist. Auch in diesem Fall wird als Identifikations-URL die `DataURL` vermerkt. Ansonsten verlaufen die weiteren Prüfungen wie im Falle der Bindung HTTP, beginnend mit der Prüfung, ob der Parameter `DataURL` der Bindung verwendet wird.

2.2. Bedingungen an Zertifikate

Die TLS- und HTTPS-Bindung sowie die Verbindung der *Bürgerkarten-Umgebung* zum `DataURL`-Server, falls eine `DataURL` mit Protokoll `https` verwendet wird, basieren auf dem Protokoll TLS, welches zum Zwecke der Authentisierung von Client und Server Zertifikate nach X.509 [X509] verwendet.

Ein verwendetes Zertifikat ist in allen drei Fällen für die Zuordnung des Zugriffs zu einer der Authentisierungsklassen nur dann relevant, wenn es von einem Zertifizierungsdiensteanbieter stammt, der von der *Bürgerkarten-Umgebung* als vertrauenswürdig eingestuft wird.

2.3. Einstufung der Befehle

Die nachfolgende Tabelle gibt für jeden der Befehle der Schnittstelle *Security-Layer* an, ob die Befehlsausführung oder die Befehlsantwort

schützenswert ist (vergleiche dazu den Entscheidungsbaum zur Einteilung eines Zugriffs in Abschnitt 2.1, „Authentisierungsklassen“).

Befehl	**schützenswert**
Signatur nach CMS erstellen (`sl:CreateCMSSignatureRequest`)	Befehlsantwort
Signatur nach XMLDSIG erstellen (`sl:CreateXMLSignatureRequest`)	Befehlsantwort
Signatur nach CMS prüfen (`sl:VerifyCMSSignatureRequest`)	Befehlsantwort
Signatur nach XMLDSIG prüfen (`sl:VerifyXMLSignatureRequest`)	Befehlsantwort
Abfrage verfügbarer Infoboxen (`sl:InfoboxAvailableRequest`)	Befehlsantwort
Lesen von Infobox-Daten (`sl:InfoboxReadRequest`)	Befehlsantwort
Verändern von Infoboxdaten (`sl:InfoboxUpdateRequest`)	Befehlsausführung
Anlegen einer Infobox (`sl:InfoboxCreateRequest`)	Befehlsausführung
Löschen einer Infobox (`sl:InfoboxDeleteRequest`)	Befehlsausführung
Abfrage der Kapseleigenschaften (`sl:GetPropertiesRequest`)	Befehlsantwort
Abfrage des Kartenstatus (`sl:GetStatusRequest`)	Befehlsantwort
NOP (`sl:NullOperationRequest`)	Befehlsantwort
Hashwert-Berechnung (`sl:CreateHashRequest`)	Befehlsantwort
Hashwert-Verifikation (`sl:VerifyHashRequest`)	Befehlsantwort
Verschlüsselung als CMS-Nachricht (`sl:EncryptCMSRequest`)	Befehlsantwort
Verschlüsselung als XML-Nachricht (`sl:EncryptXMLRequest`)	Befehlsantwort
Entschlüsselung einer CMS-Nachricht (`sl:DecryptCMSRequest`)	Befehlsantwort
Entschlüsselung einer XML-Nachricht (`sl:DecryptXMLRequest`)	Befehlsantwort

3. Zugriffsschutz

Implementoren steht es frei, welches Regelwerk in welcher Granularität sie basierend auf der vorgestellten Klassifizierung implementieren. Der Schutz des im Folgenden vorgestellten Regelwerks MUSS jedoch jedenfalls erfüllt sein.

3.1. Regeln

Regeln beschreiben eine Verknüpfung von Eingangsdaten und einer daraus resultierende Aktion bzw. Benutzerinteraktion. Regeln werden in Chains organisiert und ihrer Reihenfolge nach abgearbeitet. Die erste Regel, die zutrifft wird ausgeführt. Diese einfache Priorisierung von Regeln ist für die Zwecke des Zugriffsschutzes ausreichend.

3.1.1. Eingangsdaten

Authentisierungsklasse

Die Authentisierungsklasse, welche für ein Zutreffen der Regel mindestens erfüllt sein muss. Gültige Werte sind: `anonym`, `pseudoanonym`, `certified`, `certifiedGovAgency` (vergleiche Abschnitt 2.1, „Authentisierungsklassen“).

Identifikationsbegriff

Dieses Eingangsdatum muss die Identifikations-URL (vergleiche Abschnitt 2.1, „Authentisierungsklassen“) der Authentisierungsklasse (Ursprung der Befehlsanfrage bzw. Ziel der Befehlsantwort) matchen. Mögliche Werte sind Domänennamen, IP-Adressen und URLs. Ein Domänenname bzw. eine IP-Adresse muss den Server-Teil der Identifikations-URL matchen, eine URL die gesamte Identifikations-URL. Auch hier können Wildcards eingesetzt werden:

- `*` matched jeden beliebigen Wert;
- für Domänennamen ist die einmalige Angabe einer Wildcard `*` für ein oder mehrere Domänenteile am Beginn des Patterns (z.B. `*.gv.at` oder `*.cio.gv.at`, nicht aber `*io.gv.at`) erlaubt;
- für IP-Adressen ist die einmalige Angabe einer Wildcard `*` für ein oder mehrere Bytes der IP-Adresse am Ende des Patterns (z.B. `193.170.*` oder `193.170.251.*`, nicht aber `193.170.25*`) erlaubt;
- Für URLs ist die einmalige Angabe einer Wildcard am Ende der URL erlaubt (z.B. `https://finanzonline.bmf.gv.at/arbeitnehmerveranlagung/*`).

Befehlsname

Der lokale Name der Befehlsanfrage der Schnittstelle *Security-Layer* (z.B. `InfoboxReadRequest`). Ein Teil des Namens mit abschließender Wildcard `'*'` (z.B. `Infobox*` für alle Infobox-Befehle), oder eine Wildcard `'*'` für alle Befehle ist zulässig.

Befehlsparameter

Abhängig vom jeweiligen Befehl der Schnittstelle *Security-Layer* sind ein oder mehrere Parameter anzugeben. Bei Infobox-Befehlen gibt der Parameter *InfoboxIdentifier* den Namen der behandelten Infobox an. Bei Befehlen, die auf private Schlüssel zugreifen, gibt der Parameter *KeyboxIdentifier* den Name der verwendeten Keybox an. Beim Auslesen der Infoboxen *IdenitityLink* und *Mandates* gibt der Parameter *PersonIdentifier* an, ob die enthaltenen Stammzahlen natürlicher Personen unverändert (`base`) übermittelt, oder durch die entsprechenden abgeleiteten bereichsspezifischen Personenkennzeichen (`derived`) ersetzt werden würden. Die Verwendung einer einfachen Wildcard `'*'` für die Parameter *InfoboxIdentifier*, *KeyboxIdentifier* und *Identifier* ist zulässig.

3.1.2. Aktionen

Gültige Aktionen für Regeln sind *allow*, *deny*, und Name einer weiteren Chain. *Allow* bedeutet, dass der Funktionsaufruf zugelassen und durchgeführt werden soll. *Deny* bedeutet, dass der Funktionsaufruf nicht zugelassen und nicht ausgeführt werden soll. Dar Name einer weiteren Chain bedeutet, dass die Regelabarbeitung mit der ersten Regel der bezeichneten Chain fortgesetzt werden soll.

Für die Aktionen *allow* und *deny* muss zusätzlich die Art der Interaktion über die *Benutzer-Schnittstelle* festgelegt werden.

3.1.3. Benutzerinteraktion

Die Art der Interaktion gibt an, wie der Benutzer in die Befehlsabarbeitung eingebunden werden soll. Im Kontext der Regeln wird damit nur die mindestens durchzuführende Interaktion festgelegt. Abhängig von der abzuarbeitenden Funktion können höherwertige Interaktionen notwendig sein. Es gibt vier Interaktionstypen: `none`, `info`, `confirm`, `confirmWithSecret`. `none` bedeutet, dass die auszuführende Funktion vom *Bürger* gar nicht bestätigt werden muss; `info` bedeutet, dass der *Bürger* über die auszuführende Aktion über die *Benutzer-Schnittstelle* informiert werden MUSS; `confirm` bedeutet, dass der *Bürger* über die *Benutzer-Schnittstelle* die Erlaubnis für die Ausführung bestätigen MUSS; bei `confirmWithSecret` MUSS der *Bürger* die Erlaubnis für die Ausführung durch die Eingabe eines Passworts über die *Benutzer-Schnittstelle* erteilen.

3.2. Chains

Mindestens zwei Chains MÜSSEN vorhanden sein: `Identification` und `Command`. Die Abarbeitung beginnt immer mit der Chain `Identification`. Die Aufteilung ermöglicht es, zuerst den Ursprung der Befehlsanfrage bzw. das Ziel der Befehlsantwort einzugrenzen, danach erfolgt eine weitere Eingrenzung auf Kommandobasis.

3.3. Voreinstellungen für eine lokale Bürgerkarten-Umgebung

Bei der Auslieferung einer lokalen *Bürgerkarten-Umgebung* an den *Bürger* muss ihr Regelwerk so konfiguiert sein, dass es den nachfolgenden Standardeinstellungen genügt. Die *Bürgerkarten-Umgebung* sollte dem *Bürger* die Möglichkeit bieten, die Standardeinstellungen nach seinen persönlichen Wünschen zu verändern.

3.3.1. Chain `Identification`

Regel- nummer	Authentisierungs- klasse	Identifikations- begriff	Aktion	Benutzer- interaktion
1	`certifiedGovAgency`	*	*allow*	*confirm*
2	`pseudoanonym`	*	Chain `Command`	-
3	`anonym`	`127.0.0.1`	Chain `Command`	-
4	`anonym`	*	deny	info

3.3.2. Chain `Command`

Regel-nummer	Authentisierungs-klasse	Identifikations-begriff	Befehlsname	Befehlsparameter	Aktion	Benutzer-interaktion
1	certified	*	`InfoboxReadRequest`	`InfoboxIdentifier="IdenitityLink" PersonIdentifier="derived"`	*allow*	*confirm*
2	certified	*	`InfoboxReadRequest`	`InfoboxIdentifier="Mandates" PersonIdentifier="derived"`	*allow*	*confirm*
3	`anonym`	*	`Infobox*`	`InfoboxIdentifier="IdenitityLink" PersonIdentifier="*"`	deny	info
4	`anonym`	*	`Infobox*`	`InfoboxIdentifier="Mandates" PersonIdentifier="*"`	deny	info
5	`anonym`	*	`Infobox*`	*	*allow*	*confirm*
6	`anonym`	*	`GetPropertiesRequest`	*	*allow*	none
7	`anonym`	*	`GetStatusRequest`	*	*allow*	*none*
8	`anonym`	*	`NullOperationRequest`	*	*allow*	*none*
9	`anonym`	*	`CreateHashRequest`	*	*allow*	info
10	`anonym`	*	`VerifyHashRequest`	*	*allow*	*info*
11	`anonym`	*	*	*	*allow*	*confirm*

3.4. Voreinstellungen für eine serverbasierte Bürgerkarten-Umgebung

Bei der erstmaligen Benutzung einer serverbasierten *Bürgerkarten-Umgebung* durch den *Bürger* muss ihr Regelwerk so konfiguiert sein, dass es den nachfolgenden Standardeinstellungen genügt. Die *Bürgerkarten-Umgebung* sollte dem *Bürger* die Möglichkeit bieten, die Standardeinstellungen nach seinen persönlichen Wünschen zu verändern.

3.4.1. Chain `Identification`

Regel- nummer	Authentisierungs- klasse	Identifikations- begriff	Aktion	Benutzer- interaktion
1	`certifiedGovAgency`	*	*allow*	*confirm*
2	`pseudoanonym`	*	Chain `Command`	-
3	`anonym`	*	deny	info

3.4.2. Chain `Command`

Siehe Abschnitt 3.3.2, „Chain `Command`“.

4. Sicherheits- und Bedrohungsanalyse

4.1. Schutzbedürfnis der Signaturerstellungsbefehle

Die Einstufung, dass bei den Signaturerstellungsbefehlen nur das Resultat, nicht aber die Aktion schützenswert ist, erfolgt aus pragmatischen Gründen. Prinzipiell ist es also möglich, dass Signaturerstellungsbefehle von unbekannter Quelle abgesetzt werden und an eine bekanntes Zielübermittelt werden. Da allerdings ein potentieller Angreifer das Signaturresultat für eine sinnvolle Attacke benötigen würde, erscheint diese Einstufung gerechtfertigt.

4.2. Cross-Site-Scripting Attacken

Mit Hilfe einer Cross-Site-Scripting (XSS)-Attacke (vergleiche [XSS-FAQ]) kann ein Angreifer der *Bürgerkarten-Umgebung* unter Verwendung der Bindung HTTPS mit Client-Authentisierung einen falschen Ursprung der Befehlsanfrage vortäuschen:

Der Angreifer verleitet den Bürger dazu, einen manipulierten Link auf eine Seite auszuführen, die im Regelwerk der *Bürgerkarten-Umgebung* als vertrauenswürdig eingestuft ist (z.B. die Seite einer Behörde). Dieser Link enthält als Parameter JavaScript-Elemente, die einen HTTPS-Post zur *Bürgerkarten-Umgebung* ausführen sollen. Ist die Behördenseite tatsächlich anfällig für eine XSS-Attacke, befinden sich diese JavaScript-Elemente dann in jener Seite, die der *Bürger* von der eigentlich vertrauenswürdigen Seite als Antwort auf die Verfolgung des Links erhält und werden nach dem Laden der Seite im Browser ausgeführt. Die *Bürgerkarten-Umgebung* erhält nun über die HTTPS-Bindung eine Befehlsanfrage. Entsprechend dem Entscheidungsbaum aus Abschnitt 2.1, „Authentisierungsklassen“ prüft die *Bürgerkarten-Umgebung* den HTTP Header *Referer*, falls sich der Browser des *Bürgers* mittels Client-Zertifikat identifiziert hat. Dieser Header verweist auf die als vertrauenswürdig eingestufte Seite, daher wird der Zugriff der Klasse *pseudoanonym* zugeordnet und als Identifikations-URL fälschlicherweise die vertrauenswürdige Seite vermerkt (obwohl der Befehl an die *Bürgerkarten-Umgebung* eigentlich aus dem eingeschmuggelten JavaScript des Angreifers stammt).

Nicht zuletzt aus diesem Grund sind die Standardeinstellungen für den Zugriffsschutz in den Abschnitten Abschnitt 3.3, „Voreinstellungen für eine lokale Bürgerkarten-Umgebung“ und Abschnitt 3.4, „Voreinstellungen für eine serverbasierte Bürgerkarten-Umgebung“ so gewählt, dass für alle sensiblen Befehle jedenfalls die Bestätigung des *Bürgers* einzuholen ist.

Glossar

Glossar

Applikation

Jenes Programm, das Anfragen an die *Bürgerkarten-Umgebung* über den *Security-Layer* richtet und die entsprechenden Antworten entgegennimmt und auswertet.

Benutzer-Schnittstelle

Jene Schnittstelle, über die der *Bürger* mit der *Bürgerkarten-Umgebung* kommuniziert. Über diese Schnittstelle wird einerseits die Benutzerinteraktion abgewickelt, die gegebenenfalls zur Abwicklung eines Befehls des *Security-Layers* notwendig ist (z.B. die Anzeige eines zu signierenden Dokuments beim Befehl zur Erzeugung einer XML-Signatur); andererseits kann der *Bürger* über diese Schnittstelle seine *Bürgerkarten-Umgebung* nach seinen persönlichen Bedürfnissen konfigurieren (z.B. kann er Einstellungen zum Zugriffsschutz auf seine Infoboxen verändern). Die Vorgaben an die *Benutzer-Schnittstelle* sind in Minimale Umsetzung des Security-Layers geregelt.

Bürger

Jene Person, die die Funktionen der *Bürgerkarten-Umgebung* für die sichere Abwicklung von E-Government oder E-Commerce verwenden möchte. Die Ansteuerung der *Bürgerkarten-Umgebung* erfolgt in der Regel nicht durch den *Bürger* selbst, sondern durch die *Applikation*, welche die E-Government oder E-Commerce Anwendung repräsentiert.

Bürgerkarte

Laut [E-GovG], §10 ZI 10 ist die *Bürgerkarte* „die unabhängig von der Umsetzung auf unterschiedlichen technischen Komponenten gebildete logische Einheit, die eine elektronische Signatur mit einer Personenbindung (§ 4 Abs. 2) und den zugehörigen Sicherheitsdaten und -funktionen sowie mit allenfalls vorhandenen Vollmachtsdaten verbindet“. Im Sinne der in den Spezifikationen zur österreichischen Bürgerkarte gebrauchten Terminologie ist die *Bürgerkarten-Umgebung* die Implementierung der logischen Einheit *Bürgerkarte*.

Bürgerkarten-Umgebung

Jenes Programm bzw. jener Dienst, der die Funktionalität der *Bürgerkarte* zur Verfügung stellt. Grundsätzlich vorstellbar ist die Ausführung als Programm, das lokal am Rechner des *Bürgers* läuft (*lokale Bürgerkarten-Umgebung*), oder als serverbasierter Dienst, der über das Internet angesprochen wird (*serverbasierte Bürgerkarten-Umgebung*). Die Interaktion mit diesem Programm bzw. Dienst wird über zwei Schnittstellen abgewickelt: Über die *Benutzer-Schnittstelle* sowie über den *Security-Layer*.

Hash-Eingangsdaten

Jene Daten, die für die Berechnung des Hash-Wertes für eine `dsig:Reference` verwendet werden. Sind für die `dsig:Reference` Transformationen angegeben, entsprechen diese Daten dem Ergebnis der letzten Transformation. Sind keine Transformationen spezifiziert, gleichen die Hash-Eingangsdaten den *Referenz-Eingangsdaten*.

Impliziter Transformationsparameter

Siehe Abschnitt 2.2.2.2, „Implizite Transformationsparameter“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

Referenz-Eingangsdaten

Jene Daten, die sich aus der Auflösung der im Attribut `URI` der `dsig:Reference` angegebenen URI ergeben. Sind für die `dsig:Reference` Transformationen angegeben, werden diese Daten als Eingangsdaten zur Berechnung der ersten Transformation verwendet. Sind keine Transformationen spezifiziert, gleichen die Referenz-Eingangsdaten den *Hash-Eingangsdaten*.

Security-Layer

Jene Schnittstelle, über die die *Applikation* mit der *Bürgerkarten-Umgebung* kommuniziert. Das genaue Protokoll, das über diese Schnittstelle gesprochen werden kann, wird in Applikationsschnittstelle Security-Layer spezifiziert. Die möglichen Bindungen dieses Protokolls an Transportschichten wie HTTP oder TCP wird in Transportprotokolle Security-Layer geregelt.

Signaturmanifest

Siehe Abschnitt 2.2.2.2, „Implizite Transformationsparameter“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-

Layer .

Referenzen

[CMS] BHously, R.: *RFC 3369: Cryptographic Message Syntax (CMS)* , IETF Request For Comment, August 2002

[CMS-AES] chaad, J.: *RFC 3565: Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS)* . IETF Request For Comment, Juli 2003.

[CMS-Alg] Hously, R.: *RFC 3370: Cryptographic Message Syntax (CMS) Algorithms* . IETF Request For Comment, August 2002.

[CMS-RSAES-OAEP] Hously, R.: *RFC 3560: Use of the RSAES-OAEP Key Transport Algorithm in the Cryptographic Message Syntax (CMS)* . IETF Request For Comment, Juli 2003.

[CSS 2] Bert Bos, Håkon Wium Lie, Chris Lilley und Ian Jacobs: *Cascading Style Sheets, level 2* . W3C Recommendation, Mai 1998.

[EC14N] Boyer, John, Eastlake, Donald und Reagle, Joseph: *Exclusive XML Canonicalization. W3C Recommendation, Juli 2002* .

[ECDSA-CMS] Blake-Wilson, S., Brown, D., Lampert, D.: *RFC 3278: Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)* . IETF Request For Comment, April 2002.

[ECDSA-XML] Blake-Wilson, S., Karlinger, G. und Wang, Y.: *ECDSA with XML-Signature Syntax* . Internet-Draft, Jänner 2004.

[E-GovG] *BGBl. I Nr. 10/2004.*

[ESS-S/MIME] Hoffman, P.: *RFC 2634: Enhanced Security Services for S/MIME* , IETF Request For Comment, Juni 1999

[ETSICMS] European Telecommunications Standards Institute: *ETSI TS 101733: Electronic Signature Formats, v1.5.1* , Technical Specification, Dezember 2003

[ETSIQCert] European Telecommunications Standards Institute: *ETSI TS 101 862: Qualified certificate profile, v1.2.1* , Technical Specification, Juni 2001

[ETSIXML] European Telecommunications Standards Institute: *ETSI TS 101903: XML Advanced Electronic Signatures (XAdES), v1.2.2* , Technical Specification, April 2004

[GIF] *Graphics Interchange Format, Version 89a* . CompuServe Incorporated, Juli 1990.

[HTML4] Dave Ragget, Arnaud Le Hors und Ian Jacobs: *HTML 4.01 Specification* . W3C Recommendation, Dezember 1999.

[HTTP1.1] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leech und T. Berners-Lee: *Hypertext Transfer Protocol -- HTTP/1.1*. IETF Request For Comment, Juni 1999.

[HTTPS] E. Rescorla *HTTP over TLS*. IETF Request For Comment, Mai 2000

[ISO-8859-1] *ISO/IEC 8859-1*:1998: Information technology -- 8-bit single-byte coded graphic character sets -- Part 1: Latin alphabet No. 1.

[ISO-8859-10] *ISO/IEC 8859-10*:1998: Information technology -- 8-bit single-byte coded graphic character sets -- Part 10: Latin alphabet No. 6.

[ISO-8859-15] *ISO/IEC 8859-15*:1999: Information technology -- 8-bit single-byte coded graphic character sets -- Part 15: Latin alphabet No. 9.

[ISO-8859-2] *ISO/IEC 8859-2*:1999: Information technology -- 8-bit single-byte coded graphic character sets -- Part 2: Latin alphabet No. 2.

[ISO-8859-3] *ISO/IEC 8859-3*:1999: Information technology -- 8-bit single-byte coded graphic character sets -- Part 3: Latin alphabet No. 3.

[ISO-8859-9] *ISO/IEC 8859-9*:1999: Information technology -- 8-bit single-byte coded graphic character sets -- Part 9: Latin alphabet No. 5.

[JPEG] Eric Hamilton: *JPEG File Interchange Format, Version 1.02* . C-Cube Microsystems, September 1992.

[KEYWORDS] Bradner, S.: *RFC 2119: Key words for use in RFCs to Indicate Requirement Levels* , IETF Request For Comment, März 1997

[MIME] Freed, N. und Borenstein, N.: *RFC 2046: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types* , IETF Request For Comment, November 1996

[PersBin] Hollosi, Arno und Karlinger, Gregor: *XML-Definition der Personenbindung* . Konvention zum E-Government Austria erarbeitet von der Stabsstelle IKT-Strategie des Bundes, Technik und Standards. Öffentlicher Entwurf, Version 1.2.2, 14. Februar 2005.

[PersonData] Naber, Larissa: *PersonData Struktur - XML Spezifikation* . Konvention zum E-Government Austria erarbeitet von der Arbeitsgruppe Kommunikationsarchitekturen. Öffentlicher Entwurf, Version 2.0.0, 14. Oktober 2004.

[PKCS#12] RSA Laboratories: *PKCS#12 v1.0: Personal Information Exchange Syntax* , Juni 1999.

[port-numbers] Internet Assigned Numbers Authority: *Port Numbers*

[QCert] Santesson, S. und Nystrom M.: *RFC 3739: Internet X.509 Public Key Infrastructure: Qualified Certificates Profile* , IETF Request For Comment, März 2004

[SigG] *BGBl I Nr. 190/1999* idF BGBl I Nr. 152/2001.

[SigV] *BGBl II Nr. 30/2000* idF BGBl II Nr. 527/2004.

[Stammzahl] Hollosi, Arno und Hörbe, Rainer: *Bildung von Stammzahl und bereichsspezifischem Personenkennzeichen (bPK)* . Konvention zum E-Government Austria erarbeitet von der Stabsstelle IKT-Strategie des Bundes, Technik und Standards sowie vom Bundesminsterium für Inneres. Öffentlicher Entwurf, Version 1.0, 2. Februar 2004.

[TLS] T. Dierks und C. Allen: *The TLS Protocol Version 1.0* . IETF Request For Comment, Januar 1999.

[Unicode] The Unicode Consortium. *The Unicode Standard, Version 4.0.0* , defined by: The Unicode Standard, Version 4.0 (Boston, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1).

[URI] Berners-Lee, T. , Fielding, R. und Masinter, L.: *RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax* , IETF Request For Comment, August 1998

[VerwEig] Hollosi, Arno: *X.509 Zertifikatserweiterungen für die Verwaltung* . Konvention zum E-Government Austria erarbeitet von der Stabsstelle IKT-Strategie des Bundes, Technik und Standards. Öffentlicher Entwurf, Version 1.0.3, 21. Februar 2005.

[X509] Polk, W., Ford, W., Solo, D.: *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile* . IETF Request For Comment, April 2002.

[XHTML 1.1] Murray Altheim, Frank Boumphrey, Sam Dooley, Shane McCarron, Sebastian Schnitzenbaumer und Ted Wugofski: *Modularization of XHTML* . W3C Recommendation, April 2001.

[XHTML MOD] Daniel Austin, Subramanian Peruvemba, Shane McCarron, Masayasu Ishikawa: *Modularization of XHTML in XML Schema* . W3C Working Draft, Oktober 2003.

[XML] Bray, Tim, Paoli, Jean, Sperberg-McQueen, C.M. und Maler, Eve: *Extensible Markup Language (XML) 1.0 (Second Edition)* , W3C Recommendation, Oktober 2000.

[XMLDecTF] Hughes, Merlin, Imamura, Takeshi und Maruyama, Hiroshi: *Decryption Transform for XML Signature* . W3C Recommendation, Dezember 2002.

[XMLDSIG] Eastlake, Donald, Reagle, Joseph und Solo, David: *XML-Signature Syntax and Processing* , W3C Recommendation, Februar 2002

[XMLDSIG-URI] Eastlake, Donald: *RFC 4051: Additional XML Security Uniform Resource Identifiers (URIs)* , IETF Request For Comments, April 2005

[XMLEnc] Eastlake, Donald und Reagle, Joseph: *XML Encryption Syntax and Processing* , W3C Recommendation, Dezember 2002

[XML-Schema] Thompson, Henry S., Beech, David, Maloney, Murray und Mendelson, Noah: *XML Schema Part 1: Structures* , W3C Recommendation, Mai 2001

[XMLTYPE] Murata, M., St.Laurent, S., und Kohn, D.: *RFC 3023: XML Media Types* , IETF Request For Comment, Jänner 2001.

[XPath] Clark, James und DeRose, Steven: *XML Path Language* , W3C Recommendation, November 1999

[XPF2] Boyer, John, Hughes, Merlin und Reagle, Joseph: *XML-Signature XPath Filter 2.0* . W3C Candidate Recommendation, Juli 2002.

[XPointer] Grosso, Paul, Maler, Eve, Marsh, Jonathan und Walsh, Norman: *XPointer Framework* . W3C Recommendation, März 2003.

[XSS-FAQ] Cgisecurity.com: *The Cross Site Scripting FAQ* .

5. 6 Historie

Datum	Version	Änderungen
01. 03. 2005	1.2.1	• Erratum Erratum 28 in Die österreichische Bürgerkarte - Errata korrigiert.
14. 05. 2004	1.2.0	• Erstellt.