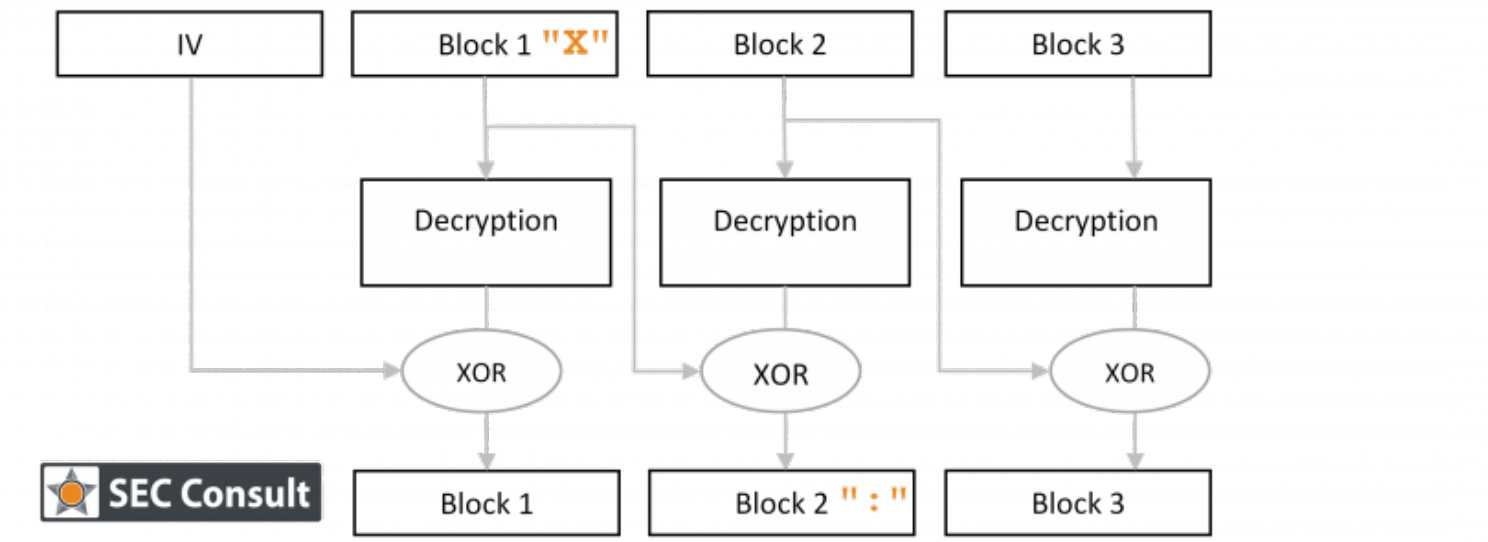
SEC Consult

BLOG



KRYPTOGRAPHISCHE SCHWACHSTELLEN IN DEUTSCHER EGOVERNMENT SOFTWAREKOMPONENTE < /BLOG/2019/02/KRYPTOGRAPHISCHE-SCHWACHSTELLEN-IN-DEUTSCHER-EGOVERNMENT-SOFTWAREKOMPONENTE/>

Am 5. Feb

Die OSCI-Transport Bibliothek ist eine Softwarekomponente, welche von vielen deutschen Behörden eingesetzt wird, um Daten gemäß dem OSCI-Transport Protokoll sicher zu übertragen. Diese Java-Bibliothek war gegen zwei potentielle Angriffe anfällig, welche es einem Angreifer ermöglichten, einige Sicherheitsmaßnahmen zu umgehen.

Seit der Einführung des OSCI-Transport Protokolls 2001 wurde dieses von diversen deutschen Behörden eingeführt, um Daten sicher über unsichere Netzwerke < https://www.xoev.de/detail.php?gsid=bremen83.c.3355.de> (wie dem Internet) zu übertragen. Das OSCI-Transport Protokoll gewährleistet Integrität, Vertraulichkeit, Authentizität und Nicht-Abstreitbarkeit für sämtliche übertragenen Datei Die OSCI-Transport Bibliothek ist eine kostenfreie Implementierung dieses Protokolls und wird von der KoSIT < https://www.xoev.de/downloads-2316#Standards> bereitgestellt.

2017 < /vulnerability-lab/advisories/#a230> konnte das SEC Consult Vulnerability Lab bereits diverse Sicherheitsschwachstellen in dieser Bibliothek feststellen. Bei einer kurzen erneuten Überprüfung der Bibliothek im September 2018 wurden zwei weitere Schwachstellen identifiziert, welche in diesem Blogpost beschrieben werden. Diese Schwachstellen erlauben einem Angreifer die Sicherheitsmechanism auszuhebeln, welche dazu gedacht sind, die Metadaten einer Nachricht zu schützen (*Verschlüsselung von Aufträgen und Auftragsantworten* und *Signatur von Aufträgen und Auftragsantworten*).

Vorausgesetzt sämtliche von OSCI-Transport bereitgestellten Sicherheitsmechanismen werden verwendet, sind die Auswirkungen der Schwachstellen daher eingeschränkt. Wenn jedoch die Signatur von Inhaltsdaten und die Verschlüsselung von Inhaltsdaten nicht aktiv sind, sind die Auswirkungen der Schwachstelle kritisch. Da SEC Consult nicht bekannt ist, wie OSCI-Transport in der Praxis üblicherweise eingesetzt wird, können wir die tatsächlichen Auswirkungen der Schwachstellen nicht einschätzen. Unabhängig von der praktischen Relevanz halten wird die technischen Details dieser Schwachstellen für sehr interessant und werden daher in diesem Blogpost näher erörtert.

SEC Consult führte kein vollständiges Security Audit durch und kann daher keine Aussage über die gesamte Sicherheit der Bibliothek oder der implementierten Sicherheitsmechanismen treffen.

ÜBERSICHT OSCI-TRANSPORT

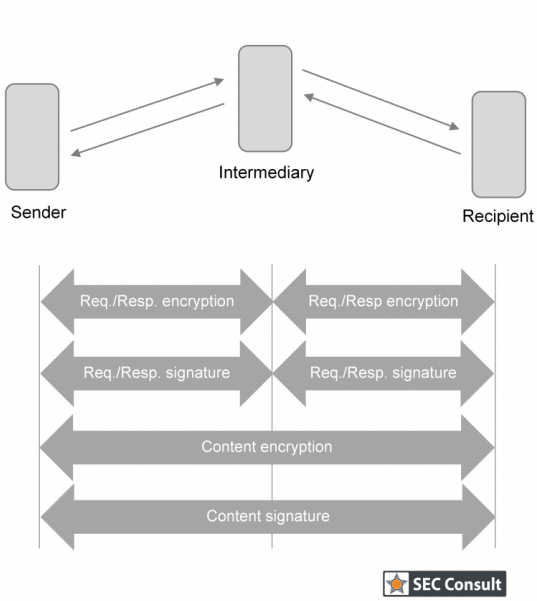
OSCI-Transport ist ein XML-basiertes Kommunikationsprotokoll, welches einen *Intermediär* benötigt, um die Kommunikation zwischen einem *Sender* und einem *Empfänger* zu ermöglichen. Die wichtigsten Sicherheitsfeatures von OSCI-Transport sind:

- Die *Signatur von Inhaltsdaten* gewährleistet Authentizität der übertragenen Daten.
- Die *Verschlüsselung von Inhaltsdaten* gewährleistet Vertraulichkeit der übertragenen Daten. Da der Intermediär die Inhaltsdaten nicht entschlüsseln kann, bietet dieser Mechanismus Ende-zu-Ende-Verschlüsselung.
- Die *Signatur von Aufträgen und Auftragsantworten* gewährleistet Authentizität für die Kommunikation zwischen einem *Sender* und dem *Intermediär* bzw. zwischen dem *Intermediär* un dem *Empfänger*.
- Die *Verschlüsselung von Aufträgen und Auftragsantworten* gewährleistet Vertraulichkeit für die Kommunikation vom und zum *Intermediär*.

Alle diese Sicherheitsmechanismen sind *optional* – die beteiligten Kommunikationspartner müssen sich vorab auf die zu verwendenden Mechanismen einigen.

Bei einem Datenaustausch baut ein Sender eine Verbindung zu einem *Intermediär* auf (z.B. über HTTP) und überträgt die OSCI-Transport XML-Nachricht. Um Daten empfangen zu können, sind zwei Szenari definiert: Ein *aktiver Empfänger* initiiert eine Verbindung mit dem Intermediär, um Nachrichten abzuholen, während ein *passiver Empfänger* auf Verbindungen eines Intermediärs wartet, um Nachrichten zu empfangen.

Das folgende Diagramm zeigt auf welchen Kommunikationsstrecken die Sicherheitsmechanismen von OSCI-Transport angewendet werden.



XML SIGNATURE WRAPPING

2017 < https://sec-consult.com/vulnerability-lab/advisories/#a230> hat SEC Consult festgestellt, dass die Prüfung der Signatur von Aufträgen und Auftragsantworten in der OSCI-Transport Bibliothek verwundbar für einen XML Signature Wrapping Angriff ist. Solch ein Angriff macht sich zu Nutze, dass ein verwundbarer Parser eine XML-Signatur auf einem Teil der Nachricht prüft, während ein anderer Te für die weitere Verarbeitung verwendet wird. Zum Beispiel enthält eine OSCI-Transport Nachricht das XML-Element *soap:Body* welches die eigentlichen Inhaltsdaten beinhaltet. Bei diesem Angriff würde ein Angreifer mehrere *body*-Elemente in ein Dokument einfügen und den Parser dazu bringen einen *body* für die Signaturprüfung zu verwenden, aber einen anderen *body* zum Auslesen der Inhaltsdaten heranzuziehen. Der Angreifer könnte einen gültig signierten *body* für die Signaturprüfung bereitstellen sowie einen manipulierten *body* mit gefälschten Inhaltsdaten, welcher von der Anwendung weiterverarbeitet wird.

Die damals identifizierte Schwachstelle wurde behoben. Jedoch konnte nun ein anderer Ansatz identifiziert werden, welcher diesen Angriff erneut zulässt:

```
<soap:Envelope>
 <soap:Header>
  ...
  <osci:ClientSignature>
    <ds:Signature>
     <ds:SignedInfo>
      <ds:Reference URI="#body">
       ...
       <ds:DigestValue>(digest value of _original_ body)</ds:DigestValue>
      </ds:Reference>
      ...
     </ds:SignedInfo>
     <ds:SignatureValue>...</ds:SignatureValue>

     <ds:Reference URI="#body">
      ...
      <ds:DigestValue>(digest value of _new_ body)</ds:DigestValue>
     </ds:Reference>

    </ds:Signature>
  </osci:ClientSignature>
  ...
 </osci:Header>
 <soap:Body Id="body">(modified content here)</soap:Body>
</soap:envelope>
```

Im Gegensatz zum 2017 demonstrierten Angriff, wird bei diesem Ansatz kein zweiter *body* sondern ein weiteres `Reference`-Element eingefügt.

`Reference`-Elemente werden verwendet, um XML-Elemente mit deren erwarteten Hash-Werten zu verknüpfen (z.B. zeigt das `Reference` Element, das sich auf den *body* bezieht, an, welcher Hashwert fü das *body*-Element und dessen Kind-Elemente zu erwarten ist). Diese `Reference`-Elemente tauchen normalerweise in einem `SignedInfo`-Element auf. Die eigentliche Signatur wird gegen dieses `SignedInfo`-Element bzw. dessen Kind-Elemente geprüft. Wenn der Inhalt eines referenzierten Elements verändert wird, würde der Parser dies erkennen, weil der im `Reference`-Element gespeicherte Hash nicht mehr zum berechneten Hash passt.

Offenbar akzeptiert die OSCI-Transport Bibliothek auch `Reference`-Elemente außerhalb von `SignedInfo`-Elementen. Werden mehrere `Reference`-Elemente angegeben, die sich auf das gleiche Eleme beziehen, wird nur das Letzte davon verarbeitet. Ein Angreifer kann ein zusätzliches `Reference`-Element nach dem `SignedInfo`-Element einfügen, welches einen manipulierten Hash anzeigt. Da das `SignedInfo`-Element nicht verändert wurde, wird die Signatur des Dokuments als gültig anerkannt.

CBC MODE SCHWACHSTELLEN

Eine weitere Schwachstelle ist auf die Verwendung des CBC-Modus zurückzuführen. Die W3C rät von der Verwendung des CBC-Modus < https://www.w3.org/TR/xmlenc-core1/#sec-Table-of-Algorithms> a und empfiehlt die Verwendung sichererer Modi. Aus diesem Grund hat der Herausgeber des OSCI-Standards (KoSIT) damit begonnen, den CBC-Modus abzulösen <

https://www.xoev.de/sixcms/media.php/13/KoSIT_Umstellung_GCM_20180625_finaleFassung.pdf> und durch den sichereren GCM-Modus zu ersetzen. Diese Übergangsphase sollte mit November 2019 abgeschlossen sein. Derzeit erlaubt die OSCI-Transport Bibliothek noch die Verwendung des CBC-Modus. Es ist SEC Consult nicht bekannt, wie viele Organisationen diesen Modus derzeit noch einsetzen.

2017 < https://sec-consult.com/vulnerability-lab/advisories/#a230> konnte das SEC Consult Vulnerability Lab einen Padding Oracle Angriff gegen die Implementierung des CBC-Modus für die Verschlüsselung von Aufträgen und Auftragsantworten demonstrieren. Diese Schwachstelle wurde zwar behoben – wir konnten jedoch den ursprünglichen Angriff (aufgrund der bekannten Schwachstellen CBC-Modus) so verändern, dass dieser wieder funktioniert.

Um die Schwachpunkte im CBC-Modus ausnutzen zu können, muss ein Angreifer eine Möglichkeit finden, die Serverantwort vom entschlüsselten Klartext einer Nachricht abhängig zu machen. Ein Padding Oracle Angriff funktioniert beispielsweise dann, wenn der Server preisgibt, ob eine entschlüsselte Nachricht ein gültiges Padding aufweist.

Der neue Angriff nutzt die Tatsache, dass die verschlüsselte Nachricht aus einem MIME Objekt besteht. Im folgenden Abschnitt ist ein Beispiel einer MIME-Nachricht zu sehen:

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary_9MnMkzyjeaR9bguP9KZvpdeoY1GEChQI; type=text/xml

--MIME_boundary_9MnMkzyjeaR9bguP9KZvpdeoY1GEChQI
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: <osci@message>
Content-Length: 123

...
--MIME_boundary_9MnMkzyjeaR9bguP9KZvpdeoY1GEChQI--
```

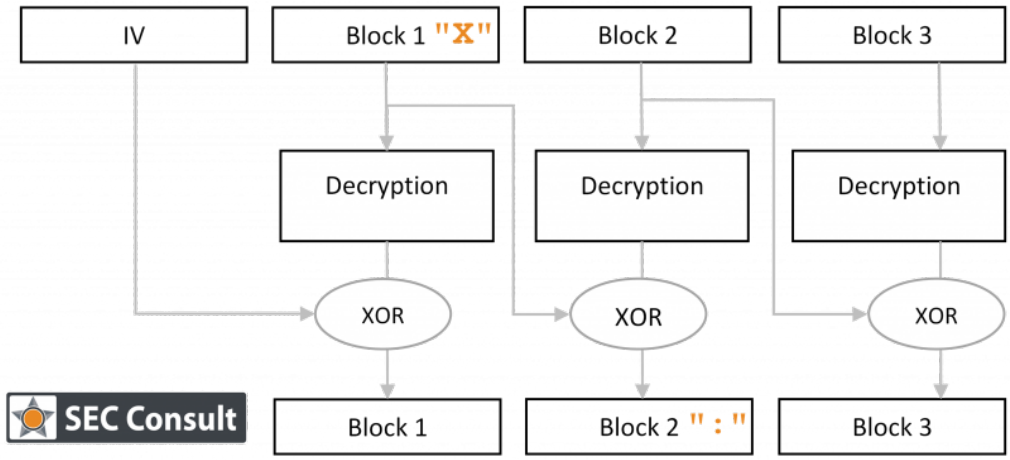
Um diesen Angriff durchführen zu können, müssen wir einen Teil der MIME-Nachricht finden, welcher – abhängig vom Inhalt – ein unterschiedliches Verhalten am Server auslöst. Dieser Teil ist der MIME-Header, genauer gesagt das Doppelpunkt-Zeichen im MIME-Header. Folgendes Beispiel demonstriert den Ansatz:

```
SomeHeader123!"%:test
SomeHeader123!"%_test
```

Die erste Zeile würde als gültiger MIME-Header interpretiert werden, während die zweite Zeile, aufgrund des fehlenden Doppelpunkts, eine Parsing-Fehlermeldung auslösen würde – das ist perfekt für einen Angriff.

CBC-Modus Hintergrund

Das folgende Diagramm zeigt ein Beispiel einer CBC-Entschlüsselungsoperation:



In diesem Beispiel wissen wir, dass das letzte Byte des zweiten entschlüsselten Blocks ein Doppelpunkt ist. Wir wissen auch, dass das letzte Zeichen des ersten verschlüsselten Blocks ein „X“ ist. Aufgrund der Struktur der CBC-Entschlüsselung können wir das letzte Byte der Ausgabe der Entschlüsselungsfunktion berechnen. Wir wissen:

Verschlüsselter Block 1 **XOR** *Entschlüsselungsfunktion Ausgabe Block 2* **=** *Entschlüsselter Block 2*

und daher:

Verschlüsselter Block 1 (letztes Byte) **XOR** *Entschlüsselter Block 2 (letztes Byte)* **=** *Entschlüsselungsfunktion Ausgabe Block 2 (letztes Byte)*

Im konkreten Fall können wir berechnen, dass „`X`“ XOR „`:`“ = „`b`“ – das ist das letzte Byte der Ausgabe der Entschlüsselungsfunktion für Block 2.

Wenn es für den Angreifer möglich ist herauszufinden, dass ein bestimmtes Zeichen im entschlüsselten Text ein Doppelpunkt ist, ist es möglich das korrespondierende Byte der Ausgabe der Entschlüsselungsfunktion zu bestimmen. Wenn wir es auf diesem Weg schaffen alle Ausgaben der Entschlüsselungsfunktion herauszufinden, können wir sehr einfach die originale Nachricht bestimmen.

Die Suche nach dem Doppelpunkt

Für den Angriff ersetzen wir den Block einer originalen Nachricht, welcher den Doppelpunkt eines MIME-Header enthält, mit mehreren manipulierten Blöcken. Da der Aufbau der Header in einer OSCI-Nachri sehr vorhersagbar ist, können wir sehr zuverlässig einschätzen, welcher Block ersetzt werden muss. Werden die manipulierten Nachrichtenfragmente vom Server entschlüsselt, sind drei Fälle denkbar („X“ repräsentiert zufällige Bytes außer `":"`, `"\r"` und `"\n"`):

```
Content-TypeXXXXXXXXXXXXXXXXtext/xml
Content-TypeXXXXXXX:XXXXXXXXtext/xml
Content-TypeXXXXXXX\r\nXXXXXXXtext/xml
```

Im ersten Fall würde ein Parsing Fehler auftreten, weil kein Doppelpunkt im Header auftritt. Im zweiten Fall würde der Header korrekt geparsed werden. Da es sich um einen unbekannten Header handelt, würde dieser ignoriert werden und es würde kein Fehler zurückgeliefert. Im dritten Fall würde ein Parsing Fehler auftreten. Der dritte Fall ist relativ unwahrscheinlich, weil zwei aufeinanderfolgende Bytes zufällig ganz bestimmte Werte annehmen müssten. Wir können den dritten Fall daher vorerst ignorieren.

Praktisch bedeutet dies, dass wir erkennen können, ob die Entschlüsselung der eingefügten Blöcke einen Doppelpunkt enthält. Wir erkennen jedoch nicht, an welcher Position sich der Doppelpunkt befindet

Der Exploit

Wir wissen nun, wie ein Angreifer sich die Serverantworten zunutze machen kann, um die Ausgaben der Entschlüsselungsfunktion zu errechnen. Wir haben eine Möglichkeit aufgezeigt, um zu erkennen ob d Entschlüsselung manipulierter verschlüsselter Blöcke einen Doppelpunkt enthält. Aus beiden Punkten kann ein Exploit entwickelt werden:

Der Exploit fügt 3 Blöcke in den MIME Header ein. Diese drei Blöcke sind:

1. Der *Testblock*: Dieser Block wird mit zufälligen Daten initialisiert. In jeder Iteration wird ein Byte ausgewählt, welches gezielt verändert wird.
2. Der *Zielblock*: Der Teil der originalen Nachricht, der entschlüsselt werden soll.
3. Der *Suffixblock*: Dieser Block enthält zufällige Daten.

Der Exploit führt folgende Schritte durch, um ein einzelnes Byte zu entschlüsseln:

1. Im Testblock wird das Byte, das entschlüsselt werden soll, auf einen bekannten Wert gesetzt (z.B: das erste Byte wird auf 0 gesetzt)
2. Die Blöcke werden an den Server zur Entschlüsselung gesendet.
3. Aufgrund der XOR-Operation des CBC-Modus beeinflusst das Byte im Testblock das korrespondierende Byte im entschlüsselten Text des Zielblocks. Aus der Serverantwort erkennen wir:
    1. Wenn der Server anzeigt, dass kein Doppelpunkt im entschlüsselten Text enthalten ist, produziert das aktuell getestete Byte keinen Doppelpunkt in der Ausgabe. Wir können daher einen Kandidaten für das originale Plaintext-Zeichen ausschließen.
    2. Wenn der Server anzeigt, dass ein Doppelpunkt gefunden wurde, lernen wird daraus nichts – der Doppelpunkt könnte zufällig in irgendeinem entschlüsselten Block aufgetreten sein (z.B. auch in der Entschlüsselung des Testblocks oder des Suffixblocks)
4. Der Byte-Wert, welcher im ersten Schritt gewählt wurde, wird auf einen anderen Wert geändert. Die Ausführung wird bei Schritt 2 fortgeführt.

Nachdem alle möglichen Byte-Werte getestet wurden, wurden damit die möglichen Klartextwerte für das gesuchte Byte eingeschränkt. Danach wird der Testblock und der Suffixblock mit zufälligen Daten gefüllt und der Prozess wird erneut durchgeführt. Dies geschieht so lange, bis nur mehr eine Möglichkeit für den Klartextwert existiert.

Im letzten Schritt werden mehrere Nachrichten gesendet, die nach den bekannten Informationen einen Doppelpunkt erzeugen sollten. Tritt dabei ein Fehler auf (d.h. das Nachrichtenfragment enthält keinen Doppelpunkt) wird der Prozess neu gestartet. Dies dient dazu, False Positives auszuschließen (z.B. wenn zuvor bei einem Schritt ein \r\n auftrat).

Dieser Prozess wird für alle Bytes der Originalnachricht wiederholt. Ein Angreifer kann dadurch die Verschlüsselung von Aufträgen und Auftragsantworten des OSCI-Transport Protokolls aushebeln.

Im Folgenden ist die Ausführung des Exploits zu sehen. Man erkennt, dass die Menge an möglichen Klartextzeichen mit jeder Iteration eingeschränkt wird:

DER PATCH

Das SEC Consult Vulnerability Lab meldete diese Schwachstellen im September 2018 an CERT-Bund, welches bereits in der Vergangenheit sehr hilfreich bei der Koordination von Schwachstellen war. KoSIT und der Softwarehersteller Governikus analysierten die Schwachstellen und stellten den betroffenen Parteien einen Patch für die Java- und .Net-Version der Software bereit. Um den betroffenen Parteien genügend Zeit für die Anwendung des Patches zu geben, verschob SEC Consult die Veröffentlichung des Security Advisories auf 5. Februar 2019.

Es wird dringend empfohlen auf die Version 1.8.3 (Java oder .Net) der OSCI-Transport Bibliothek zu wechseln und sobald wie möglich auf den GCM-Modus umzusteigen. Wir empfehlen Organisationen, die noch weiter auf CBC angewiesen sind, TLS als zusätzlichen Security-Layer einzusetzen.

Dieser Research wurde von Wolfgang Ettlinger (@ettisan < https://twitter.com/ettisan>) im Rahmen des SEC Consult Vulnerability Lab erarbeitet und auch als Security Advisory (Englisch) veröffentlicht. < /en/blog/advisories/multiple-vulnerabilities-in-osci-transport-library-1-2-for-german-egovernment/>

SEC Consult ist einer der führenden Berater im Bereich Cyber- und Applikationssicherheit. Das Unternehmen ist Spezialist für die Einführung von Informationssicherheits-Management, Sicherheitsaudits, Penetrationstests und DDoS Benchmarktests, Zertifizierungsbegleitung für ISO 27001, Cyber Defence und sichere Software.