Diplomarbeit

Entwurf einer Sprache zur Klassifizierung von Malware-Vorfällen

Michel Messerschmidt 2004-01-15

Übersicht

- Ausgangssituation
  - Reportformat des aVTC
  - Reportformate von Antivirus-Produkten
- Konzept
- Grundlagen
  - XML
  - XML Schema
- Umsetzung

Ausgangssituation

- Für die Auswertung der aVTC-Tests sind viele von Antivirus-Produkten verwendete Reportformate unzureichend
- Auch das in der aVTC Auswertung bisher verwendete Format verursacht zusätzlichen Aufwand
- Es existiert kein Standardformat für Berichte über Malware

Ausgangssituation
aVTC-Format

- ASCII-Text
- Jede Zeile beschreibt eine Datei mittels Dateiname, Pfad und Malware-Identifikation
- In diesen Angaben sind keine Kleinbuchstaben und keine Leerzeichen erlaubt
- DOS-Konventionen: Dateinamen im 8+3 Format
- Beispiel:

```
I:\FILE\23W\MRONON\B\MSUB\5441\EXA_012_.EXE W95/SPACES.1445.B
I:\FILE\23W\MRONON\B\TSAEB\27414\A\EXA_000_.EXE W32/BEAST
I:\FILE\23W\MRONON\C\HIC\3001\A\EXA_000_.EXE W95/CIH.A
N:\FILE\23W\MRONON\B\MSUB\5441\ARJ.ARJ\EXA_012_.EXE WIN95.SPACES.1445
N:\FILE\23W\MRONON\B\MSUB\5441\ZIP.ZIP\EXA_012_.EXE WIN95.SPACES.1445
```

Ausgangssituation
Formate von Antiviren-Produkten

- sehr unterschiedlich
- meist textbasiert, aber auch dBase, CSV, RTF und HTML-ähnliche Formate
- Probleme:
  - Es können nicht beliebige Datei- bzw. Virusnamen dargestellt werden

```
I:\W97M\MELISSA\A@MM\W97_000_.DOC Virus found W97M/Melissa

/usr/local/public/testdatei ohne virus.exe ... is OK.

/usr/local/public/testdatei ohne virus.exe

Objekt: "html" in Pfad "W:\TIKNOCV\MRONON\V\SBV\TPIRCSNU.R\2N↵
EG\EXA_000_.EXE/[From:  "KHJ" <khj@yahoo.com>][Date:  Fri, 9 ↵
Feb 2001 01:45:00 +0700]". Status: "Virus (erkannt); archivie↵
rt in Datei "W:\TIKNOCV\MRONON\V\SBV\TPIRCSNU.R\2NEG\EXA_000_↵
.EXE"".
```

Ausgangssituation
Formate von Antiviren-Produkten (2)

- weitere Probleme:
  - Datei-Inhalte werden nicht eindeutig gemeldet

```
N:\FILE\23W\MRONON\C\HIC\3001\A\ZIP.ZIP\EXA_011_.EXE

N:\FILE\23W\MRONON\C\HIC\3001\A\ZIP.ZIP->FILE/23W/MRONON/C/HI↵
C/3001/A/EXA_001_.EXE

P:\ZIP.ZIP\EXA_000_.EXE ... Found the W32/Hybris.gen@MM virus
P:\ZIP.ZIP\EXA_001_.EXE ... Found the W32/Hybris.gen@MM virus
P:\ZIP.ZIP\EXA_002_.EXE ... Found the W32/Hybris.gen@MM virus
P:\ZIP.ZIP\EXA_000_.EXE ... Found the W32/MsInit.worm.a virus
```

  - Beschränkungen der maximalen Pfadlänge

```
R:\FILE\23W\MRONON\C\HIC\3001\A\CAB.CAB>>...>>...>>...>>...>>↵
...>>...>>...>>...>>EXA_000_.EXE

R:\MALW\MROW\MRONON\W\23W\TINISM\MROW\B\ZIP.ZIP>>...>>...>>..↵
.>>...>>...>>...>>...>>...>>MALW/MRO...
```

Ausgangssituation
Formate von Antiviren-Produkten (3)

- weitere Probleme:
  - Archiv-Inhalte werden als infiziert gemeldet, das Archiv selber aber als „OK“
  - Verwirrende Statusmeldungen

```
H:\W97M\CLASS\FA\CLASS-FA.DOT  Infection: W97M/Not_a_virus (exact)

L:\IRC\M\MIRCNEW\A\IRC_000_.INI hat unübliches Datum: 11.05.2080

O:\TROJAN\VBS\SEEKER\H\VBS_000_.HTM - ARCHIVE HTML
O:\TROJAN\VBS\SEEKER\H\VBS_000_.HTM\JSCRIPT.ENCODE-0 PACKED BY↵
 ENCODED SCRIPT
O:\TROJAN\VBS\SEEKER\H\VBS_000_.HTM\JSCRIPT.ENCODE-0 INFECTED WITH↵
 TROJAN.SEEKER

210518123017,5,1,1,METEOR,Administrator,W95.Spaces.1445,N:\FILE\23W↵
\MRONON\B\MSUB\5441\JAVA.JAR>>I:/FILE/23W/MRONON/B/MSUB/5441/EXA_01↵
2_.EXE,4,4,4,2147483904,33571876,"",1056473303,,0,,0,29234,0,0,0,1,↵
1,13,20020619.005,17467,0,4,0,,{D035D199-7948-4BA7-8A70-AF7A2FAF8D7↵
},,,,VTC,,8.0.9374
```

Konzept

- Anforderungen des aVTC
  - Datei-Identifikation und -Klassifikation müssen eindeutig unterscheidbar sein
  - Dateinamen und Pfade müssen unverändert wiedergegeben werden
  - Nicht-ASCII-Zeichen (Umlaute, Akzente) müssen verarbeitet werden können
  - Berichte über Dateien und Datei-Inhalte sollten nicht vermischt werden
  - Wünschenswert sind standardisierte Klassifikations-Kategorien

Konzept
Andere Anwendungsmöglichkeiten

- Einheitliches Reportformat auch für verwandte Produkte:
  - „Intrusion Detection System“
  - „Vulnerability scanner“
  - „Integrity checker“
  - „Firewall“
- Allgemeinere Klassifikation von Software
  - bisher nur Klassifikation als „maliziös“ (bzw. „Malware“) oder „nicht maliziös“
  - alternativer Ansatz mit drei Kategorien:
    - „malicious“ bzw. „Malware“
    - „unknown“
    - „verified“ bzw. „Goodware“

Konzept
Weitere Anforderungen

- Verarbeitung von anderen Datenobjekten außer Dateien muß möglich sein
  - z. B. auch Bootsektoren, Netzwerkpakete, Speicherbereiche
  - es muß eine eindeutige Identifikation für jedes Datenobjekt möglich sein
- keine Einschränkung durch Textkodierungen
  - jedes beschreibbare Datenobjekt muß auch darstellbar sein
- es soll nicht möglich sein, Malware mit diesem Format zu transportieren
  - also darf keine vollständige Beschreibung aller Eigenschaften eines Datenobjekts möglich sein
- einfach zu verstehen
- portabel
- ohne spezielle Anwendungen verarbeitbar

Konzept
Realisierung

- XML als Basistechnologie
  - basiert auf Unicode
  - kann mit vielen Textkodierungen verwendet werden
  - offener, frei verfügbarer Standard
  - einfache Transformation in andere Formate
  - gute Unterstützung der Datenstrukturierung
- mögliche Alternativen:
  - SGML
    - ist deutlich komplexer
    - kein frei verfügbarer Standard
  - ASCII
  - eine komplette Eigenentwicklung verspricht ebenfalls keine Vorteile

Grundlagen
XML

- XML („Extensible Markup Language“) ist seit 1999 ein W3C Standard
- stark vereinfachte Weiterentwicklung von SGML („Standard Generalized Markup Language“, ISO-8879:1986)
- Metasprache zur Definition anderer Auszeichnungs-sprachen (bekannteste Anwendung: XHTML)
  - definiert eine feste Syntax aber (fast) keine logischen Strukturen

```
<elementname attributname="attributwert">
    Elementinhalt
</elementname>
```

- Dem XML-Standard entsprechende Dokumente werden „wohlgeformt“ genannt
- Etliche Erweiterungen des XML-Standards (u.a. XSL, XML Namespaces, XPath, XLink, XPointer)

Grundlagen
XML (2)

- konkrete Auszeichungssprachen wurden ursprünglich nur per DTD („Document Type Definition“) definiert
- einer DTD entsprechende Dokumente werden „gültig“ genannt
- Nachteile von DTDs:
  - andere Syntax (DTDs sind nicht „wohlgeformt“)
  - kaum Kontrolle über Element-Inhalte, insbesondere sind fast keine Datentyp Angaben möglich
  - keine Unterstützung der „XML Namespaces“
- seit einigen Jahren werden Alternativen entwickelt, die zum Teil bereits standardisiert sind:
  - XML Schema, Relax NG, Schematron, XDR, ...

Grundlagen
XML Schema

- W3C Standard seit 2001
- Vorteile:
  - unterstützt XML Namespace & XInclude
  - viele vordefinierte Datentypen
  - Möglichkeit zur Definition eigener Datentypen
  - mehrere unterschiedliche Inhaltsmodelle
  - objektorientierte Ansätze (Vererbung)
  - gute Unterstützung der „Inline“-Dokumentation
- Nachteile:
  - sehr umfangreiche und komplexe Spezifikation
  - zum Teil unnötig kompliziert (Beschränkung auf deterministische Inhaltsmodelle)

Grundlagen
XML Schema (2)

- Deklaration von Elementen und Attributen durch Angabe von Namen und Datentyp

```
<xsd:element name="alter" type="xsd:nonNegativeInteger"/>
```

```
<alter>42</alter>
```

```
<xsd:attribute name="id" type="xsd:ID"/>
```

```
<alter id="THX1138">42</alter>
```

- Deklaration von Element- und Attribut-Gruppen
- Definition von Datentypen
  - einfache Typen enthalten nur Text (ohne Subelemente)

```
<xsd:simpleType name="checksumType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="MD5"/>
        <xsd:enumeration value="SHA1"/>
    </xsd:restriction>
</xsd:simpleType>
```

  - komplexe Typen:
    - „simple content“ einfache Typen mit Attributen
    - „complex content“ enthält nur Subelemente
    - „mixed content“ enthält Subelemente und Text
    - „empty content“ Elemente ohne Inhalt (Attribute erlaubt)

Grundlagen
XML Schema (3)

- Beispiel für „simple content“

```
<xsd:complexType name="alterType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:nonNegativeInteger">
            <xsd:attribute name="id" type="xsd:ID"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
```

- Beispiele für „complex content“

```
<xsd:complexType name="identificationType">
    <xsd:choice>
        <xsd:element name="personalausweisNr" type="xsd:token"/>
        <xsd:element name="reisepassNr" type="xsd:token"/>
        <xsd:element name="secretkeyNr" type="xsd:token"/>
        <xsd:element name="fingerprintNr" type="xsd:token"/>
    </xsd:choice>
</xsd:complexType>
```

```
<xsd:complexType name="malwareClassificationType">
    <xsd:sequence>
        <xsd:element name="malwaretype" type="malwareDatatype"/>
        <xsd:element name="malwarename" type="xsd:token"/>
        <xsd:element name="reliability" type="reliableType"/>
    </xsd:sequence>
</xsd:complexType>
```

Umsetzung
Report

Beispieldokument

```
<?xml version="1.0" encoding="UTF-8"?>
<report version="1.0" xmlns="http://www.michel-messerschmidt.de/scl/v1">
    <source>
        <date>2004-12-01T23:09:00+01:00</date>
        <system>192.168.42.8 (Linux 2.4)</system>
        <program>handmade</program>
        <version>v1.12</version>
    </source>
    <object>...</object>
    <object>...</object>
</report>
```

Schema

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="http://www.michel-messerschmidt.de/scl/v1">

<xsd:element name="report">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="source" type="sourceType" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="object" type="objectType" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="version" type="xsd:token" default="1.0"/>
  </xsd:complexType>
</xsd:element>
```

Umsetzung
Identifikation (1)

Beispieldokument

```
<object>
  <identification>
    <file>...</file>
  </identification>
  <classification>
    ...
  </classification>
</object>
```

```
<object>
  <identification>
    <file>...</file>
  </identification>
  <content>
    <header>...</header>
    <object>...</object>
    <object>...</object>
  </content>
</object>
```

Schema

```
<xsd:complexType name="objectType">
  <xsd:sequence>
    <xsd:element name="identification">
      <xsd:complexType>
        <xsd:choice>
          <xsd:element name="file" type="fileIdentType"/>
          <xsd:element name="sector" type="sectorIdentType"/>
          <xsd:element name="sectorgroup" type="sectorgroupType"/>
          <xsd:element name="memory"  type="memoryIdentType"/>
          <xsd:element name="packet" type="packetIdentType"/>
          <xsd:element name="octetstream" type="octetstreamIdentType"/>
        </xsd:choice></xsd:complexType></xsd:element>
    <xsd:choice>
      <xsd:element name="content" type="contentType"/>
      <xsd:element name="classification" type="classificationType"/>
    </xsd:choice></xsd:sequence></xsd:complexType>
```

Umsetzung
Identifikation (2)

Beispieldokument

```
<file>
  <name>sample.exe</name>
  <path>/u/file/w32/Klez/H/</path>
  <mimetype>
    application/x-msdos-program
  </mimetype>
  <checksum type="MD5">
    f6a69fe04b358f65ee5e126473169801
  </checksum>
</file>
```

```
<file>
  <uri>
    file:///J:\XM\LAROUX\A\RAR3.RAR
  </uri>
  <mimetype>
    application/x-rar-compressed
  </mimetype>
  <checksum type="MD5">
    a24ff17709f1c162ccad34d5646a493a
  </checksum>
</file>
```

Schema

```
<xsd:complexType name="fileIdentType">
  <xsd:sequence>
    <xsd:choice>
      <xsd:sequence>
        <xsd:element name="name" type="xsd:string"/>
        <xsd:element name="path" type="xsd:string" minOccurs="0"/>
      </xsd:sequence>
      <xsd:element name="uri" type="xsd:anyURI"/>
    </xsd:choice>
    <xsd:element name="mimetype" type="mimeDatatype" minOccurs="0"/>
    <xsd:element name="checksum" type="checksumType"
                 minOccurs="0"  maxOccurs="unbounded"/>
    <xsd:element name="creationtime" type="xsd:dateTime" minOccurs="0"/>
    <xsd:element name="modificationtime" type="xsd:dateTime" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

Umsetzung
Klassifikation (1)

Beispieldokument

```
<classification>
  <malicious>
    <malwaretype>
      virus
    </malwaretype>
    <malwarename>
      W32/Klez.H@MM
    </malwarename>
    <reliability>
      exact identification
    </reliability>
  </malicious>
</classification>
```

```
<classification>
  <unknown>
    <property>
      not analysed
    </property>
    <reason>
      compression format not supported
    </reason>
  </unknown>
</classification>
```

Schema

```
<xsd:complexType name="classificationType">
  <xsd:choice>
    <xsd:element name="malicious" type="malClassType" maxOccurs="unbounded"/>
    <xsd:element name="verified" type="goodClassType"/>
    <xsd:element name="unknown" type="unknownType"/>
  </xsd:choice>
</xsd:complexType>
```

Umsetzung
Klassifikation (2)

Beispieldokument

```
<classification>
  <malicious>
    <malwaretype>
      virus
    </malwaretype>
    <malwarename>
      W32/Klez.H@MM
    </malwarename>
    <reliability>
      exact identification
    </reliability>
  </malicious>
</classification>
```

```
<classification>
  <malicious>
    <malwaretype>
      trojan
    </malwaretype>
    <malwarename>
      IRC/Flood.gen
    </malwarename>
    <reliability>
      generic identification
    </reliability>
  </malicious>
</classification>
```

Schema

```
<xsd:complexType name="malClassType">
  <xsd:sequence>
    <xsd:element name="malwaretype" type="mwDataType"/>
    <xsd:element name="malwarename" type="xsd:token"/>
    <xsd:element name="reliability" type="reliableType"/>
    <xsd:element name="alertlevel" type="xsd:unsignedByte" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

Umsetzung
Klassifikation (3)

Beispieldokument

```
<classification>
  <malicious>
    <malwaretype>
      virus
    </malwaretype>
    <malwarename>
      W32/Klez.H@MM
    </malwarename>
    <reliability>
      exact identification
    </reliability>
  </malicious>
</classification>
```

```
<classification>
  <unknown>
    <property>
      not infected
    </property>
    <property>
      may be a corrupted file
    </property>
    <property>
      contains malicious code
    </property>
  </unknown>
</classification>
```

Schema

```
<xsd:simpleType name="mwDataType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="virus"/>
    <xsd:enumeration value="worm"/>
    <xsd:enumeration value="backdoor"/>
    <xsd:enumeration value="trojan"/>
    ...
    <xsd:enumeration value="unknown"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:complexType name="unknownType">
  <xsd:sequence>
    <xsd:element name="property"
                 type="xsd:token"
                 minOccurs="0"
                 maxOccurs="unbounded"/>
    <xsd:element name="reason"
                 type="xsd:token"
                 minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

Umsetzung
Klassifikation (4)

Beispieldokument

```
<classification>
  <verified>
    <type>system core application</type>
    <name>Command interpreter</name>
    <method>signature</method>
    <trustbase>Microsoft root certificate</trustbase>
  </verified>
</classification>
```

Schema

```
<xsd:complexType name="goodClassType">
  <xsd:sequence>
    <xsd:element name="type" type="xsd:token"/>
    <xsd:element name="name" type="xsd:token"/>
    <xsd:element name="method" type="xsd:token"/>
    <xsd:element name="trustbase" type="xsd:token"/>
  </xsd:sequence>
</xsd:complexType>
```

Ausblick

- Anforderungen des aVTC erfüllt
  - einheitliche Kategorien aber eher utopisch
- Allgemeine Klassifikation ist beschreibbar
  - mangels Anwendungen schwer für einen praktischen Einsatz zu optimieren
- Andere Datenobjekte sind darstellbar
  - eindeutige Identifikation schwierig
  - weitere „high level“ Objekte notwendig (rfc822), z. B. Mail nach RFC822 / MIME
- Klassifikationen für verwandte Produkte fehlen noch