H. Langweg, M. Meier, B.C. Witt, D. Reinhardt (Hrsg.): Sicherheit 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 253

Fallstricke bei der Inhaltsanalyse von Mails: Beispiele, Ursachen und Lösungsmöglichkeiten

Steffen Ullrich[1]

Abstract: E-Mail ist eine der Hauptangriffswege zur Infektion mit Malware und zum Phishing von Zugangsdaten. Waren Mails vor 1996 auf ASCII-Zeichen und eine Zeilenlänge von 1000 Zeichen beschränkt, so ermöglicht die Nutzung der MIME-Standards heute die Abbildung beliebiger Zeichenkodierungen und binärer Anhänge innerhalb der ursprünglichen Beschränkungen. Die durch die Komplexität und Flexibilität dieser Standards bedingten Implementationsdifferenzen ermöglichen jedoch die Konstruktion von Mails, welche unterschiedlich in Sicherheits- und Endsystemen interpretiert werden. Wir haben exemplarisch untersucht, wie dadurch die Analyse in existenten Sicherheitsprodukten umgangen werden kann und welche Möglichkeiten es gibt, dieses Problem in der Praxis zu addressieren.

Keywords: Evasion, Semantic-Gap, Mail, MIME, Phishing, Malware, Firewall, IDS

1 Einführung

Die Abbildung der in heutigen E-Mails genutzten Features, wie beliebiger Zeichen und binärer Anhänge, auf die historischen Restriktionen von einer maximalen Zeilenlänge von 1000 Zeichen und nur ASCII, geschieht durch die Nutzung der MIME-Standards, welche 1996 und 1997 definiert wurden. Das folgende Beispiel zeigt eine typische Mail, in der die wichtigsten dieser Standards verwendet werden.

Deutlich erkennbar sind dabei die Definition von mehreren Mailteilen wie Text und Attachment mit jeweils eigenen Meta-Daten entsprechend RFC 2045 [FB96a] (Zeilen 7 bis 10 und 12 bis 15) und die Separation dieser Teile über einen textbasierten Trenner nach RFC 2046 [FB96b] (Zeilen 4, 6, 11 und 16). Da die Inhalte der Teile binäre bzw. nicht-ASCII-Zeichen haben, werden sie über die in RFC 2045 definierten Kodierungen Base64 (Zeile 15) bzw. Quoted-Printable (Zeile 10) zu ASCII-Text transformiert und dieses in den Meta-Daten entsprechend deklariert (Zeilen 8 und 13). Zusätzlich werden die Umlaute in den Meta-Daten der Mail bzw. Mailbestandteile kodiert, und zwar nach RFC 2047 [Mo96] im Subject der Mail (Zeile 3) sowie nach RFC 2231 [FM97] für den Dateinamen des Anhangs (Zeile 12):

[1] genua GmbH, 85551 Kirchheim bei München, Domagstr. 7, Deutschland, Steffen_Ullrich@genua.de

doi:10.18420/sicherheit2018_20

```
1  From: me@example.com
2  To: you@example.com
3  Subject: Viele =?UTF-8?Q?Gr=C3=BC=C3=9Fe?=
4  Content-type: multipart/mixed; boundary=trenner
5
6  --trenner
7  Content-type: text/plain; charset=UTF-8
8  Content-Transfer-Encoding: quoted-printable
9
10 Viele Gr=C3=BC=C3=9Fe von mir.
11 --trenner
12 Content-type: application/zip; name*=utf-8''%c3%bcbel.zip
13 Content-Transfer-Encoding: base64
14
15 UEsDBAoAAAAAAO6Tn0m2hH8nEwAAABMAAAAIA ...
16 --trenner--
```

Schon aus diesem Beispiel ist erkennbar, dass die Standards eine teilweise unnötige Flexibilität und damit einhergehende Komplexität aufweisen. Dazu kommt eine Mangel an klarer Definition in Grenzfällen. In der Praxis setzen daher viele Implementierungen nur einen Teil der Standards um und dieses oft leicht unterschiedlich. Zusätzlich werden nicht-standardisierte Erweiterungen eingesetzt. Durch gezieltes Ausnutzen dieser Implementationsdifferenzen kann ein Angreifer eine bösartige Mail so kodieren, dass sie in der gewünschten Weise vom Mail-Programm des Empfängers interpretiert wird, jedoch Mail-Filter, Antivirus-Produkte, Firewalls oder Intrusion-Detection-Systeme den Angriff nicht entdecken. Dieses Problem ist besonders brisant, weil Mail derzeit der primäre Verbreitungsweg für Malware und Phishing von Zugangsdaten ist. Wir konnten auch bereits Mails in der Praxis beobachten, die versuchen, derartige Interpretationsdifferenzen auszunutzen.

Im Folgenden zeigen wir exemplarisch einige Stellen, wo unterschiedliche Interpretationen typischerweise existieren. Wir haben untersucht, wie ein Analysesystem zur erfolgreichen Bekämpfung dieser Gefahren beschaffen sein muss und wie aktuelle Technologien und Produkte diese Anforderungen erfüllen.

2 Umgehung durch ambivalente und widersprüchliche Aussagen

Innerhalb der Meta-Daten wird zum einen die Kodierung des jeweiligen Mailteils deklariert (`Content-Transfer-Encoding`), wie auch die Art des Teils (`Content-Type`) sowie der textuelle Trenner bei Multipart-Mails (`boundary`). Auch wenn offensichtlich nur jeweils höchstens eine Deklaration Sinn macht, ermöglicht der Syntax der Meta-Daten mehrfache und widersprüchliche Deklarationen. Da die MIME-Standards keinen klaren Umgang mit solchen Fällen definieren, werden diese von Implementationen unterschiedlich behandelt. Als Beispiel zeigen wir das Verhalten bei einer widersprüchlichen Deklaration des Trenners in Multipart-Mails:

```
1  Content-type: multipart/mixed; boundary=foo
2  Content-type: multipart/mixed; boundary=bar
3
4  --foo
5  Content-type: text/plain
6
7  --bar
8  Content-type: application/octet-stream; name=malware.doc
7  Content-Transfer-Encoding: base64
8  ...
```

Hier benutzen die Mail-Clients Outlook und mutt die letzte Deklaration mit dem Trenner "bar". Sie ignorieren daher alles bis zum ersten Vorkommen von `--bar` in Zeile 7 als MIME-Preamble und sehen somit den Anhang mit dem Malware-Dokument. Apple-Mail, Thunderbird und Roundcube benutzen hingegen die erste Deklaration mit dem Trenner "foo" und interpretieren daher den Inhalt ab Zeile 7 als einfachen Text. Die gleiche Interpretation haben auch der Mail-Filter Amavisd, das Intrusion-Detection-System Snort und das Antivirus-Produkt ClamAV, d.h. diese interpretieren die Inhalte anders als das weit verbreitete Outlook. Ähnliche Effekte lassen sich mit einer Mehrfachdeklaration des `Content-Transfer-Encoding` erreichen. In dieser Situation benutzt Outlook nicht die letzte sondern die erste Deklaration und verhält sich damit anders als zum Beispiel das IDS Snort.

Auch die Kodierung von nicht-ASCII-Daten mittels Base64 oder Quoted-Printable selber kann verschieden interpretiert werden, zum Beispiel hinsichtlich Reaktion auf nicht erlaubte oder nicht erwartete Zeichen. So ended bei Base64 die Kodierung laut Standard eigentlich sobald ein Gleichheitszeichen auftritt. Demzufolge sollte im folgenden Beispiel Zeile 3 als Base64 interpretiert und Zeile 4 ignoriert werden. Die korrekte Dekodierung wäre damit "MALW" und wird so auch von Outlook oder mutt durchgeführt. Thunderbird und Roundcube hingegen machen mit der Dekodierung in Zeile 4 weiter und erhalten als Resultat entsprechend "MALWARE". Dieses Verhalten widerspricht den meisten Analysesystemen und ermöglicht so eine Umgehung der Analyse zum Beispiel bei Amavisd, Snort, ClamAV oder einer bekannten kommerziellen Firewall:

```
1  Content-Transfer-Encoding: base64
2
3  TUFMVw==
4  QVJF
```

Die bisher gezeigten Beispiele stellen nur einen kleinen Ausschnitt der gefundenen Interpretationsdifferenzen und damit einhergehenden Umgehungsmöglichkeiten dar. Weitere Varianten sind die Nutzung von nicht standardisierten Kodierungen wie uuencode oder y-encode, innovative Nutzung von Leerzeichen oder Zeilenumbrüchen an unerwarteten

Stellen oder einfach auch nur die Kodierung von binären Anhängen mit dem typischerweise für Texte genutzten Quoted-Printable statt Base64. Im Zuge der Forschungen haben wir eine umfangreiche Testsuite entwickelt, die einen stark automatisierten Test von Analysesystemen ermöglicht und so in kurzer Zeit eine Vielzahl von Problemen auch bei kommerziellen Systemen gefunden hat.

3 Ursachen und Lösungsansätze

Neben den bereits erwähnten tiefgreifenden Problemen der Standards selber, sehen wir ein mangelndes Verständnis bei den Autoren von Analysesystemen für diese Art von Umgehungen. Da zumeist davon ausgegangen wird, dass Mails eindeutig interpretierbar sind, werden Verfahren und Bibliotheken zur Extraktion der Inhalte genutzt, welche für Endsysteme nutzbar sind, aber nicht zur Analyse speziell präparierter Mails taugen.

Ein naiver Ansatz zum Umgang mit dem Problem wäre, alle potentiell ambivalenten Mails zu blockieren. Unsere Erfahrungen aus der Praxis zeigen jedoch, dass durch die große Vielfalt mailgenerierender Systeme derart problematische Mails in der Realität häufiger als gehofft auftreten. Analog legen wir auch keine Hoffnung in eine potentielle Überarbeitung der MIME-Standards, da hier die Vielfalt aller existenten Implementierungen berücksichtigt werden müsste und der neue Standard so noch komplexer und unbrauchbarer als bisher wird.

Die beste Option sehen wir in einer Konvertierung problematischer Mails in eine Form, bei der nur noch die von allen Systemen gleich verstandenen Kernfeatures der Standards benutzt werden. Da eine derartige Normalisierung die Modifikation der Daten bedeutet, kann sie nur in Systemen mit aktiver Analyse, wie Mail-Filtern und Applikations-Layer-Firewalls eingesetzt werden. Systeme mit passiver Analyse, wie Antivirus-Produkte, Intrusion-Detection-Systeme oder paketinspizierende Firewalls, müssten stattdessen sämtliche mögliche Interpretationsvarianten durchprobieren. Dieses scheitert in der Praxis aber bereits daran, dass nicht alle Varianten bekannt sind.

Literaturverzeichnis

[FB96a] Freed, Ned; Borenstein, Nathaniel S.: , Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. Internet Requests for Comments, November 1996. http://www.rfc-editor.org/rfc/rfc2045.txt.

[FB96b] Freed, Ned; Borenstein, Nathaniel S.: , Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. Internet Requests for Comments, November 1996. http://www.rfc-editor.org/rfc/rfc2046.txt.

[FM97] Freed, N.; Moore, K.: , MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations. Internet Requests for Comments, November 1997.

[Mo96] Moore, Keith: , MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text. Internet Requests for Comments, November 1996. http://www.rfc-editor.org/rfc/rfc2047.txt.