Sehen heißt glauben – Aufdeckung von Webseiten Manipulation

Tobias Urban · Norbert Pohlmann

Institut für Internet-Sicherheit – if(is)
Westfälische Hochschule
{urban | pohlmann}@internet-sicherheit.de

Zusammenfassung

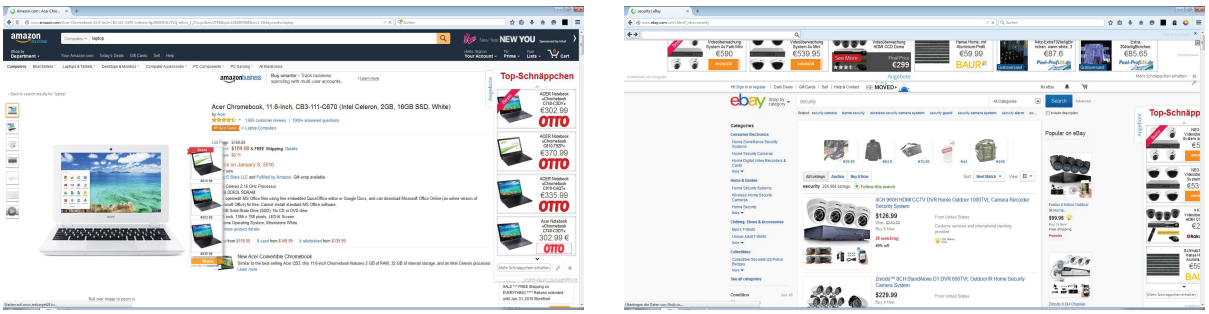
Die Darstellung von Webseiten wird zunehmend in unberechtigter Weise clientseitig durch Malware manipuliert. Zurzeit versuchen gängige Antiviren-Produkte, die Malware auf einem Endgerät aufzudecken, welche die Webseiten manipuliert. Trotz dieser Bemühungen gibt es immer mehr schadhafte Programme, denen genau dies gelingt. Diese Arbeit beschäftigt sich nicht mit der Identifizierung von Malware, welche eine clientseitige Manipulation von Webseiten vornimmt. Wir stellen ein Verfahren vor, welches zur Erkennung clientseitiger Webseiten-Manipulation eingesetzt werden kann. Dazu werden Informationen, über den strukturellen Aufbau einer Website, von vielen verteilten Endsystemen direkt beim Besuchen einer Webseite an ein zentrales Manipulations-Erkennungs-System (MES) übermittelt. Dieser Server kann anhand dieser Informationen entscheiden, ob eine Webseite, die gerade lokal auf einem Endgerät angezeigt wird, manipuliert wurde. Wir haben 17 Malware Samples unterschiedlicher Familien genutzt, um über 2.100 Webseiten zu manipulieren. Dabei zeigen wir, dass die meisten Manipulationen effizient und effektiv von unserem Manipulationserkennungsserver erkannt werden können.

1 Einleitung

Durch ein immer stärker werdendes Aufkommen von Diensten in der Cloud bieten Web-Browser einen immer größer werdenden Zugang zu Informationen, ermöglichen eine hohe Zahl von sozialen Interaktionsmöglichkeiten und bieten die Möglichkeit, private Daten auf entfernten Servern zu speichern. Diese breite Funktionalität des Browsers bringt aber auch neue Sicherheitsrisiken für die Nutzer mit sich. Beispiele für solche Gefahren sind der Diebstahl von Daten eines Nutzers (z.B. Kreditkarteninformationen), die Bildung von Nutzungsprofilen des Nutzers oder der Versuch, das Endgerät des Nutzers mit Schadsoftware zu infizieren, welche es erlaubt, diesen zu übernehmen und fernzusteuern. Eine weitere Gefahr, welche in der Vergangenheit drastisch zugenommen hat, ist die unberechtigte clientseitige Manipulation von Webseiten („Web-Injects“) durch einen Angreifer, der Malware auf dem Endgerät des Nutzers installiert hat. Diese Manipulationen reichen beispielsweise von dem Einfügen von Werbung auf beliebigen Webseiten bis hin zu der Manipulation von Online-Banking Webseiten. Die Manipulation von Webseiten, insbesondere das Einfügen von Werbung, hat in der letzten Zeit dramatisch zugenommen. Unter den 20 am häufigsten entdeckten bösartigen Objekten (verantwortlich für über 95% der Angriffe im Internet), welche von der Antiviren-Software von Kaspersky Lab entdeckt wurden, kam es zu einer explosionsartigen Erhöhung von AdWare Produkten. So steigerte sich die Verbreitung von AdWare von lediglich 4,91% im ersten Quartal 2014 [CEGU14] auf 34,07% im dritten Quartal 2015 [EGU+15].

P. Schartner (Hrsg.) · D•A•CH Security 2016 · syssec (2016) 63-74.

Die Manipulationen werden durch sog. „Man-in-the-Browser“ Angriffe durchgeführt. Bei diesem Angriffsvektor übernimmt der Angreifer die Kontrolle über die Browser-Kommunikation des Opfers und somit über die Darstellung der Webseite sowie alle Informationen die zwischen Client und Server ausgetauscht werden. Zunächst haben vor allem Banking-Trojaner den Angriffsvektor genutzt, um Webseiten von Kreditinstituten für die Sichtweise des Nutzers zu manipulieren. Zunehmend wird der Angriffsvektor auch von AdWare genutzt, um unberechtigt clientseitig Werbung in Webseiten einzufügen. Abbildung 1 zeigt Beispiele für manipulierte Webseiten.



(a) Amazon **(b)** Ebay

Abb. 1: Zwei Beispiele für manipulierte Webseiten

Die Hauptaugenmerke dieser Arbeit liegen auf den folgenden Punkten:

- Aufbau eines repräsentativen Datensets, welches manipulierte und nicht manipulierte Webseiten enthält. Das Datenset stellt ein realistisches Set von Webseiten dar, die auch bei einem praktischen Einsatz des Manipulationserkennungsserver auftreten würden.
- Vorstellen eines Konzeptes, das zum maschinellen Lernen des strukturellen Aufbaus einer Webseite eingesetzt werden kann. Der strukturelle Aufbau ist hier die Menge der HTML-Tags, welche die HTML-Seite eindeutig beschreiben.
- Entwicklung und Evaluierung eines Systems zur Manipulationserkennung, das effizient und effektiv clientseitige Manipulationen auf Webseiten aufdeckt.

Der Rest dieser Ausarbeitung ist wie folgt aufgebaut: In Abschnitt 2 wird das Konzept der Arbeit vorgestellt und die wichtigsten Begriffe werden definiert. Die Umsetzung des Konzeptes wird in Abschnitt 3 beschrieben. Zum Ende wird ein Fazit gezogen (Abschnitt 4), und es werden verwandte Arbeiten vorgestellt (Abschnitt 5).

2 Terminologie und Methodik

2.1 Begrifflichkeiten

Wir wollen diese Arbeit mit der Definition von drei wichtigen Begriffen beginnen.

2.1.1 Backlink

Zunächst soll die Maßzahl „Backlink“ definiert werden. Unter einem Backlink wird die Anzahl an Verweisen auf eine Webseite verstanden, welche von anderen Seiten zu dieser Seite existieren. Beispiele für die Anzahl von Backlinks einer Webseite sind (1) www.amazon.de - 121.501

Backlinks (2) www.adbasta.com - 5 Backlinks (3) www.syssec.at - 41 Backlinks. Eine geringe Anzahl an Backlinks deutet also auf eine geringe Popularität und Vertrauenswürdigkeit einer Webseite hin. Die Maßzahl wird im weiteren Verlauf genutzt, um Quellen zu bewerten aus denen Inhalte geladen werden. Es wird angenommen, dass Quellen, aus denen Malware Inhalte lädt, eine geringe Anzahl an Backlinks aufweisen.

2.1.2 HTML-Page und Webpage

Innerhalb dieser Arbeit definieren wir eine HTML-Seite (HTML-Page) über das Tupel HTML-Seite $:= (w, path, p, \Delta, t)$ mit w der Webpage, zu der diese Seite gehört, $path$ dem Path-Teil der URL, p den HTTP-GET-Parametern, Δ der Menge an HTML-Tags, welche die Seite beschreiben und t dem Zeitpunkt an dem die Webseite aufgerufen wurde. Innerhalb dieser Arbeit wird unter einer Webpage eine Sammlung von mehreren HTML-Seiten, die zu derselben Domain gehören, verstanden. So ergibt sich das Tupel webpage $:= (url, \Phi)$ mit url dem Resource-name und Φ allen HTML-Seiten, die zu dieser Webpage gehören.

2.2 Konzept des Manipulations-Erkennungs-Systems

Die Idee, auf der diese Arbeit beruht, ist die, dass verschiedene Besucher einer Webseite ihr Wissen über den Aufbau dieser Seite miteinander teilen. Durch diese geteilten Informationen ist es für jeden Endnutzer möglich, für die Webseite, die er gerade betrachtet, zu entscheiden, ob diese durch Malware, die auf dem Endgerät des Nutzers installiert ist, manipuliert wurde oder nicht. Das entwickelte Manipulation Detection System erhält von den Endnutzern Beispiele von dem strukturellen Aufbau von Webseiten und prüft, ob Webseiten, die von Endnutzern besucht werden, von diesen beispielhaften Aufbauten abweichen.

Für jede Webseite muss das Manipulations-Erkennungs-System (MES) zunächst den strukturellen Aufbau der Seite verstehen, um so ein passendes Modell für diese Seite erstellen zu können. Dieses Trainieren des MESs geschieht durch Informationen zu dem Aufbau einer Seite, die von verschiedenen Endgeräten an das MES übermittelt werden. Bei dem Training wird angenommen, dass die Webseiten nicht manipuliert wurden. Liegt eine hinreichend große Anzahl von unterschiedlichen Beispielen zu einer Webseite vor, kann ein passendes Modell aufgestellt werden (vgl. Schritt 1 und 2 in Abbildung 2). Die grundlegende Arbeitsweise des entwickelten MESs, für Endnutzer, deren Endgerät infiziert wurde (Schritt 4-6) und für Nutzer, deren Darstellung nicht manipuliert wird (Schritt 1-3), zeigt Abbildung 2 und wird im Folgenden kurz beschrieben:

1. Mehrere Nutzer, die verschiedene Endgeräte verwenden, rufen eine Webseite auf.
2. Jeder Endnutzer sendet Informationen zu dem strukturellen Aufbau der Webseite, die auf seinem jeweiligen Endgerät dargestellt wird, an das MES.
3. Das MES prüft die übermittelten Daten gegen das interne Modell, welches es von der Webseite angelegt hat und teilt jedem einzelnen Client mit, dass keine Manipulation erkannt wurde.
4. Ein Endnutzer, dessen Endgerät mit Malware infiziert ist, die Webseiten clientseitig manipuliert, besucht eine Webseite.
5. Die Informationen über den strukturellen Aufbau der manipulierten Seite, inklusive der Manipulationen, werden an das MES gesendet.
6. Bei der Prüfung des übermittelten Aufbaus gegen das interne Modell der Seite stellt das

MES fest, dass eine Manipulation vorliegt. Das MES warnt den Endnutzer, dass er eine manipulierte Seite betrachtet.

Die Kommunikation zwischen dem Endgerät des Nutzers und dem MES findet über ein speziell entwickeltes Add-On für den Web-Browser („MES-Client“) statt.

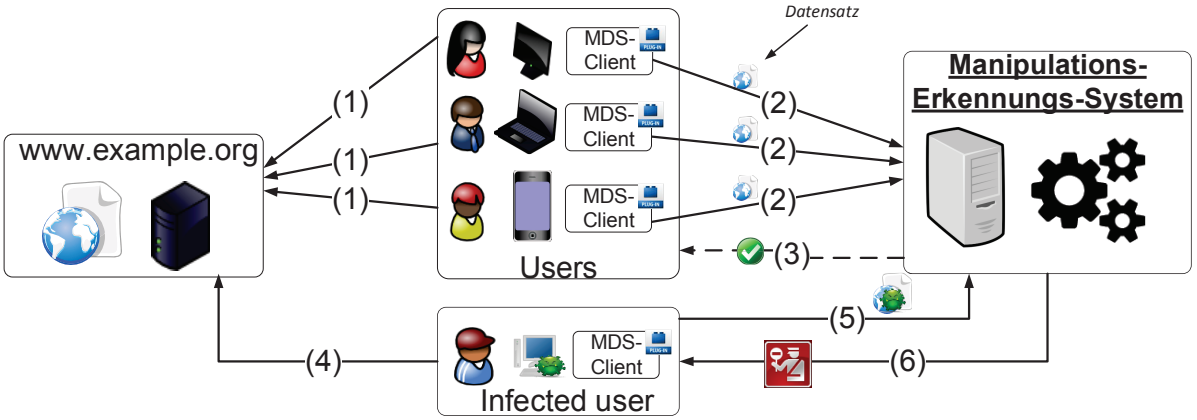


Abb. 2: Übersicht zu dem Konzepts des MESs

2.3 Datenbasis

Der Datenbestand, der zum Trainieren und Testen des MESs genutzt wird, ist von besonderer Bedeutung für die Bewertung der Effektivität des MESs. Die Datenbasis, die innerhalb dieser Arbeit zum Einsatz kommt, besteht aus insgesamt 4.635 einzelnen HTML-Seiten, welche von 42 unterschiedlichen Domains gesammelt wurden. Alle betrachteten Malware Samples, bis auf eines, haben nur bestimmte, äußerst beliebte, Webseiten manipuliert. Daher ist es nicht trivial möglich, weitere sinnvolle Samples von anderen Domains zu sammeln. Für jede Domain wurden, falls möglich, die 100 beliebtesten HTML-Seiten (z.B. Videos oder Produkte) als Sample gewählt. Diese Datensätze werden in zwei Sets aufgeteilt. Ein Trainingsset, welches aus 1.659 Datensätzen (HTML-Seiten) besteht und zum initialen Trainieren des MESs genutzt wird. Das Testset besteht dementsprechend aus 2.976 Datensätzen. Diese beiden Sets werden im Folgenden genauer beschrieben.

2.3.1 Trainingsset

Das Trainingsset besteht lediglich aus nicht veränderten HTML-Seiten. Für jede HTML-Seite wurden ein bis drei Beispiele für den Aufbau der Seite gesammelt. Diese Variante erlaubt es ein genaues Wissen über den allgemeinen legitimen Aufbau von Webseiten zu erlangen. Die Entscheidung, ob eine Webseite manipuliert wurde, besteht darin zu testen, ob diese zu dem gelernten Schema der Webseite passt. Sollte dies nicht der Fall sein, kann von einer Manipulation ausgegangen werden. Da das MES nicht aus manipulierten Daten lernt, reduziert sich auch die Gefahr, dass neue Varianten von Manipulationen nicht erkannt werden. Für das Trainingsset ist von besonderer Bedeutung, dass es aus realistischen Daten besteht, die auch bei einem praktischen Einsatz des MESs auftreten würden. Bei dem Sammeln der Webseiten des Datensets wurden die folgenden Merkmale verändert, um ein passendes Set zu erstellen:

- **Unterschiedliche User-Agenten**: Der größte Unterschied, der bei der Darstellung einer Seite auftreten kann, ist der, dass verschiedene User-Agenten eingesetzt werden.

Die Seiten, innerhalb des Trainingssets, wurden mit unterschiedlichen Versionen der drei gängigsten Browser (Mozilla Firefox, Microsoft Internet Explorer und Google Chrome) gesammelt.

- **Unterschiedliche Herkunftsländer**: Das Land, aus dem eine Webseite aufgerufen wird, kann Einfluss auf die Darstellung derselben haben. Das Copyright von gewissen Inhalten variiert in unterschiedlichen Ländern deutlich, was zur Folge hat, dass Inhalte unterschiedlich dargestellt werden können (z.B. Musikvideos, die auf der Video-Plattform YouTube eingestellt werden). Das Trainingsset wurde aus zwei unterschiedlichen Ländern gesammelt, zum einen aus den Vereinigten Staaten von Amerika und zum anderen der Bundesrepublik Deutschland.
- **Unterschiedliche Sprachen**: Ergänzend wurde ebenfalls die Sprache des User-Agenten zufällig zwischen Deutsch und amerikanischem Englisch gewählt. So werden mögliche unterschiedliche Darstellungen, die aus der gewählten Sprache resultieren, berücksichtigt.

2.3.2 Testset

2.976 unterschiedliche Datensätze befinden sich im Testset. Diese Datensätze teilen sich nochmals in 2.135 manipulierte und 841 nicht manipulierte Webseiten auf. Zu jeder manipulierten Webseite existiert ein nicht manipuliertes Äquivalent in dem Datenset. Die nicht manipulierten HTML-Seiten wurden auf dieselbe Weise wie die Webseiten des Trainingssets gesammelt. Die manipulierten Webseiten wurden lediglich unter Einsatz des Mozilla Firefox Browsers in den Versionen 36 bis 38 gesammelt. Die Manipulation der Webseite soll möglichst schnell erkannt werden, es ist daher nötig, dass das Testset nicht nur aus Elementen besteht, welche bereits vollständig manipuliert sind, sondern deren Manipulation noch im Gange ist. Zur Umsetzung dieser Anforderung speichert ein entwickeltes Add-On alle fünf Sekunden das gesamte DOM, falls eine Änderung am DOM der besuchten Webseite vorgenommen wurde (Pro Webseite und Malware-Sample wird nur ein Datensatz erzeugt). Die Seiten werden nach 5-30 Sekunden wieder geschlossen. Dies hat zur Folge, dass aktive Malware nicht unbedingt alle Manipulationen durchführen konnte bzw. nicht alle Inhalte geladen wurden. Bei der Analyse treten also „unterschiedliche Phasen“ der Manipulation auf. Beispielsweise kommen leere iFrame Elemente vor, welche zu einem späteren Zeitpunkt noch mit Inhalt gefüllt werden.

2.4 Angriffe auf das MES

Aufgrund der Verteidigungsmechanismen gegen ungewollte Manipulationen, die das entwickelte MES bietet, ist davon auszugehen, dass das entwickelte Add-On selbst zum Ziel der Angreifer wird. Dabei bietet das Add-On verschiedene Angriffsflächen, die von Malware ausgenutzt werden können. Drei mögliche Angriffe werden im Folgenden kurz dargestellt:

- **Deaktivieren des Add-Ons**: Um Webseiten weiterhin unbemerkt manipulieren zu können, kann Malware versuchen das Add-On zu deaktivieren oder zu löschen. Dies könnte durch Software verhindert werden, die im Kernel-Modus ausgeführt wird und die Integrität sowie Aktivität des Add-Ons prüft und den Nutzer warnt, sollte das Add-On nicht mehr korrekt arbeiten.
- **Manipulation der Übertragenden Daten**: Die Kommunikation zwischen dem entwickelten Add-On und dem Webserver, welcher die Analyse vornimmt, geschieht über einen TLS-gesicherten Kanal. Trotzdem hat ein Angreifer die Möglichkeit, die übertragenen Daten zu verändern. Dies gibt dem Angreifer die Möglichkeit die Antwort des MES

zu verändern, sodass diese angibt, dass keine Manipulation vorliegt oder die Daten, die an das MES gesendet werden, zu manipulieren. Die einfache Änderung der Antwort auf dem Client könnte durch eine digitale Signatur der Nachricht durch das MES deutlich erschwert werden.

- **Test von Manipulationen gegen das MES**: Entwickler von AdWare oder Banking-Trojanern können auch die Veränderungen an Webseiten direkt gegen das MES testen. Verschiedene Manipulationen können immer wieder gegen das MES getestet werden, um festzustellen welche Manipulationen durch das entwickelte MES aufgedeckt werden und welche unentdeckt bleiben. Somit können Manipulationen geschaffen werden, die von dem MES unerkannt bleiben. Eine Verteidigung gegen diese Angriffsstrategie könnte sein, dass eine IP-Adresse gesperrt wird, wenn das System feststellt, dass von dieser immer wieder Anfragen zu der selben manipulierten Webseite gestellt werden.

3 Proof of Concept

3.1 Clustering

Aus der Definition einer HTML-Seite geht hervor, dass Webseiten mit unterschiedlichen GET-Parametern als unterschiedliche Seiten aufgefasst werden. Dies hat direkt zur Folge, dass nicht davon ausgegangen werden kann, dass alle Seiten, die von Nutzern besucht werden, bereits von dem MES gelernt wurden. Allerdings ist davon auszugehen, dass ähnliche Seiten bereits besucht wurden. Beispielsweise ist der Inhalt der Seiten: (1) https://www.google.com/search?q=**women's**%20**world**%20**cup** und (2) https://www.google.com/search?q=**le**%20**tour**%20**de**%20**france** unterschiedlich, der Aufbau der Seiten ist aber sehr ähnlich, was die Möglichkeit bietet, aus dem Aufbau einer der Seiten etwas über den Aufbau der anderen Seite zu lernen. Um diese Eigenschaft optimal nutzen zu können, werden die Trainingsdaten in Cluster zusammengefasst, welche jeweils Seiten einer Domain enthalten, die einen ähnlichen Aufbau haben. Falls eine Webseite überprüft werden soll, welche dem MES nicht bekannt ist, kann diese gegen die Seiten eines Clusters geprüft werden, zu welchem die zu prüfende Seite zugeordnet werden kann. Implizit folgt so, dass eine Webseite nur dann nicht geprüft werden kann, wenn diese keinem Cluster zugeordnet werden kann und dem MES somit nicht bekannt ist. So wird auch die Größendifferenz zwischen Testset und Trainingsset egalisiert.

3.1.1 Erzeugung der Cluster

Im Folgenden wird dargestellt, in welcher Art die benötigten Cluster erzeugt werden. Für das Clustern der Webseiten wurde ein agglomeratives hierarchisches Clusterverfahren entwickelt (siehe Algorithmus 1). Hierarchische Clusterverfahren haben sich als effektiv für das Clustern von Dokumenten erwiesen (vgl. [Wil88] und [ZKF05]). Dies bedeutet, dass für jede HTML-Seite, einer Domain, zunächst ein initiales Cluster erstellt wird, in dem sich nur diese HTML-Seite befindet (L: 1). Diese kleinen Cluster werden beim Durchlaufen des Algorithmus zusammengefasst, wenn der Abstand zwischen den Clustern nicht größer als Ω ist (L: 2-7). Die Distanz zwischen zwei Clustern wird von der Methode `distance()` bestimmt, welche in Algorithmus 2 beschrieben wird. Ω wurde anhand der Trainingsdaten heuristisch bestimmt.

Der wichtigste Teil bei einem hierarchischen Cluster Verfahren ist die Bestimmung der Distanz zwischen zwei Clustern. Zum Bestimmen der Distanz zwischen zwei Clustern werden die N häufigsten HTML-Tags („Features“) genutzt, die auf den Seiten des Clusters vorkommen. Jedes Cluster wird eindeutig über N Features definiert. Zusätzlich werden zu den Features für jedes

Algorithmus 1 : Cluster generation

input : A webpage that consist of n HTML-pages
output : A list of clusters

```
[1] clusters ← CreateInitialClusters()
[2] while new cluster was generated do
[3]     foreach outerCluster ∈ clusters do
[4]         foreach innerCluster ∈ clusters do
[5]             if outerCluster ≠ innerCluster ∧ distance(outerCluster, innerCluster) ≤ Ω then
[6]                 clusters ← clusters \ {innerCluster} \ {outerCluster}
[7]                 clusters ← clusters ∪ CombineCluster(innerCluster, outerCluster)
[8] return clusters
```

Cluster noch die weiteren im Cluster vorkommenden HTML-Tags verwaltet, die zwar auf den Webseiten vorkommen, aber nicht zu den N häufigsten gehören („Non-Features“). Es werden lediglich die N häufigsten HTML-Tags verwendet, da es bei der Berücksichtigung von allen Tags zu sehr starken Schwankungen, vor allem bei „seltenen“ Tags, kommt. Algorithmus 2 beschreibt die Berechnung der Distanz zwischen zwei Clustern. Zunächst wird die gewichtete absolute Differenz der Features berechnet (L: 4-6). Sollte ein Feature des ersten Clusters kein Feature des zweiten Cluster sein, wird der Non-Feature Wert, mit entsprechender Gewichtung, für die Distanzberechnung verwendet (L: 7-9).

Wenn ein Cluster zu einer zu prüfenden Seite gesucht wird, wird der Abstand der Seite zu jedem Cluster der entsprechenden Domain berechnet. Es wird das Cluster verwendet, welches den geringsten Abstand zu der Seite hat. Das entwickelte Verfahren zur Bestimmung der Distanz ist nicht symmetrisch, was bedeutet, dass `distance(c1, c2)` und `distance(c2,c1)` nicht unbedingt dasselbe Ergebnis liefern müssen.

Algorithmus 2 : distance

input : Two clusters, c1 and c2. Each cluster consists of two maps which contain the feature and non-feature attributes of the cluster.
output : The distance between the two clusters

```
[1] distance ← 0
[2] featWeight ← N
[3] foreach feature f of c1 do
[4]     if f ∈ c2.features then
[5]         distance ← distance + featWeight · |f.value − c2.FeatureVal(f)|
[6]         featWeight ← featWeight−1
[7]     else if f ∈ c2.nonFeatures then
[8]         distance ← distance+ featWeight/2 · |f.value − c2.NonFeatVal(f)|
[9]         featWeight ← featWeight−1
[10]    else
[11]        return ∞
[12] return distance
```

3.2 Aufdeckung der Manipulationen

Für die Aufdeckung der Manipulationen wurden Mechanismen entwickelt, welche spezielle Tags untersuchen und ermitteln, ob die gerade betrachtete Webseite eingefügte Tags von genau diesem Typ enthält. In diesem Abschnitt wird von gelernten (learned bzw. known) Elementen auf einer HTML-Seite gesprochen. Dies bezieht sich auf die Daten, die aus den Trainingsdaten genutzt werden, um die besuchte HTML-Seite (observed bzw. visited) zu bewerten. Zusätzlich wird „gelernte HTML-Seite“ als Synonym für das Cluster genutzt, zu dem die HTML-Seite zugeordnet wurde.

3.2.1 Analyse von iFrame Tags

iFrames bieten dem Angreifer ein einfaches Mittel, um dynamische Inhalte in eine Webseite einzubinden. Aus diesem Grund ist eine solide Erkennung von unberechtigt eingefügten iFrames für den Schutz des Nutzers äußerst wichtig und wird im Algorithmus 3 dargestellt.

Algorithmus 3 : iFrameEvaluator

input : knownSmallExternalFrames(KsmallExt), knownEmptyFrames,
visitedSmallExternalFrames(VsmallExt), vistedEmptyFrames
output : Probability of the given HTML-site containing inserted iFrames.

```
[1] prob ← 0
[2] interestingExtFrames ← FindNewTargets(KsmallExt, VsmallExt)
[3] interestingEmptyFrames ← vistedEmptyFrames \ knownEmptyFrames
[4] if interestingExtFrames ≠ ∅ then
[5]     prob ← prob +1
[6] foreach frame ∈ interestingEmptyFrames do
[7]     prob ← prob + α / |knownEmptyFrames|
[8] return min(prob, 1)
```

Algorithmus 3 ermittelt zunächst alle ’kleinen’ iFrames, welche auf der gelernten und der besuchten HTML-Seite vorkommen und ihren Content aus einer externen Quelle laden. Unter ’kleinen’ iFrames werden alle iFrames verstanden, deren Größe nicht angegeben wurde oder deren Fläche kleiner als 25pixel^2 ist. Bei der Prüfung einer HTML-Seite werden leere iFrames nicht direkt als Manipulation eingestuft, allerdings wird eine zu hohe Anzahl neuer leerer iFrames als mögliche Manipulation angesehen (L: 6-8). Der Wert $|KnwonEmptyFrames|$ wird aus dem Cluster der Webseite gelernt und gibt an, wie viele leere iFrames auf nicht manipulierten Seiten des Clusters vorkommen. α ist die Anzahl der leeren iFrames auf der besuchten Webseite. Die Methode `FindNewTargets()` ermittelt alle iFrames (des zweiten Arguments), die ihren Content aus einer Quelle laden, welche nicht in den Trainingsdaten der HTML-Seite (erstes Argument) genutzt wird. Wird ein iFrame gefunden, der Inhalte aus bisher unbekannten Quellen lädt, wird die Manipulation der vorliegenden Seite angenommen (L: 4+4).

3.2.2 Analyse von JavaScript Tags

Zur Aufdeckung von eingefügten JavaScript-Elementen, werden alle Skript-Elemente, die eine geringe Anzahl an Backlinks(<500) aufweisen, ermittelt (L: 1+2). Anschließend werden die Skript-Elemente untersucht, welche ihren Skript-Code von einer bisher ungenutzten Quelle

laden (`FindNewSources()` - L: 3-6). Sollten diese Elemente keinen bösartigen Code enthalten, werden die neuen JavaScript- Elemente der besuchten HTML-Seite untersucht, die in reiner Textform auf der Seite vorhanden sind (L: 7-10). Die Methode `isEvilScript()` prüft, ob ein Skript bösartig ist oder nicht. Dabei wird geprüft, ob das Skript die Schlüsselwörter: (`eval()` und `document.write()`) oder `append()` enthält. Forschungsarbeiten haben gezeigt, dass diese Wörter unter anderem meist nur in bösartigen Skripten vorkommen (siehe z.B. [CKV10] oder [CLZS11]).

Algorithmus 4 : ScriptEvaluator

input : learned and observered JavaScript elements of a specific HTML-page.
output : Probability of the given site containing inserted JavaScript elements.

```
[1]  learnedScripts ← getLowBackLinkScripts(AllLearnedScripts)
[2]  observedScripts ← getLowBackLinkScripts(AllObservedScripts)
[3]  scriptsToExamine ← FindNewSources(learnedScripts, observedScripts)
[4]  foreach script ∈ scriptsToExamine do
[5]      if isEvilScript(script) then
[6]          return 1
[7]  scriptsToExamine ← getTextScripts(observedScripts \ learnedScripts)
[8]  foreach script ∈ scriptsToExamine do
[9]      if isEvilScript(script) then
[10]         return 1
[11] return 0
```

3.2.3 Analyse von Attributen

Die bisher vorgestellten Verfahren analysieren direkt einige spezielle Tags, die zur Manipulation von Webseiten genutzt werden können. Ein anderer Ansatz, zur Erkennung von Manipulationen ist, spezielle Eigenschaften aller Tags einer Webseite zu analysieren. Dies bietet den Vorteil, dass alle Tags betrachtet werden, also auch Tags, welche den Aufbau der Seite beschreiben, aber nicht direkt sichtbar sind (z.B. `<div>` oder `<span>` Tags). In der vorliegenden Arbeit wurden die Eigenschaften `id` und `class` gewählt, um anhand dieser Manipulationen aufzudecken. Mit dem Theorem von Bayes wird versucht, unberechtigt eingefügte Tags zu identifizieren. Da innerhalb dieser Arbeit eine One-Class Klassifikation durchgeführt wird, muss das Theorem von Bayes an diese Problemstellung angepasst werden (siehe auch [MY02]). Im Folgenden werden die betrachteten Eigenschaften mit Wort (ω) bezeichnet. Eine HTML-Seite wird als Dokument (d) bezeichnet. Die Wahrscheinlichkeit, ob ein Dokument zu einer bestimmten Klasse (E) gehört, wird wie folgt bestimmt:

$$P(d) = p(d|E) \tag{1}$$

E ist in diesem Fall die Klasse „nicht manipuliert“. Zur Berechnung dieser Wahrscheinlichkeit ist es essentiell zu wissen, mit welcher Wahrscheinlichkeit ein Wort ω_i in dem Dokument d vorkommt. Diese einzelnen Wahrscheinlichkeiten für die jeweiligen Wörter werden durch eine Multiplikation zusammengefasst.

$$p(d|E) := \prod_i p(\omega_i|E) \tag{2}$$

Die Wahrscheinlichkeit, dass ein Wort ω_i in einem Dokument vorkommt, wird durch das Verhältnis der Wörter, die in einem Dokument vorkommen, zu den Schlüsselwörtern bestimmt.

$$p(\omega_i|E) := \frac{n_\omega + 1}{n + m} \tag{3}$$

Hier ist n_ω, die Häufigkeit von ω in d, m die Anzahl der unterschiedlichen Eigenschaften und n die Summe aller betrachteten Eigenschaften.

3.2.4 Weitere Analysen

Zusätzlich zu den vorgestellten Verfahren wurden noch Mechanismen für die Aufdeckung von Hyperlinks, Bildern, Eingabeelementen und Formularen entwickelt. Diese Mechanismen arbeiten prinzipiell wie die Analyse der iFrames und JavaScript Elemente. Es werden neue Elemente gesucht, die Inhalte aus Quellen mit einer geringen Anzahl von Backlinks laden bzw. an eine solche senden. Bei Formularen wird zusätzlich nach neuen Formularfeldern gesucht. Diese Verfahren können, aus Platzgründen, innerhalb dieses Papers leider nicht genauer dargestellt werden.

4 Fazit

4.1 Ergebnisse der Analyse

Tabelle 1 zeigt das Ergebnis der Analyse. Dabei werden die Analyseergebnisse der jeweiligen Klassifikatoren aufgezeigt, wenn nur diese genutzt werden und wenn alle Klassifikatoren sequenziell hintereinander genutzt werden. Zur Beschleunigung des Entscheidungsprozesses (in der sequenziellen Bearbeitung) wird ein Entscheidungsbau eingesetzt. Der Baum ist anhand der Erkennungsraten der jeweiligen Algorithmen aufgebaut (höchste Erkennungsrate zuerst).

Tab. 1: Analyseergebnis des MESs

	iFrame	JavaScript	Input	Image	Link	Attrib.	Form	**All**
Correct classification	2.243	2.173	1.464	2.463	1.735	1.324	840	**2.840** (95,82 %)
Wrong classification	691	66	761	459	1.199	1.610	2.094	**82** (2,78 %)
False positive	0	3	10	1	8	9	3	**29** (0,98 %)
False negative	691	758	1.460	458	1.191	1.601	2.091	**53** (1,79 %)
Impossible classification	42	42	42	42	42	42	42	**42** (1,72 %)

Die 42 nicht möglichen Klassifikationen kommen von Webseiten, die keinem Cluster zugeordnet werden konnten. 82 Webseiten (2,78%) werden falsch von dem MES eingestuft. Eine händische Analyse hat gezeigt, dass false positive Fehler auf Seiten auftreten, die Seiten mit wenigen Backlinks als Suchergebnis zurückliefern (z.B. Hinweise auf lokale Geschäfte). Die false positive Fehler bei der Input Analyse kommen vor, wenn ungewöhnlich viele input Felder auf der Webseite vorkommen. Bei einem größeren Trainingsset ist davon auszugehen, dass die Fehlerzahl hier deutlich minimiert wird. False negative Klassifikationen entstehen, wenn kein manipuliertes Element auf einer Webseite vorkommt (das von dem jeweiligen Klassifikator untersucht wird) oder wenn die Quellen, aus denen die Inhalte geladen werden, nicht mehr zur Verfügung stehen und somit eine Analyse nicht mehr möglich ist, bzw. die Quellen nicht mehr bewertet werden können.

4.2 Laufzeit verhalten des MES

Alle Berechnungen wurden auf einer virtuellen Maschine durchgeführt, die mit einem 2,4Ghz Prozessor (2 Kerne) und 4GB RAM ausgestattet ist. Tabelle 2 zeigt das Laufzeitverhalten des Systems für zwei Fälle auf: (A) wenn alle Trainings und Testdaten verwendet werden oder (B) für ein kleineres Set bestehend aus 234 Trainingsseiten und 740 Testseiten. Alle Mechanismen wurden sequentiell durchgeführt. Das Clustering aller Webseiten ist mit fast 6 Minuten (Fall A) noch vertretbar, da es nur beim initialen Starten des MESs und bei einem möglichen erneuten Trainieren durchgeführt werden muss. Bei einem größeren Trainingsset würde die benötigte Zeit für das Clustering exponentiell zunehmen. Die Schwankungen in der Analysezeit entstehen vor allen durch die absolute Anzahl der untersuchten Elemente auf einer Webseite. Bei einem größeren Testset ist nicht von einer starken Änderung dieses Laufzeitverhaltens auszugehen.

Tab. 2: Laufzeitverhalten des MESs

	A	B		A	B
Avg analysis time:	27 ms	23 ms	Max analysis time:	5.069 ms	5.387 ms
Median:	0 ms	0 ms	Min analysis time:	0 ms	0 ms
Standard deviation:	271 ms	240 ms	Clustering took:	358.407 ms	38.776 ms

5 Verwandte Arbeiten

Die Identifikation von bösartigen Webseiten, welche gewollt oder ungewollt, das Endgerät bei dem Besuch der Webseite mit Malware infizieren („Drive-By-Downloads“), ist ein aktives Forschungsfeld. Dabei wird zwischen statischen (z.B. [SWK08] oder [CCVK11]) und dynamischen Methoden (z.B. [CKV10] oder [SS08]) unterschieden. Die Arbeiten [SS08] und [CCVK11] nutzen, wie auch das vorgestellte Verfahren, die HTML-Elemente einer Seite, um zu entscheiden, ob diese bösartig ist oder nicht.
Charles Reis et. al untersuchen in [CSNT08] die Manipulation von Webseiten die per HTTP übertragen werden. Dabei zeigen sie, dass ca. 1% aller Webseiten während des Transports manipuliert werden. Dabei wird nicht zwischen gewollten Manipulationen (z.B. Entfernen von Werbung) oder ungewollten Manipulationen unterschieden.
In [TBG+15] stellen Kurth Thomas et. al. ebenfalls ein System vor, welches eingefügte Hyperlinks, iFrames oder JavaScript-Elemente auf Webseiten aufdecken soll. Dazu wird dem Client eine Whitelist an Quellen übermittelt, welche auf der Webseite genutzt werden.
Der Unterschied dieser Arbeit zu den genannten Arbeiten ist, dass diese unabhängig von der besuchten Webseite und für alle Webseiten einsetzbar ist. Dies liegt daran, dass eine zusätzliche Entität eingesetzt wird und an dem Webserver direkt keine Änderungen vorgenommen werden.
Content Security Policys (kurz CSP) sind ein Sicherheitskonzept, welches entwickelt wurde, um das Einfügen von Inhalten in Webseiten zu verhindern. CSPs wurden nicht entwickelt, um das Einfügen zu verhindern, sondern viel mehr als „defense-in-depth“ Mechanismus, der den Schaden, der durch den eingefügten Inhalt entsteht, minimieren soll. Dies soll dadurch umgesetzt werden, dass Inhalte nur aus festgelegten Quellen geladen werden dürfen. Der beschriebene Mechanismus scheint das Problem, das in dieser Arbeit behandelt wird, zu lösen. CSPs haben sich bis jetzt noch nicht durchgesetzt, was verschiedene Gründe hat. Der Hauptgrund ist die Tatsache, dass Browser Add-Ons durch CSPs oft unwirksam werden, da diese oft Inhalte in Webseiten einfügen und auch Daten aus anderen Quellen laden (siehe auch [HMS15]).

Danksagung
Diese Arbeit wurde von dem Bundesministerium für Bildung und Forschung (Förderkennzeichen: 13N13252, BOB) unterstützt. Wir möchten ebenfalls Frau Prof. Dr. S. Wetzel (Stevens Institut of Technologie) und Herrn Dr. C. Rossow (Center for IT-Security, Privacy, and Accountability) für die Unterstützung während der Erstellung dieser Arbeit danken. Dank gebührt auch den anonymen Reviewern, die mit wertvollen Anregungen zu dieser Arbeit beigetragen haben. Die Aussagen, Meinung, Empfehlungen und Schlussfolgerung, die innerhalb dieser Arbeit getroffen werden, sind die der Autoren und entsprechen nicht unbedingt der Auffassung der Unterstützer.

Literatur

[CCVK11] D. Canali, M. Cova, G. Vigna, and C. Kruegel. Prophiler: Fast Filter for the Large-Scale Detectionof Malicious Web Pages. In S. Sadagopan, Krithi Ramamritham, Arun Kumar, M. P. Ravindra, Elisa Bertino, and Ravi Kumar, editors, *the 20th international conference*, page 197, 2011.

[CEGU14] V. Chebyshev, D. Emm, M. Garnaeva, and R. Unuchek. IT threat evolution Q1 2014 – Securelist, 2014.

[CKV10] M. Cova, C. Kruegel, and G. Vigna. Detection and analysis of drive-by-download attacks and malicious JavaScript code. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *the 19th international conference*, page 281, 2010.

[CLZS11] C. Curtsinger, B. Livshits, B. Zorn, and C. Seifert. ZOZZLE: Fast and Precise In-browser JavaScript Malware Detection. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, page 3, Berkeley, CA, USA, 2011. USENIX Association.

[CSNT08] C. Reis, S.D. Gribble, N.C. Weaver, and T. Kohno. Detecting in-flight page changes with web tripwires, 2008.

[EGU+15] D. Emm, M. Garnaeva, R. Unuchek, D. Makrushin, and A. Ivanov. IT threat evolution in Q3 2015 – Securelist, 2015.

[HMS15] D. Hausknecht, J. Magazinius, and A. Sabelfeld. May I? Content Security Policy Endorsement for Browser Extensions. In Magnus Almgren, editor, *Detection of intrusions and malware, and vulnerability assessment*, volume 9148 of *Lecture notes in computer science*, pages 261–281. Springer, Cham, 2015.

[MY02] L.M. Manevitz and M. Yousef. One-class Svms for Document Classification. *J. Mach. Learn. Res.*, 2:139–154, 2002.

[SS08] C. Seifert and R. Steenson. Capture-HPC, 2008.

[SWK08] C. Seifert, I. Welch, and P. Komisarczuk. Identification of Malicious Web Pages with Static Heuristics. In *2008 Australasian Telecommunication Networks and Applications Conference (ATNAC)*, pages 91–96, 2008.

[TBG+15] K. Thomas, E. Bursztein, C. Grier, G. Ho, N. Jagpal, A. Kapravelos, D. McCoy, A. Nappa, V. Paxson, P. Pearce, N. Provos, and M. Abu Rajab. Ad Injection at Scale: Assessing Deceptive Advertisement Modifications. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2015.

[Wil88] P. Willett. Recent trends in hierarchic document clustering: A critical review. *Information Processing & Management*, 24(5):577–597, 1988.

[ZKF05] Y. Zhao, G. Karypis, and U. Fayyad. Hierarchical Clustering Algorithms for Document Datasets. *Data Mining and Knowledge Discovery*, 10(2):141–168, 2005.