The right to be forgotten
-
Seminar aus Informatik 2015

Seminararbeit

von

Stefan Vikoler

Betreuung:

Univ.-Prof. Dipl.-Ing. Dr. Wolfgang Pree

Universität Salzburg
Fachbereich Computerwissenschaften
Jakob-Haringer-Strasse 2
A-5020 Salzburg

Austria

Abstract

In dieser Seminararbeit wird das Urteil des Europäischen Gerichtshofes vom 13. Mai 2014 behandelt, welches das Recht eines jeden sichert, Suchergebnisse von Suchmaschinen im Internet unter gewissen Umständen löschen zu lassen - besser bekannt als „The right to be forgotten“. Hierbei wird erläutert, wie es zu diesem Gesetz kam, welche Auswirkungen dieses auf Suchmaschinen im Internet hat und wie sie dazu reagieren. Außerdem wird diskutiert welche Probleme ein „digitales Vergessen“ haben kann und wie man dieses erreichen kann. Des weiteren wird ein Framework vorgestellt, mit welchem ein digitales Vergessen ermöglicht werden kann, indem Daten eine gewisse Lebenszeit bekommen.

Inhaltsverzeichnis

Kapitel 1

Hintergründe

1.1 Einleitung

Bei *„The right to be forgotten“*[11] handelt es sich um einen Präzedenzfall am Europäischen Gerichtshof[1], welcher am 13. Mai 2014 sein Urteil dazu traf. Dabei ging es darum, dass Google[2] Links die Privatsphäre von Einzelnen verletzen können. Der Europäische Gerichtshof erließ daraufhin das Recht auf Löschung von Suchresultaten von Suchmaschinen im Internet und versuchte eine Balance zwischen privaten und öffentlichen Interessen zu wahren.

1.2 Der Präzedenzfall

Der spanischer Anwalt Mario Costeja Gonzáles[10, 4] hatte im Jahr 1998 Sozialversicherungsschulden und musste daher Grundbesitz auf einer Auktion versteigern. Zu dieser Affäre gab es zwei Artikel in der Tageszeitung „La Vanguardia Ediciones“[3][4]. Nachdem Herr Gonzáles seine finanziellen Probleme bereinigte und alle Registereinträge zu diesem Vorfall verfallen sind, klagte dieser im Jahr 2010 die Zeitung und forderte, dass sie die ihm betreffenden Inhalte entfernen, da die Artikel immer noch im Internet lesbar und auffindbar waren.
Die Zeitung weigerte sich dem nachzukommen und berief sich auf Presse- und Meinungsfreiheit von Presse und Zeitungen. Daraufhin klagte Herr Gonzáles

[1] http://europa.eu/about-eu/institutions-bodies/court-justice/index_de.htm
[2] https://www.google.com/
[3] http://hemeroteca.lavanguardia.com/preview/1998/03/09/pagina-13/33837533/pdf.html
[4] http://hemeroteca.lavanguardia.com/preview/1998/03/09/pagina-23/33842001/pdf.html

gegen Google, um die zwei Suchergebnisse zu löschen, welche erschienen, sobald man seinen Namen auf Google suchte. Auch Google weigerte sich die Suchresultate zu löschen, wodurch der Prozess bis zum Europäischen Gerichtshof getragen wurde.

1.3 Urteil

Der Europäische Gerichtshof erließ daraufhin das Urteil[11], dass man ein Recht auf Löschung von Suchergebnissen von Suchmaschinen im Internet hat, wenn die verlinkten Daten

- unangemessen
- irrelevant
- exzessiv
- nicht mehr aktuell
- nicht mehr relevant

sind und nicht in Konflikt mit dem öffentlichen Interesse stehen.
Von dem Urteil sind alle Suchmaschinen im Internet betroffen, da der Europäische Gerichtshof die Suchmaschinen nicht nur als Präsentatoren von Daten im Internet, sondern auch als Data Controller einstufte und diese daher in eine andere Gerichtsbarkeit fallen. Natürlich beruft sich das Urteil nur auf den europäischen Raum.

Kapitel 2

Reaktionen

2.1 Google

Google wehrte sich anfangs mit der Begründung, dass sie keine Daten im Internet kontrollieren sondern lediglich lokalisieren und präsentieren. Außerdem warnten sie vor einer Balkanisierung des Internet, wenn sich Europa eigenständig machen möchte. Der Europäische Gerichtshof sah das anders und das Urteil trat dennoch in Kraft.
Daraufhin stellt Google ein Onlineformular zur Verfügung, womit ein Jeder die Möglichkeit hat, einen Antrag auf Löschung von Google Inhalten und Suchergebnissen zu stellen. Bereits in den ersten fünf Monaten erhielt Google circa 143000 Löschanfragen, das zur Konsequenz hatte, dass Google die Rechtsabteilung aufstocken musste. Denn die Aufgabe zu beurteilen, welche Daten unangemessen, irrelevant, exzessiv, nicht mehr aktuell oder nicht mehr relevant sind und nicht in Konflikt mit dem öffentlichen Interesse stehen, liegt bei den Suchmaschinen.

2.2 Kritik

„The right to be forgotten“ bekam auch viel Kritik, dass es die Redefreiheit untergräbt und eine Zensur des Internet auslöst. Außerdem kann es vorkommen, dass öffentliche Interessen nicht mehr gewahrt werden und eventuell die Möglichkeit besteht, die Geschichte umzuschreiben. Ein weiterer große Kritikpunkt war der Interpretationsspielraum den der Europäische Gerichtshof offen ließ, welche Resultate nun das Recht auf Löschung haben und welcher nicht. Es obliegt den Suchmaschinen Entscheidungen zu treffen, welche nicht einmal der Europäische Gerichtshof fällen konnte oder wollte.
Das Urteil bringt eine große Umstellung für Suchmaschinen mit sich, was

kein Problem für einen Giganten, wie Google darstellt, welcher sich bereits zuvor mit dieser Materie auseinander setzen musste. Es dürfte jedoch schwer für kleine Suchmaschinen umzusetzen sein. Zusätzlich bringt das Gesetz kein wirkliches Vergessen, sondern die zu vergessenden Daten werden lediglich schwerer auffindbar.

2.3 International

International gesehen gibt es *„The right to be forgotten“* noch in Argentinien, welche auch ein Gesetz für die Löschung von Suchresultaten von Suchmaschinen im Internet erließen. Natürlich gab es zu dem Urteil viel globales Aufsehen und weltweit gibt es auch einige Prozesse dazu, welche jedoch bis jetzt immer zu Gunsten des öffentlichen Interesses endeten. Man denkt jedoch, dass es ein Thema für die Zukunft werden kann.
Man muss sich auch die Frage stellen, ob sich Herr Gonzáles der Ironie bewusst war, dass er bei Erfolg des Prozesses auf ewig dafür in Erinnerung bleibt und auch die Informationen, welche er löschen lassen wollte, noch mehr Aufmerksamkeit dadurch erhalten.

Kapitel 3

Digitales Vergessen

3.1 Vergeben und Vergessen

Das Recht auf Vergessen ist keine Erfindung des Europäischen Gerichtshof, sondern viel mehr eine seit Jahrhunderten angewandte Praxis, welche verschieden Hintergründe haben kann. Ein Jeder kennt das kleine Sprichwort *„Vergeben und Vergessen“*. Es gibt schon immer religiöse oder kulturelle Gepflogenheiten, um einem Mitmenschen zu ermöglichen, dass seine Taten vergeben werden können. Und vergebene Taten werden in der Regel auch vergessen.[1]
Falls manche Taten nicht vergeben und somit auch nicht vergessen werden können, gab es früher immer noch die Möglichkeit einen Neustart in einer neuen Umgebung zu machen und so seinen Sünden zu entfliehen. Heutzutage sind Fehltritte meist nur eine Googlesuche entfernt und ein Neustart wird beinahe Unmöglich. Daher sollte man darüber nachdenken, ob nicht auch digitale Sünden vergeben werden können und somit auch vergessen.

3.2 Digitale Identität

Das Internet bietet die Möglichkeit sich neu zu erfinden und mehrere Pseudonyme anzunehmen. Dies wurde anfangs noch gefördert, indem man sich zum Beispiel verschiedene Alias in Foren anlegen konnte. Heutzutage ist es mit den Sozialen Medien nicht mehr so einfach verschiedene digitale Identitäten zu betreiben, da es zusehends zu einer Verschmelzung von privaten und öffentlichen Leben kommt. Außerdem sollte man im Internet immer im Hinterkopf behalten, dass nicht immer alles stimmen muss, was man zu lesen bekommt. Das Internet wird immer mehr als Lügennetz missbraucht.

3.3 Möglichkeiten

In den nächsten Absätzen wird darauf eingegangen welche grundlegenden Möglichkeiten und Methoden es gibt, um Daten verschwinden zu lassen.

3.3.1 Verbreitung kontrollieren

Ein Ansatz ist, dass man die Verbreitung seiner Daten im Internet von Anfang an kontrolliert. Eine Möglichkeit dafür ist der Originator-Controlled Access Control[7] womit man als Ersteller der Daten anderen Zugang gewähren und wieder verbieten kann. Eine weitere Idee ist, dass man nur Links zu seinen Daten verbreitet und nicht die Daten selbst. So kann man die Daten jederzeit wieder vom Netz nehmen. Ein anderer Vorgang bietet an, dass man allen Daten eine gewisse Lebenszeit mitgibt, sodass die Daten und alle Kopien davon nach einer gewissen Zeit nicht mehr zur Verfügung stehen.
Hier stellt sich jedoch die Frage, wer die Daten besitzt. Nur der Ersteller beziehungsweise Besitzer der Daten kann die Verbreitung kontrollieren.

3.3.2 Verstecken

Eine weitere Annäherung seine Daten zu schützen ist, dass man sie durch Irreführen und Überfluten versteckt. Beispielsweise wurde Deutschland im zweiten Weltkrieg getäuscht, indem die Alliierten haufenweise falscher Daten produzierten und lieferten, dass sie über Sardinien einfallen, obwohl das Angriffsziel Sizilien war. Ein weiteres bekanntes Beispiel dafür sind Honeypods, die dazu genutzt werden, einen potenziellen Angreifer von den wirklichen Daten abzulenken und ihm ein anderes Ziel stattdessen anzubieten.
Man kann auch versuchen unerwünschte Daten über sich selbst jemand anderes unterzuschieben. So bleiben die Daten in Takt, jedoch wird das Ziel der Informationen geändert. Diese Herangehensweise ist ethnisch sehr fragwürdig.

Kapitel 4

Making Programs Forget

4.1 Einleitung

Sobald Daten in eine Applikation geladen werden, ist es schwierig Richtlinien festzulegen, wie diese Daten behandelt werden und wie sensible Daten geschützt oder wieder gelöscht werden, ohne dass die Applikation schon im Vorhinein auf diese Richtlinien programmiert wurde. Nur neue Applikationen, die diese Richtlinien einhalten, können eine Lösung für das Problem bieten.
Weiters ist keinem Benutzer garantiert, dass das Leeren der History oder das Löschen einer E-Mail ausreicht, sodass nichts mehr der sensiblen Daten im Speicher oder auf der Festplatte vorhanden ist. Sie können bereits versehentlich in die Zwischenablage kopiert, von einem Programm im Speicher gecached oder von einer E-Mail zu einer anderen gesendet worden sein.
Daher stellt diese Arbeit die *„guaranteed data lifetime“*-Eigenschaft von Kannan et al.[8] vor, womit Daten nur eine minimale Dauer in einem System aufrecht erhalten bleiben, wie sie benötigt werden. Solch eine Eigenschaft ist leicht zu erreichen, wenn Applikationen diese unterstützen. Doch ein Ziel ist auch, dass man nicht auf die Unterstützung von Programmen angewiesen ist. Ein einfacher Weg um dieses Ziel zu erreichen, ist eine Virtual Machine, welche vom Benutzer gestartet wird und solange verwendet wird, wie sensible Daten benötigt oder verarbeitet werden. Jedoch werden dadurch auch andere Daten nach Beendigung entsorgt und eine Veränderung des Benutzerverhaltens ist nötig.

4.1.1 Ziel

Die *„guaranteed data lifetime“*-Eigenschaft stellt drei Bedingungen, die erzielt werden sollen.

Zuerst soll es eine garantierte Lebenszeitgrenze geben, ohne auf andere Applikationen angewiesen zu sein.
Zweitens darf die korrekte Ausführung der Programme nicht beeinflusst werden und die Programme dürfen bei Ende der Lebensdauer von Daten nicht abstürzen.
Drittens soll die Benutzbarkeit für den Benutzer nicht eingeschränkt werden. Der Benutzer soll keine Veränderung im Verhalten der Applikation bemerken.

4.1.2 State of the Art

Um *„guaranteed data lifetime“* zu verwirklichen, gibt es bereits einige Ansätze. Chow et al.[5] schlägt die sofortige Löschung von Daten nach deren Deallozierung vor. Borders et al.[3] empfiehlt alle sensiblen Daten in einer stand-alone Virtaul Machine zu bearbeiten. TightLip[12] verwendet Shadow Prozesse, um Datenverlust von Applikationen zu externen Maschinen zu verhindern. Und Perlman[9] präsentiert ein File System, in dem Dateien mit Richtlinien assoziiert werden können, welche nach einer gewissen Zeit nicht mehr lesbar sein sollen, indem eine Datei mit einem Schlüssel verschlüsselt wird, welcher nur eine gewisse Zeit von einem Master Server verfügbar ist.

4.2 Framework

4.2.1 Status-Wiederherstellung

Um die Lebensdauer von Daten zu garantieren verwendet das Framework Status-Wiederherstellung. Beim Empfang von sensiblen Daten wird kurz vorher ein Snapshot der Applikation als Checkpoint genommen, auf welchen nach Beendigung der Lebensdauer zurückgesetzt wird. Dadurch werden aber auch alle Status und Veränderungen überschrieben, die seit dem Snapshot geschehen sind und für den Benutzer auch noch nach der Lebensdauer wichtig sind. Um das zu verhindern, werden vom Framework alle Inputs nach Empfang der sensiblen Daten mitgeloggt und nach dem zurücksetzen zum Checkpoint wieder abgespielt, sodass die Applikation dem aktuellen Status ohne den sensiblen Daten entspricht.

4.2.2 Drei Phasen

Während der Laufzeit durchläuft das Framework drei Phasen in der Applikations- oder Betriebssystemschicht.

In der ersten Phase muss sensibler Input erkannt werden, welcher über verschiedene Wege wie Tastatur oder Netzwerk an die Applikation geliefert werden kann. Es werden zwei Arten für den Input vorgeschlagen. Die Erste läuft via Tastatur, über die der Benutzer sensiblen Input mit einer speziellen Buchstabensequenz mit den Informationen über die Lebensdauer eingeben kann. Bei der Zweiten kommt der Input von einer externen Datenquelle, wie zum Beispiel über SMTP. So kann zum Beispiel der Sender mittels einer speziellen SMTP Option sensible Daten markieren und deren Lebensdauer mitgeben.
Sobald ein sensibler Input erkannt wird, wird ein Checkpoint der Applikation generiert. Dieser beinhaltet den Applikationszustand, Speicher, Register, Threads und deren Zustände. Ferner werden alle nachfolgenden Inputs zur Applikation, sensibel oder nicht, geloggt.
Die wichtigste Phase beginnt, wenn die Lebensdauer sensibler Daten vorüber ist. Wenn die Applikation mit den sensiblen Daten bereits beendet wurde, wird nichts unternommen und falls nicht, wird der Speicher gelöscht und die Applikation wird von dem System zum Checkpoint zurückgesetzt und alle nachfolgenden, mitgeloggten Inputs werden Schritt für Schritt wiederhergestellt, sodass die Applikation wieder in einem äquivalenten Status weiterläuft. Dieser Ansatz kann auch mit mehreren sensiblen Daten und somit mehreren Checkpoints durchgeführt werden.

4.3 Herausforderungen

Die Status-Wiederherstellung bringt auch einige Probleme mit sich.

4.3.1 Fidelity

Die wichtigste Herausforderung stellt sich beim Applikationszustand nach der Wiederherstellung, dass dieser mit den Erwartungen des Benutzers übereinstimmt und das Fehlen von sensiblen Daten nicht mit der Reihenfolge oder dem Index für andere Daten in Konflikt kommt.
Der erste und einfachste Lösungsvorschlag ist Wiederherstellung mit Wegfall. Hier wird der Input von weiteren sensiblen Daten während der Wiederherstellung untersagt, womit die Sicherheit garantiert werden kann, jedoch die Benutzbarkeit und Verwendbarkeit der Applikation sehr darunter leiden muss.
Ein weiterer Vorschlag ist die Wiederherstellung mit Ersetzung, wo abgelaufene sensible Daten durch nicht-sensible Daten ersetzt werden. Zum Beispiel wird eine sensible E-Mail durch eine E-Mail mit dem Inhalt „Dies ist eine sensible E-Mail, deren Lebensdauer abgelaufen ist.“ ausgetauscht. Hier kann

die Applikation eventuell Probleme mit den Metadaten dieser E-Mail bekommen, da sich Werte wie die Länge der E-Mail auf einmal ändern können.
Die letzte Lösung ist Wiederherstellung mit gleichbleibender Ersetzung, welche dem vorherigen Vorschlag entspricht, jedoch sensible Daten durch gleichwertige Daten ersetzt, sodass die Applikation mit den neuen Daten einfach zurechtkommt und keine Daten vermisst. Diese Methode verspricht die Erfolgversprechendste zu sein.

4.3.2 Pervasiveness

Ein weiteres Problem birgt die mögliche Verbreitung der Daten, denn es kann eventuell vorkommen, dass sensible Daten im System verweilen. Input für Applikationen wird normalerweise über das Betriebssystem gesendet und verarbeitet. Dadurch können Kopien vom Betriebssystem angelegt werden, die später in Phase drei nicht gelöscht werden. Genauso können auch Kopien von Screen-Output und Netzwerk-Output angelegt werden.
Ein Ansatz dieses Problem zu beseitigen ist, dass ein Virtual Machine Monitor (VMM) auf Betriebssystemschicht implementiert wird, sodass das Framework alle Inputs und Outputs des Betriebssystems erfassen kann. Man muss jedoch noch beachten, dass der VMM nicht selbst Daten in den Buffer schreibt.

4.3.3 Containment

Oft kommunizieren Applikationen mit anderen Entitäten auf der selben oder auf externen Maschinen. Also können durch Output sensible Daten verbreitet werden oder Inputs kommen, welcher nur für die originale Ausführung anwendbar sind.
Sendet die Applikation einen Output zu anderen Entitäten, kann in manchen Umständen überprüft werden, ob die andere Maschine ebenfalls über das *„guaranteed data lifetime“* Framework verfügt. Falls das nicht der Fall oder nicht möglich ist, kann gewartet werden bis die Lebensdauer der Daten abgelaufen ist und danach die unsensiblen ersetzten Daten gesendet werden. Der einfachste Weg ist die Entscheidung des Benutzers einzuholen und nur auf Bestätigung sensible Daten zu versenden.
Erhält die Applikation Input von anderen Entitäten steigt die Schwierigkeit mit der Wiedergabegenauigkeit nach der Wiederherstellung, sodass es zu Problemen mit der Benutzbarkeit kommen kann. Man kann versuchen bekannte Requests und Responses von anderen Maschinen abzufangen und entsprechend für das Programm aufarbeiten oder Inputs die Probleme bereiten können werden einfach ignoriert. Dieses Problem muss noch weiter

untersucht werden, um eine zufriedenstellende Lösung zu finden.

4.3.4 Overhead

Zu den vorherigen Problemen stellt sich noch die Frage, ob das System die Performance nachteilig beeinflusst. Dabei stellte sich heraus, dass der Overhead beim Erkennen von sensiblen Daten, ob über Keyboard oder über das System, auch bei großen Mengen klein ist. Auch das Erstellen der Checkpoints ist eine einmalige Sache und birgt nur wenig Schwierigkeiten. Ein Problem kann es beim Loggen der Inputs geben, da dies zu jeder Zeit geschieht bis die Lebensdauer der sensiblen Daten abgelaufen ist. Jedoch kann auch diese Herausforderung von heutigen multi-core System bewältigt werden. Zuletzt muss vom System noch die Wiederherstellung verarbeitet werden, welche jedoch parallel zur Applikation geschehen kann und somit keine Auswirkung auf die Performance bringt.

4.4 Fazit

Kannan et al.[8] stellen ein solides Konzept zur Verfügung, welches sensible Daten auf Dauer mit einer Lebenszeit versehen kann ohne dass diese eventuell in falsche Hände geraten können und Schaden verursachen. Alles mit dem Vorteil, dass man von Applikationen unabhängig bleibt und deren korrekte Ausführung beeinflusst oder die Benutzbarkeit für den Benutzer verschlechtert. Gewisse Herausforderungen müssen noch überarbeitet werden, doch grundsätzlich sind Lösungen dafür vorhanden.

Literaturverzeichnis

[1] Matt Bishop, Emily Rine Butler, Kevin Butler, Carrie Gates, and Steven Greenspan. Forgive and forget: Return to obscurity. In *Proceedings of the 2013 Workshop on New Security Paradigms Workshop*, NSPW '13, pages 1–10, New York, NY, USA, 2013. ACM.

[2] Cheng Bo, Guobin Shen, Jie Liu, Xiang-Yang Li, YongGuang Zhang, and Feng Zhao. Privacy.tag: Privacy concern expressed and respected. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, SenSys '14, pages 163–176, New York, NY, USA, 2014. ACM.

[3] Kevin Borders, Eric Vander Weele, Billy Lau, and Atul Prakash. Protecting confidential data on personal computers with storage capsules. In *Proceedings of the 18th Conference on USENIX Security Symposium*, SSYM'09, pages 367–382, Berkeley, CA, USA, 2009. USENIX Association.

[4] Lee A. Bygrave. A right to be forgotten? *Commun. ACM*, 58(1):35–37, December 2014.

[5] Jim Chow, Ben Pfaff, Tal Garfinkel, and Mendel Rosenblum. Shredding your garbage: Reducing data lifetime through secure deallocation. In *Proceedings of the 14th Conference on USENIX Security Symposium - Volume 14*, SSYM'05, pages 22–22, Berkeley, CA, USA, 2005. USENIX Association.

[6] David Crowe and Wasim A. Al-Hamdani. Google privacy: Something for nothing? In *Proceedings of the 2013 on InfoSecCD '13: Information Security Curriculum Development Conference*, InfoSecCD '13, pages 27:27–27:32, New York, NY, USA, 2013. ACM.

[7] R. Graubert. On the need for a third form of access control. In *Proceedings of the 12th National Computer Security Conference*, pages 296–304, 1989.

[8] Jayanthkumar Kannan, Gautam Altekar, Petros Maniatis, and Byung-Gon Chun. Making programs forget: Enforcing lifetime for sensitive data. In *Proceedings of the 13th USENIX Conference on Hot Topics in Operating Systems*, HotOS'13, pages 23–23, Berkeley, CA, USA, 2011. USENIX Association.

[9] Radia Perlman. File system design with assured delete. In *Proceedings of the Third IEEE International Security in Storage Workshop*, SISW '05, pages 83–88, Washington, DC, USA, 2005. IEEE Computer Society.

[10] Mike Wagner and Yun Li-Reilly. The right to be forgotten. *THE ADVOCATE*, 72(6):823–833, November 2014.

[11] Philip Ward. The „right to be forgotten". Library, House of Commons, September 2014. SN/HA/6983.

[12] Aydan R. Yumerefendi, Benjamin Mickle, and Landon P. Cox. Tightlip: Keeping applications from spilling the beans. In *Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation*, NSDI'07, pages 12–12, Berkeley, CA, USA, 2007. USENIX Association.