Sicherheitsanalyse der Private Cloud Interfaces von openQRM

Frederic Schulz · Dennis Felsch · Jörg Schwenk

Horst Görtz Institut für IT-Sicherheit
Ruhr-Universität Bochum
{frederic.schulz | dennis.felsch | joerg.schwenk}@ruhr-uni-bochum.de

Zusammenfassung

Eine zentrale Eigenschaft einer Infrastructure as a Service (IaaS) Cloud ist das Vorhandensein von Interfaces, mit denen ein Nutzer seine virtuellen Maschinen verwalten kann [MeGr11]. Häufig werden dazu interaktive HTML5 Web-Interfaces eingesetzt. Doch mit steigendem Grad an Interaktivität steigt auch die Gefahr durch Sicherheitslücken. Dieses Paper zeigt, wie anfällig eine Cloud-Infrastruktur wird, wenn das genutzte Web-Interface Schwachstellen aufweist. Anhand der Sicherheits-Analyse der Interfaces der Software openQRM, einer in Deutschland entwickelten Cloud-Plattform, wird gezeigt, dass web-basierte Angriffe bis in eine virtuelle Maschine hinein Auswirkungen haben können. Dazu muss ein Angreifer nicht einmal direkten Zugriff auf ein Cloud-Interface haben. Für einen erfolgreichen Angriff ist es ausreichend, dass ein unvorsichtiger Mitarbeiter einen bösartigen Link aufruft. Alle gefundenen Schwachstellen wurden den Entwicklern von openQRM gemeldet.

1 Einleitung

Private Infrastructure as a Service (IaaS) Clouds werden oft mit dem Versprechen beworben, sicherer als Public Clouds zu sein. Begründet wird dies mit der vollen Kontrolle über die Infrastruktur sowie dem Schutz der Cloud-Interfaces durch Netzwerk-Separation und Firewalls. Doch der Betrieb einer eigenen, privaten Cloud-Infrastruktur sorgt nicht zwangsläufig für einen Sicherheits-Gewinn. Der Einsatz moderner HTML5-Web-Oberflächen, die heute von allen etablierten Cloud Management Plattformen mitgeliefert werden, kann dazu führen, dass ein Angreifer selbst bei gut konfigurierter Sicherheits-Infrastruktur die volle Kontrolle über eine Cloud-Plattform übernehmen kann.

1.1 Cloud-Modelle

Die NIST Special Publication 800-145 [MeGr11] kategorisiert Cloud Computing in drei verschiedene Service-Modelle: Software as a Service (SaaS), Platform as a Service (PaaS) und Infrastructure as a Service (IaaS). Wir betrachten im Folgenden ausschließlich IaaS Cloud Systeme, bei denen der Kunde eine virtuelle Maschine (VM) bekommt. Für diese kann der Kunde aus einer Reihe von vorbereiteten Hardware-, Netzwerk-, und Betriebssystem-Konfigurationen wählen.

Die NIST definiert in [MeGr11] auch Einsatzmodelle, die in aller Regel für IaaS-Clouds verwendet werden: *Public Clouds*, die von jedem Kunden gebucht werden können, *Private Clouds*, die nur von einem Kunden (ein Unternehmen oder eine Organisation) genutzt werden und zwei

P. Schartner · K. Lemke-Rust · M. Ullmann (Hrsg.) · D•A•CH Security 2015 · syssec (2015) 1-11.

Zwischenmodelle, die *Community* und *Hybrid Clouds*. Bei einer *Community Cloud* ist die Gruppe der Kunden nicht so stark eingeschränkt wie bei der Private Cloud, sie steht aber auch nicht jedem potentiellen Kunden offen. Eine *Hybrid Cloud* kombiniert mehrere Cloud-Systeme und -Einsatzmodelle, um z.B. einen Lastausgleich zwischen den Clouds zu realisieren. Im Folgenden gehen wir von einer Private Cloud als Einsatzmodell aus.

1.2 Angriffe auf Web-Oberflächen

Web-Browser sind heute der gängigste Zugang zu einer Cloud Management Plattform. Sie sind auf jedem Betriebssystem zu finden und die Technologien im Hintergrund (HTML5, XML, JSON, JavaScript, CSS, etc.) sind plattformunabhängig. Darüber hinaus werden ihre Kommunikationsprotokolle (HTTP, HTTPS) in praktisch jeder Firewall erlaubt.

Der stetig wachsende Funktionsumfang der Browser bietet jedoch auch eine stetig wachsende Angriffsfläche. Durch bösartiges JavaScript (XSS) sowie automatisches Nachladen von Ressourcen (CSRF) können Angriffe unbemerkt vom Opfer durchgeführt werden. Indem ein Nutzer den gleichen Browser für das Surfen im Web und die Bedienung sicherheitskritischer Anwendungen verwendet, können Angriffe durch Firewalls hindurch geschehen.

1.3 Angriffe auf Clouds

Da Private Clouds nur von einem Unternehmen oder einer Organisation genutzt werden, sind Angriffe auf der Hypervisor- oder Virtualisierungs-Ebene nur durch einen internen Angreifer möglich, der auch in der Lage ist, eine virtuelle Maschine anzulegen. Daher konzentrieren wir uns auf die Verwaltungsebene der Cloud, die im Allgemeinen über unterschiedlichste APIs angesteuert werden kann.

Die grundlegende Idee unserer Angriffe ist es, den Browser des Opfers zu einem Werkzeug des Angreifers innerhalb des Unternehmensnetzes zu machen. Dies geschieht, indem das Opfer auf einen Link klickt, der vom Angreifer kommt. Über die offenen Ports in der Firewall kann so der Browser die Instruktionen empfangen oder auch Daten aus dem Unternehmensnetz zum Angreifer senden.

2 Related Work

Viele bisherige Sicherheitsanalysen von Infrastructure as a Service Clouds beschäftigen sich mit Co-Location [RTSS09, VKFR+12, AIES14, SIYA11, ZJRR12]. Dabei wird versucht zu erkennen, ob zwei virtuelle Maschinen auf einem physischen Host ausgeführt werden. In einem weiteren Schritt wird dann versucht, Seitenkanäle zwischen den VMs zu finden und diese auszunutzen.

Eine Sicherheitsanalyse über alle Ebenen des Cloud Computing einschließlich Web-Angriffen hinweg gibt [MPBP+13]. Zwei Sicherheitsanalysen mit Fokus auf Betriebssystemsicherheit der openQRM-Konkurrenzsysteme OpenStack und Eucalyptus werden in [RiGD13] und [GuRD13] durchgeführt.

Untersuchungen zu dem SOAP-basierten Interface der Public Cloud *Amazon Elastic Compute Cloud (EC2)* wurden von [GrIa09] durchgeführt. Diese Arbeiten wurden von [SHJS+11] erweitert. Dort wurden die Interfaces von Amazon und Eucalyptus über Cross-Site-Scripting and XML-Angriffe gebrochen.

Einen Überblick über Cross-Site-Scripting (XSS) geben [John11, Heid12]. Die Konzepte und Varianten von XSS erläutert auch Abschnitt 4.1. Forschung zu neuen Arten von XSS wurden in [HNSH+12] und [HSFM+13] durchgeführt, bei der *Scriptless Attacks* und *mXSS* entdeckt wurden. Bei *Scriptless Attacks* können Angriffe ohne die Ausführung von Skripten durchgeführt werden. *mXSS* erlaubt es, eigentlich harmlose Inhalte durch die Fehlertoleranz-Funktionalität der Browser in Schadcode zu transformieren. Andere web-basierte Angriffstechniken wie Cross-Site Request Forgery and UI-Redressing können in [ZeFe08] und [NiSc12] nachgelesen werden.

3 Modell

Für eine Sicherheitsanalyse einer Private Cloud ist es notwendig, aus technischer Sicht zu definieren, wie die Cloud aufgebaut und an welchen Stellen Schutzmechanismen eingesetzt werden. Des Weiteren müssen die Möglichkeiten des Angreifers definiert werden.

3.1 Modell einer Private IaaS Cloud

Der wesentliche technische Unterschied zwischen einer Private Cloud und einer Public Cloud ist die Erreichbarkeit der Cloud-Interfaces aus dem Internet heraus. Eine Public Cloud muss aus dem Internet erreichbar sein, damit Kunden diese nutzen können. Bei einer Private Cloud dagegen ist dies nicht notwendig; in der Praxis wird es durch technische Maßnahmen (z.B. Firewalls, NAT, VLANs, etc.) unterbunden (siehe Abbildung 1). Ein legitimer Zugriff auf die Cloud-Interfaces aus dem Internet ist dann nur unter speziellen Bedingungen möglich, z.B. wenn der Client sich per VPN einwählt.

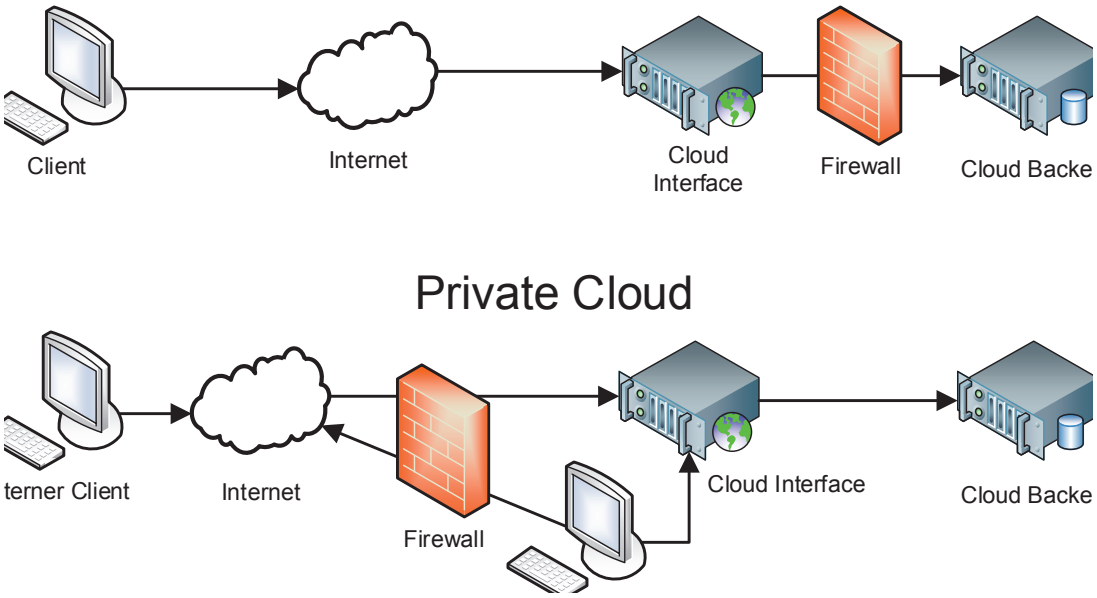

Abb. 1: In einer Public Cloud (oben) ist das Cloud Interface notwendigerweise aus dem Internet erreichbar. In einer Private Cloud (unten) haben nur Clients innerhalb des Netzwerkperimeters uneingeschränkten Zugang zum Cloud Interface, können jedoch auch ins Internet kommunizieren.

Auf die Erreichbarkeit der virtuellen Maschinen, die in der Cloud betrieben werden, aus dem Internet hat die Unterscheidung zwischen Private Cloud und Public Cloud dagegen keinen direk-

ten Einfluss. Beide Einsatzmodelle erlauben es, die Netzwerkanbindung der VMs unabhängig von der Erreichbarkeit der Cloud-Interfaces zu konfigurieren.

3.2 Angreifermodell

Für unsere Analyse werden zwei Arten von Angreifern betrachtet:

1. Der **interne Angreifer** hat einen gültigen Zugang zum Cloud-System mit Benutzerrechten. In einer Public Cloud entspricht dies einem normalen Nutzer, der sich selbst bei der Cloud registriert hat. Im Szenario einer Private Cloud entspricht dieser Angreifer einem Mitarbeiter, der seinem Arbeitgeber schaden möchte.
2. Der **externe Angreifer** hat keinen Zugang zum Cloud-System, das er angreifen will. Falls die Cloud ein Open Source System einsetzt, kann der Angreifer sich jedoch ein eigenes Setup aufsetzen. Es wird angenommen, dass der externe Angreifer den internen DNS-Namen der Private Cloud kennt[1]. Der Angreifer muss diesen DNS-Namen nicht notwendigerweise auflösen können. Darüber hinaus kann der externe Angreifer Mitarbeitern des Unternehmens Nachrichten schicken. Dies können z.B. Emails mit Links sein, die ein Mitarbeiter unbedacht öffnet.

Es ist offensichtlich, dass der interne Angreifer mächtiger als der externe Angreifer ist. Einen speziellen Fokus bilden in der Analyse daher Sicherheitslücken, die auch ein externer Angreifer ausnutzen kann, selbst wenn die Cloud hinter dem Netzwerk-Perimeter eines Unternehmens liegt.

4 Angriffe auf Web-Interfaces

Im Folgenden werden drei Klassen von Angriffen auf Web-Anwendungen beschrieben, die sich unter den zehn häufigsten Sicherheitsrisiken für Web-Anwendungen [owa13] befinden: Cross-Site-Scripting (XSS), Cross-Site-Request-Forgery (CSRF) und Code-Injection (SQL-Injection).

4.1 Cross-Site-Scripting (XSS)

Für einen Cross-Site-Scripting (XSS) Angriff versucht der Angreifer seinen schädlichen JavaScript-Code in die Webseite, die von einer Webanwendung ausgeliefert wird, zu injezieren. Erreicht der Angreifer dieses Ziel, kann er damit die *Same Origin Policy (SOP)* umgehen, die sonst Session Cookies, Formularfelder, etc. vor dem Zugriff des Angreifers schützt.

Es existieren drei Varianten von XSS:

1. Bei **reflektiertem XSS** bringt der Angreifer den Browser des Opfers dazu, den Schadcode zuerst an den verwundbaren Server zu senden (z.B. als GET-Parameter oder als Body eines POST-Requests). Der Server integriert den Schadcode dann in eine dynamisch generierte Seite, die er dem Client zurück schickt, wo dieser dann ausgeführt wird.
2. Bei **persistentem XSS** speichert der Angreifer den Schadcode persistent in der verwundbaren Web-Anwendung (z.B. einem Diskussions-Forum). Ruft das Opfer die präparierte Unterseite der Web-Anwendung ab, so wird der Schadcode mit der Seite ausgeliefert und beim Opfer ausgeführt.

[1] Interne DNS-Namen von Cloud-Installationen findet man z.B. häufig in Support-Foren.

3. Bei **DOMXSS** nutzt der Angreifer aus, dass moderne Browser die aktuell dargestellte Seite dynamisch über Skripte anpassen können. Dabei werden z.B. Informationen aus der URL in die Seite integriert. Im Gegensatz zu den anderen zwei Varianten von XSS wird der Schadcode bei DOMXSS nicht notwendigerweise an den Server gesendet und kann daher dort nicht abgewehrt werden.

Die wichtigste Gegenmaßnahme gegen alle Varianten von XSS ist serverseitiges Filtern von Skriptcode. Einige clientseitige Filter (z.B. *NoScript* für den *Mozilla Firefox* [nos]) können das Schutzniveau weiter erhöhen.

4.2 Cross-Site Request Forgery (CSRF)

Einige Aktionen eines Web-Browsers können ohne eine Interaktion des Nutzers ausgelöst werden. So sorgt das Markup `<img src="http://example.com/pic.jpg"/>` in einer Webseite dafür, dass der Browser versucht, `pic.jpg` von `example.com` mittels eines HTTP GET-Requests zu laden.

Ist der Angreifer in der Lage, das Opfer eine Seite öffnen zu lassen, die das Markup `<img src="http://pizza.org/order?type=13&deliver=attacker"/>` enthält, während das Opfer bei `pizza.org` angemeldet ist, so reicht das Betrachten der Seite des Angreifers dazu aus, dass dem Angreifer eine Pizza geliefert wird, die dem Opfer in Rechnung gestellt wird.

CSRF Angriffe können abgewehrt werden, indem in alle Formulare einer Seite unsichtbar für den Nutzer zufällige Werte (sog. Anti-CSRF-Tokens) eingefügt werden. Der Server verarbeitet dann nur Anfragen, die den korrekten Token enthalten. Da der Angreifer diesen Token nicht erraten kann, schlägt der Angriff fehl.

4.3 SQL-Injection (SQLi)

Bei einem SQL-Injection (SQLi) Angriff schleust der Angreifer Daten ein, die dann von der Datenbank als Code interpretiert werden. Wird z.B. die Anfrage an die Datenbank mit dem Befehl `query = "SELECT * FROM users WHERE id="+request.get("id")` erzeugt und der Angreifer ruft im Folgenden die verwundbare Web-Anwendung mit der URL `"http://example.com/users?id=1 or 1=1"` ab, so antwortet die Datenbank nicht nur mit den Daten eines Nutzers, sondern mit allen Datensätzen der Tabelle `users`.

Auf diese Art und Weise kann der Angreifer nicht nur Daten auslesen, sondern häufig auch verändern oder löschen. SQLi-Angriffe können verhindert werden, indem konsequent zwischen Daten und Code für die Datenbank unterschieden wird. Die wichtigste Technik dies zu gewährleisten sind *Prepared Statements*, die für viele Programmiersprachen zur Verfügung stehen.

5 Analyse von openQRM

Gegenstand der Untersuchung, die für dieses Paper durchgeführt wurde, ist die Private Cloud Software von *openQRM Enterprise GmbH* [ope] mit Sitz in Bonn. Sie vertreibt eine Infrastructure as a Service (IaaS) Cloud als kostenfrei verfügbare *Community Edition* und alternativ mit Support und weiteren technischen Features versehene *Enterprise Edition*. Die Software basiert auf der Skriptsprache PHP. Seit 2013 zählt neben der Fachhochschule Technikum Wien, der

Firma Aragon Software Internet-Service, der Kunsthochschule Utrecht und weiteren internationalen Unternehmen auch die DB Systel, eine Tochter der Deutschen Bahn AG, zu den Kunden von openQRM Enterprise GmbH.

Für dieses Paper wurden die Interfaces der aktuellen *Community Edition* Version 5.1 sowie *Enterprise Edition* Version 5.2 von openQRM auf die vorgestellten Schwachstellen und Angriffe hin manuell untersucht. Dabei wurden die vom Hersteller vorgegebenen Standard-Einstellungen beibehalten. Weiterhin wurde die Software nach Interface-übergreifenden Lücken zwischen den zwei Web-Oberflächen für Administratoren und normale Benutzer sowie anderen Cloud-Interfaces, wie dem intergrierten SOAP Webservice, untersucht. Alle gefundenen Schwachstellen wurden den Entwicklern gemeldet.

5.1 Allgemeine Sicherheit

Als Authentifizierungsmethode verwendet openQRM HTTP Basic Authentication [RFC], bei der die Anmeldeparameter im Klartext als Header versandt werden. Aus diesem Grund sollten die Web-Oberflächen von openQRM nicht ohne TLS genutzt werden. Eine diesbezügliche Konfigurations-Empfehlung seitens der Entwickler wird jedoch nicht gegeben. Lediglich die online verfügbare Konfigurationsanleitung für die Enterprise Cloud Zones gibt an, dass aufgrund der Nutzung des SOAP Webservices für diese Oberfläche HTTPS benötigt werde.

Die Benutzer- und Administrator-Passwörter werden im Klartext in einer Datenbank gespeichert obwohl darauf schon 2009 seitens der Internet-Community bereits als Sicherheitsmangel hingewiesen wurde [Kla]. HTTP-Header zur Vorbeugung von Web-Angriffen wie `X-Frame-Options` oder `Content-Security-Policy` werden nicht eingesetzt.

Alle Web-Oberflächen nutzen die JavaScript-Bibliotheken JQuery 1.3.2 und JQueryUI 1.7.1. Diese Versionen sind nicht mehr aktuell und anfällig gegenüber XSS [cve13, cve14]. Die anfälligen Funktionen sind jedoch in den untersuchten Versionen von openQRM nicht in Gebrauch.

Fortlaufende Nummerierungen bei IDs und fehlende Zugriffskontrollen in der Benutzer-Oberfläche ermöglichen es dem Nicht-Admin, Informationen über VMs, Profile und private Festplatten-Images anderer Benutzer zu erlangen. Dies wurde in Version 5.2 erschwert, indem die fortlaufenden IDs durch Pseudozufallszahlen, die auf der PHP Microtime basieren, ersetzt wurden. Die Lücken in der Benutzeroberfläche, die die Gefahr des Datendiebstahls bergen, wurden hingegen nicht vollständig behoben. Darüber hinaus setzt nur der Zugriff auf VM-Beschreibungen anderer Benutzer eine entsprechende Authentifizierung voraus.

5.2 XSS in openQRM

Die beiden Web-Oberflächen für Administratoren und Benutzer von openQRM setzen zwei unterschiedliche Filter zur Abwehr von XSS ein. Sie sollen das Einschleusen von schädlichem JavaScript-Code verhindern. Die Filter sind Eigenentwicklungen und basieren auf regulären Ausdrücken, die schädliche Zeichenfolgen entfernen. Jedoch sind diese Filter unvollständig und lassen sich leicht umgehen. Beispielsweise löscht einer der Filter den Text `<script>` aus der Eingabe. Da der Filter nur einmalig eingesetzt wird, lässt sich der Filter umgehen, indem der Text `<sc<script>ript>` als Eingabe eingesetzt wird. Der zweite Filter selektiert dagegen `javascript` und kann mit Hilfe von `<javascriptscript>` umgangen werden.

Insbesondere in openQRM 5.1 gibt es in der Benutzer-Oberfläche gänzlich ungefilterte Einga-

befelder. In Version 5.2 werden alle Eingabefelder mit dem Filter, der in der Vorgängerversion in der Administrator-Oberfläche eingesetzt wurde, gefiltert.

Ein interner Angreifer kann die XSS-Schwachstellen der Web-Oberfläche ausnutzen, um Schadcode in den nur für Administratoren nutzbaren Bereich zu injizieren. Indem der Angreifer die Account-Details oder die Beschreibungstexte der virtuellen Maschinen mit Angriffsvektoren speist, werden diese im Browser des Administrators ausgeführt, sobald dieser die entsprechend manipulierte Seite abruft. Hierbei kann sich der Angreifer unbemerkt einen Administrator-Account erstellen lassen und erhält damit die volle Kontrolle über die Cloud-Installation.

Eine Schwachstelle, die wir gefunden haben, betrifft die Administrator-Oberfläche von openQRM 5.1. Diese zeigt Nachrichten-Boxen an, die über erfolgreich durchgeführte Aktionen informieren. Der Inhalt dieser Nachrichten ist über HTTP GET-Parameter steuerbar. Mit einem speziell gestalteten Link kann so ein reflektiertes XSS ausgelöst werden (siehe Abbildung 2). Da diese Schwachstelle auch ausgenutzt werden kann, ohne direkten Zugriff auf das Web-Interface zu haben, kann sie auch von einem externen Angreifer verwendet werden.

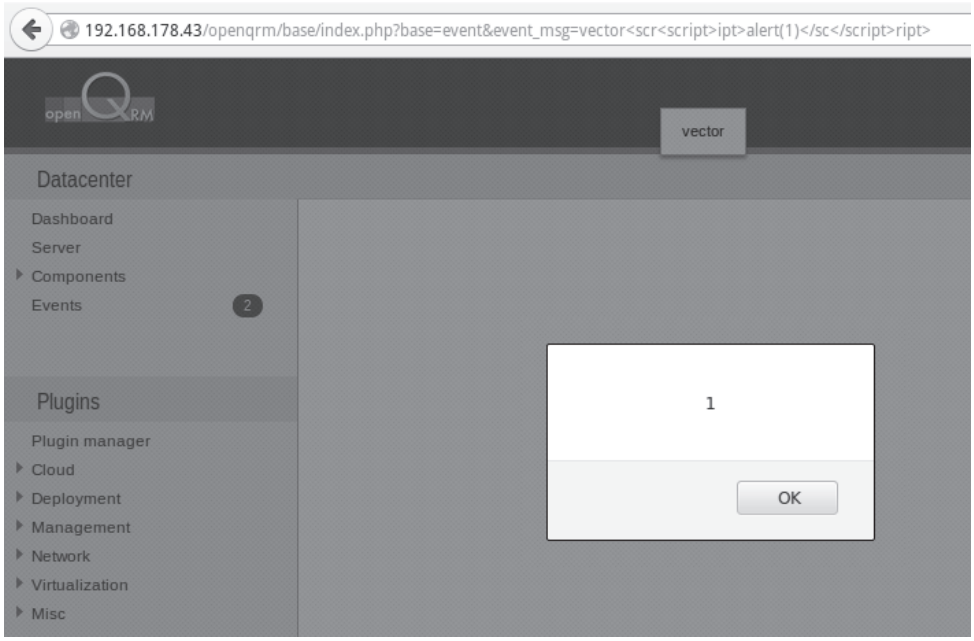

Abb. 2: Ein speziell gestalteter Link auf die Administrator-Oberfläche von openQRM 5.1 löst eine XSS-Schwachstelle aus.

5.3 CSRF in openQRM

openQRM trifft keine Sicherheitsvorkehrungen gegen CSRF. Damit steht sowohl dem internen als auch dem externen Angreifer die Möglichkeit offen, Einstellungen und Daten zu verändern oder neue Accounts zu erstellen. Einzige Voraussetzung für Angriffe ist, dass ein Benutzer oder Administrator, der gerade bei der Cloud angemeldet ist, auf einen bösartigen Link klickt. Auf diese Art und Weise kann auch der externe Angreifer persistenten XSS-Schadcode hinterlegen. Insbesondere in Kombination mit den XSS-Schwachstellen sind den Angreifern hier praktisch keine Grenzen gesetzt.

5.4 SQLi in openQRM

SQL-Injection-Angriffe sind in verschiedenen Eingabefeldern sowohl in der Admin-Oberfläche als auch der Nutzer-Oberfläche möglich. Mit diesen kann der interne Angreifer Daten und Klartext-Passwörter sowohl von Administratoren als auch von anderen Benutzern auslesen und modifizieren. Mithilfe geeigneter Datenbank-Manipulationen kann dieser Angreifer z.B. den angebotenen Service entgegen der Konfiguration nutzen, ohne dass ihm seine Ressourcen-Nutzung in Rechnung gestellt wird. Andere Manipulationen können gezielt einzelne Funktionen stören oder die Cloud vollständig stilllegen.

Der externe Angreifer kann zwar keine Daten direkt aus der Datenbank lesen, allerdings ist auch er in der Lage, in Kombination mit XSS und CSRF schädliche Änderungen an der Cloud-Datenbank vorzunehmen. Durch die Manipulation von Email-Adressen von Benutzern können beide Angreifer so an interne Daten, wie z.B. das Root-Passwort und die IP einer neu aufgesetzten VM, gelangen und diese, sofern sie direkt im weltweiten Netz erreichbar ist, übernehmen.

5.5 Interface-übergreifende Angriffe über SOAP

Der SOAP Webservice von openQRM, den wir in Version 5.2 testen konnten, bietet im Wesentlichen die gleiche Funktionalität wie die Web-Oberflächen. Jedoch wird auch dort nicht konsequent gefiltert. Insbesondere können über einen CDATA-Abschnitt beliebige Zeichenfolgen genutzt werden. So kann der Angreifer mithilfe des Webservices persistenten XSS-Schadcode hinterlegen, der in dieser Form wegen eines Filters über das Web-Interface nicht hätte hinterlegt werden können. Ruft nun ein Administrator die Web-Oberfläche ab, so wird erneut Schadcode mit Administrator-Rechten ausgeführt.

Da wir davon ausgehen, dass der SOAP Webservice einer openQRM Private Cloud nicht aus dem Internet heraus erreichbar ist, ist diese Schwachstelle nur für einen internen Angreifer interessant.

6 Gegenmaßnahmen

Um die vorgestellten Schwachstellen und die daraus resultierenden Angriffe zu erschweren, schlagen wir folgende Gegenmaßnahmen vor:

6.1 Gegenmaßnahmen durch den Nutzer

Nutzer können sich vor CSRF schützen, indem sie separate Browser für die Nutzung der Cloud-Interfaces und für das Surfen im Internet verwenden. Diese Gegenmaßnahme schränkt jedoch die Benutzerfreundlichkeit ein.

Ein separater Browser schützt jedoch nicht vor XSS. Daher empfehlen wir den Einsatz einer clientseitigen Filter-Software wie das bereits erwähnte *NoScript* für den *Mozilla Firefox* [nos]. Leider ist auch die korrekte Konfiguration eines solchen Filters für unerfahrene Nutzer schwierig.

6.2 Gegenmaßnahmen durch den Administrator

Administratoren müssen bedenken, dass der Betrieb einer Private Cloud innerhalb des Netzwerk-Perimeters des eigenen Unternehmens nicht alle Sicherheitsprobleme löst. Die Software sollte immer auf dem letzten stabilen Release betrieben werden, um von Sicherheits-Fixes der Ent-

wickler zu profitieren und Risiken zu minimieren. Außerdem sollten Administratoren die aktivierten Interfaces im Blick halten und nicht genutzte Interfaces deaktivieren.

Werden Web-Interfaces genutzt, so sollten Administratoren prüfen, welche HTTP-Header mit Sicherheitsbezug eingesetzt werden und diese gegebenenfalls über einen serverseitigen Proxy erweitern. Die Gültigkeit von Sessions sollte verkürzt werden, um die Wahrscheinlichkeit, dass ein Benutzer angegriffen wird, der vergessen hat sich abzumelden, zu verringern.

6.3 Gegenmaßnahmen durch den Entwickler

Entwickler von Cloud Systemen müssen Web-Angriffe im Hinterkopf haben, wenn sie ein Web-Interface entwickeln, auch wenn dieses Interface gar nicht für die Bereitstellung im Internet gedacht ist. Wie unsere Analyse zeigt, ist es durchaus möglich, den Browser des Opfers hinter dem Netzwerk-Perimeters für einen Angriff zu instrumentalisieren. Spezifische Gegenmaßnahmen gegen die vorgestellten Schwachstellen wurden bereits in Kapitel 4 erwähnt.

Das Rechte-Management einer Cloud sollte berücksichtigen, dass es immer einen internen Angreifer geben kann. Da das Web-Interface einer Cloud ein zentraler Punkt zur Steuerung für Nutzer und Administratoren ist, sind Rechteausweitungen sehr mächtig.

Entwickler sollten die Aufgabe, die korrekten HTTP Header zu setzen, nicht den Administratoren überlassen, da einige Header nur eingesetzt werden können, wenn der Code des Web-Interfaces dies vorsieht. Insbesondere der `Content-Security-Policy` Header erlaubt es, feingranular einzustellen, welchen Code eine Webseite ausführen darf. Dazu müssen allerdings die legitimen Skripte der Webseite gewisse Bedingungen erfüllen.

7 Schlussfolgerung und Ausblick

Im Zusammenhang mit Infrastructure as a Service Clouds zeigen die präsentierten Angriffe, dass Schwachstellen in Web-Interfaces nicht vernachlässigt werden dürfen. Angriffe auf diese Schwachstellen können bis auf die Ebene der virtuellen Maschinen gelangen und es dem Angreifer erlauben, Kontrolle über die VMs zu erlangen. Die Tatsache, dass der Angreifer dazu noch nicht einmal direkten Zugriff auf die verwundbare Oberfläche haben muss, zeigt, dass die pauschale Annahme, Private Clouds seien sicherer als Public Clouds, nicht in jedem Fall so gerechtfertigt ist.

Weitere Forschungsarbeiten müssen zeigen, ob openQRM eine Ausnahme darstellt oder ob die Interfaces anderer Cloud-Systeme von diesen Problemen auch betroffen sind. Darüber hinaus sollte untersucht werden, ob es Möglichkeiten gibt, sich über Web-Angriffe noch tieferen Zugang in die virtuellen Maschinen in einer Cloud zu verschaffen und so z.B. VMs anzugreifen, die nicht aus dem Internet erreichbar sind.

Literatur

[AIES14] G. I. Apecechea, M. S. Inci, T. Eisenbarth, B. Sunar: Fine grain Cross-VM Attacks on Xen and VMware are possible! In: *IACR Cryptology ePrint Archive*, 2014 (2014), 248.

[cve13] CVE-2011-4969 (2013), [online] `http://www.cvedetails.com/cve/CVE-2011-4969/`.

[cve14] CVE-2012-6662 (2014), [online] http://www.cvedetails.com/cve/CVE-2012-6662/.

[GrIa09] N. Gruschka, L. L. Iacono: Vulnerable cloud: Soap message security validation revisited. *In: Web Services, 2009. ICWS 2009. IEEE International Conference on*, IEEE (2009), 625–631.

[GuRD13] M. Gusev, S. Ristov, A. Donevski: Security vulnerabilities from inside and outside the Eucalyptus cloud. *In: Proceedings of the 6th Balkan Conference in Informatics*, ACM (2013), 95–101.

[Heid12] M. Heiderich: Towards elimination of xss attacks with a trusted and capability controlled dom. Dissertation, Ruhr-Universität Bochum (2012).

[HNSH+12] M. Heiderich, M. Niemietz, F. Schuster, T. Holz, J. Schwenk: Scriptless Attacks–Stealing the Pie Without Touching the Sill. *In: ACM Conference on Computer and Communications Security (CCS)* (2012).

[HSFM+13] M. Heiderich, J. Schwenk, T. Frosch, J. Magazinius, E. Z. Yang: mXSS Attacks: Attacking well-secured Web-Applications by using innerHTML Mutations. *In: ACM Conference on Computer and Communications Security (CCS)* (2013).

[John11] M. Johns: Code Injection Vulnerabilities in Web Applications-Exemplified at Cross-site Scripting. Dissertation, Universität Passau (2011).

[Kla] Cloud plugin: Hide clear-text passwords. [online] http://sourceforge.net/p/openqrm/discussion/529932/thread/e8240651/#756a.

[MeGr11] P. Mell, T. Grance: The NIST definition of cloud computing. In: *NIST Special Publication 800-145* (2011).

[MPBP+13] C. Modi, D. Patel, B. Borisaniya, A. Patel, M. Rajarajan: A survey on security issues and solutions at different layers of Cloud computing. In: *The Journal of Supercomputing*, 63, 2 (2013), 561–592.

[NiSc12] M. Niemietz, J. Schwenk: UI Redressing Attacks on Android Devices. In: *Black Hat Abu Dhabi* (2012).

[nos] NoScript Home Page. , [online] http://noscript.net/.

[ope] OpenQRM Homepage. [online] http://www.openqrm.com/.

[owa13] The OWASP Foundation: The Ten Most Critical Web Application Security Risks (2013), [online] http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf.

[RFC] HTTP Authentication: Basic and Digest Access Authentication. [online] https://www.ietf.org/rfc/rfc2617.txt.

[RiGD13] S. Ristov, M. Gusev, A. Donevski: OpenStack cloud security vulnerabilities from inside and outside. *In: CLOUD COMPUTING 2013, The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization* (2013), 101–107.

[RTSS09] T. Ristenpart, E. Tromer, H. Shacham, S. Savage: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. *In: ACM Conference on Computer and Communications Security (CCS)*, ACM (2009), 199–212.

[SHJS+11] J. Somorovsky, M. Heiderich, M. Jensen, J. Schwenk, N. Gruschka, L. L. Iacono: All Your Clouds are Belong to us – Security Analysis of Cloud Management Interfaces. *In: The ACM Cloud Computing Security Workshop (CCSW)* (2011).

[SIYA11] K. Suzaki, K. Iijima, T. Yagi, C. Artho: Memory deduplication as a threat to the guest OS. *In: Proceedings of the Fourth European Workshop on System Security*, ACM (2011), 1.

[VKFR+12] V. Varadarajan, T. Kooburat, B. Farley, T. Ristenpart, M. M. Swift: Resource-freeing attacks: improve your cloud performance (at your neighbor's expense). *In: ACM Conference on Computer and Communications Security (CCS)*, ACM (2012), 281–292.

[ZeFe08] W. Zeller, E. W. Felten: Cross-site request forgeries: Exploitation and prevention. `https://www.eecs.berkeley.edu/~daw/teaching/cs261-f11/reading/csrf.pdf` (2008).

[ZJRR12] Y. Zhang, A. Juels, M. K. Reiter, T. Ristenpart: Cross-VM side channels and their use to extract private keys. *In: ACM Conference on Computer and Communications Security (CCS)*, ACM (2012), 305–316.