alert("XSS")

Sicherheitslücke bei Vodafone

In der Vodafone-Website klaffte eine kleine Sicherheitslücke, mit fataler Wirkung: Angreifer konnten Kundendaten einsehen, Rechnungen in die Höhe treiben und mehr. Vodafone reagierte wochenlang nicht.

Von Ronald Eikenberg

Schon mit dem ersten Satz hatte Daniel Werner unsere volle Aufmerksamkeit: „In der Onlinepräsenz von Vodafone habe ich Anfang des Monats eine Sicherheitslücke mit enormem Gefahrenpotenzial gefunden", schrieb der Quality Assurance Engineer an die Redaktion. Man könne beliebigen JavaScript-Code in die Vodafone-Website injizieren, es sei sogar möglich, einen Keylogger auf der Site zu platzieren, der die Login-Daten der Kunden ausspäht. Das klingt dramatisch, dennoch war es Werner offenbar nicht gelungen, sich Gehör bei dem Provider zu verschaffen: „Ich habe diesen Umstand Vodafone umgehend per Mail mitgeteilt, jedoch haben diese bis jetzt keinerlei Reaktion gezeigt." Deshalb wandte er sich an die Redaktion von c't und heise Security – in der Hoffnung, dass das Sicherheitsloch endlich gestopft wird, wenn die Presse bei Vodafone nachfragt.

Bevor wir uns mit dem Unternehmen in Verbindung setzten, wollten wir erst mal überprüfen, ob die Lage tatsächlich so fatal ist wie durch Werner geschildert. In seiner Nachricht fanden wir einen Link auf die Privatkundenseite von Vodafone. Als wir die Adresse ansteuerten, öffnete sich eine Seite mit Suchergebnissen auf vodafone.de, die sich merkwürdig verhielt: Zum einen zeigte sie nur den Hinweis `Suchergebnisse für „"` an, zum anderen öffnete sich ein Hinweisfenster mit der Zahl 12. Auffällig an dem präparierten Link war der URL-Parameter `q`, der häufig für Query steht und den Begriff transportiert, den der Websitebesucher in das Suchfeld einer Website eingetippt hat:

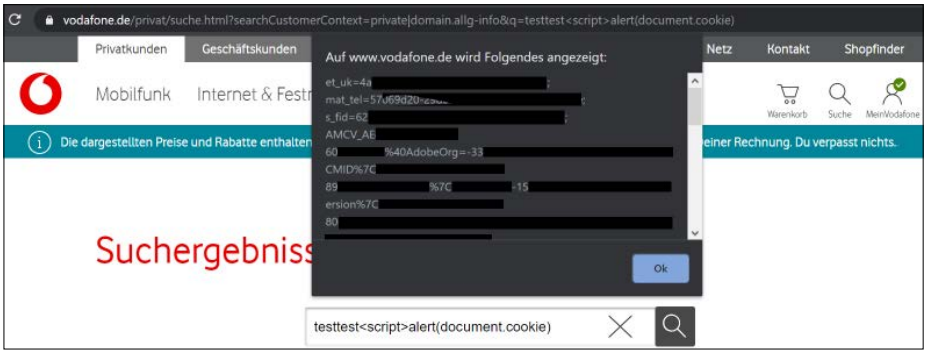
```
q=testtest<script>alert(12)
```

Der Parameter enthielt also nicht nur den Suchbegriff „testtest", sondern auch den JavaScript-Befehl `alert()`, der den Browser anweist, ein Hinweisfenster anzuzeigen. Der Befehl wurde mit dem Wert „12" gefüttert, was zu dem Hinweisfenster passt, das beim Aufrufen der URL angezeigt wurde. Auf den ersten Blick ist das unspektakulär, doch wer sich mit den Sicherheitskonzepten von Browsern und Websites auskennt, bei dem schrillen jetzt die Alarmglocken. Denn die Website hatte tatsächlich JavaScript-Code ausgeführt, der von außen eingeschleust wurde. Wo man einen `alert`-Befehl einschleusen kann, da ist auch beliebiger anderer Code möglich, wie auch weitere Experimente belegten.

Reflektierter Code

Die Wurzel des Problems ist so simpel wie gefährlich: Die Suchfunktion der Vodafone-Website zeigt den eingegebenen Suchbegriff über den Suchergebnissen an, also etwa `Suchergebnisse für „kabel"`, wenn man nach „kabel" gesucht hat. Wurde jedoch nicht nur ein einfacher Suchbegriff, sondern auch JavaScript-Code an die Suchfunktion übergeben, dann wurde dieser ebenfalls in die Ergebnisseite eingefügt – und fatalerweise vom Browser ausgeführt. Die Website hat den Inhalt des Query-Parameters unzureichend überprüft und wiedergegeben. Man spricht in solchen Fällen von „Reflected Cross-Site-Scripting" (Reflected XSS).

Das ist höchst gefährlich, weil der Browser den eingeschleusten Code im Kontext der verwundbaren Website ausführt und ihm somit Zugriff auf sämtliche Ressourcen der Website einräumt. Der Code darf die Website beliebig manipulieren und kann Tastatureingaben belauschen. So wäre ein Keylogger denkbar, der Vodafone-Kunden bei der Eingabe des Passworts ausspioniert. Ferner ist auch das Ausnutzen von Sicherheitslücken in Browser und Plug-ins möglich, sowie das Verbreiten von Malware. Häufig zielen XSS-Angriffe auf die Session-Cookies der Opfer ab: Über das Objekt `document.cookie` kann der eingeschleuste Code sämtliche Cookies abrufen, die der Browser für die



Der eingeschleuste JavaScript-Code kann auf die Sitzungs-Cookies von vodafone.de zugreifen und diese an einen externen Server schicken. Ein Angreifer kann damit die Sitzung des eingeloggten Vodafone-Kunden übernehmen.

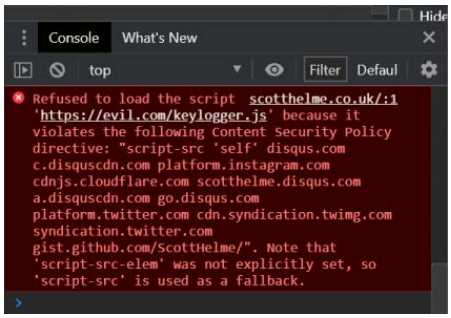
verwundbare Website abgelegt hat. Damit kann der Angreifer leicht die aktive Sitzung des eingeloggten Kunden übernehmen und beliebig schalten und walten.

Großes Missbrauchspotenzial

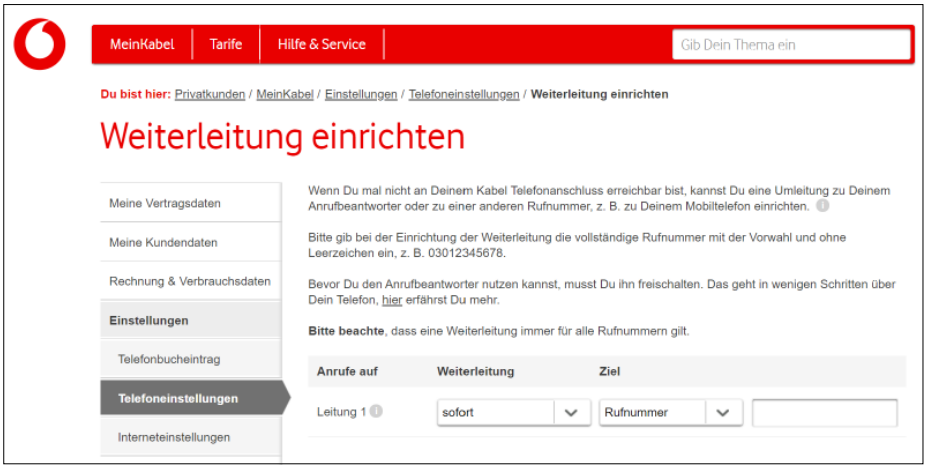
Im Fall von Vodafone wäre es höchstwahrscheinlich möglich gewesen, persönliche Daten sowie Rechnungen einzusehen und sogar eine Rufumleitung einzurichten. Das ist eine gängige Methode, um schnelle Kasse zu machen: Angreifer lassen Rufnummern zu teuren Premiumrufnummern oder ins Ausland umleiten und verdienen an den horrenden Verbindungskosten mit. Die Opfer bemerken den Betrug häufig erst mit der nächsten Telefonrechnung, die astronomisch hoch ausfällt. Weiterhin hängen an dem vodafone.de-Account auch die Mail-Accounts der Kunden, somit wäre eine Übernahme weiterer Online-Accounts vorstellbar gewesen. Dazu müsste der Angreifer lediglich die „Passwort vergessen"-Funktion bei einem anderen Dienst anstoßen, den der Vodafone-Kunde nutzt und bei dem er seine Vodafone-Mailadresse als Backup hinterlegt hat.

Gefahr bekannt

Für einen Angriff hätte der Cyber-Ganove seine Opfer lediglich auf eine spezielle Vodafone-URL locken müssen, zum Beispiel durch eine Mail im Vodafone-Look oder eine automatische Umleitung von einer anderen Seite. Nachdem wir die von Daniel Werner geschilderte Sicherheitslücke verifiziert hatten, setzten wir uns mit Vodafone in Verbindung. Plötzlich kam Bewegung in die Sache: Als wir tags darauf einen Blick auf die Website warfen,



Abgeblitzt: Mit einer Content-Security-Policy können Website-Betreiber verhindern, dass eingeschleuste Skripte ausgeführt werden. In diesem Beispiel verhindert eine Policy, dass keylogger.js von evil.com ausgeführt wird.



Wer Zugriff auf den Vodafone-Account hat, kann unter anderem Kundendaten einsehen und Rufumleitungen einrichten. Gelingt dies einem Angreifer, kann das für den Kunden erhebliche Folgen haben.

hatte das Unternehmen das Sicherheitsleck bereits gestopft. Wenige Tage später erhielten wir auch eine Antwort auf unsere Mail – und unsere Fragen, die wir routinemäßig gestellt hatten. Der Provider erklärte, dass ihm das Problem bereits seit Anfang August bekannt war, was die Sache nicht besser machte: Denn als wir das Unternehmen am 31. August auf die Lücke aufmerksam machten, hatte sich noch nichts getan.

„Hinweise über Missbrauchsfälle oder Auffälligkeiten aus dieser geschlossenen potenziellen Schwachstelle" lägen dem Unternehmen nicht vor, ergänzt der Provider. Warum die von Daniel Werner entdeckte Schwachstelle in diesem prominenten Teil der Website erst mehrere Wochen nach Bekanntwerden geschlossen wurde, bleibt ungeklärt.

XSS verhindern

Der aktuelle Fall zeigt anschaulich, wie gefährlich es für Websitebetreiber sein kann, von außen zugelieferte Daten ungefiltert weiterzuverarbeiten. Im schlimmsten Fall kann dies sogar dazu führen, dass der Code nicht nur im Browser des Besuchers, sondern direkt auf dem Server ausgeführt wird. Es ist unerlässlich, Eingaben von außen mit großer Vorsicht zu verarbeiten und potenziell gefährliche Zeichen herauszufiltern. Die genaue Vorgehensweise hängt von der im Einzelfall eingesetzten Servertechnik und Web-Anwendung ab. Bei PHP-Skripten, die HTML-Output generieren, ist die Funktion `htmlentities()` hilfreich, die Zeichen in Strings in HTML-Entitäten umwandelt, wenn es eine Entsprechung gibt. Aus dem String

```
<script>alert("XSS")</script>
```

wird die Zeichenfolge

```
&lt;script&gt;alert(&quot;XSS&quot;)&
lt;/script&gt;
```

Es fällt auf, dass PHP unter anderem die spitzen Klammern gegen die entsprechenden HTML-Entitäten ausgetauscht hat (`<` und `>`). Sollte die entschärfte Variante über die Website ausgegeben werden, dann wird der String vom Browser zurück in lesbaren Text umgewandelt, der dem Eingabewert entspricht. Entscheidend ist jedoch, dass der Browser diese entschärfte Version nicht als Inline-JavaScript interpretiert und ausführt.

Für den Fall, dass es einem Angreifer dennoch gelingt, Skriptcode in ausführbarer Form in die Website zu schleusen, kann man mit Content-Security-Policys vorsorgen. Dabei handelt es sich um eine Schutzfunktion der Browser, die man über den HTTP-Header der Website konfiguriert. In einer solchen Policy ist etwa festgelegt, dass Inline-Skripte (`<script>...</script>`) generell verboten sind und Skript-Dateien (.js) nur von der Domain der Site nachgeladen werden dürfen. Das macht XSS-Attacken deutlich aufwendiger, denn der Angreifer muss in diesem beispielhaften Fall seinen Code zunächst auf dem Server platzieren, den er attackiert.

Auf vodafone.de waren bei Redaktionsschluss keine Content-Security-Policys aktiv. Offenbar ist das Unternehmen optimistisch, von außen eingeschleusten Code künftig zuverlässig herauszufischen, ehe er ausgeführt wird. *(rei@ct.de)* c't