che, und dann auf dem Clientrechner für die jeweilige Grafikhardware kompiliert. Und die Grafikkarte kann nur auf die Speicherbereiche zugreifen, auf die ihr Treiber den Zugriff erlaubt.

Anders formuliert: Die Sicherheit des gesamten Systems ist beim Einsatz von WebGL von der Sicherheit des Grafikkartentreibers abhängig. Und die sind i. A. nicht in Hinsicht auf besondere Sicherheit entwickelt. Zumal sich die Anforderungen an Grafikausgaben und die Anforderungen an sichere Software diametral gegenüber stehen: Bei der Grafikausgabe kommt es vor allem auf die Performance an, gefolgt von Genauigkeit. Bei der Sicherheit kommt Sorgfalt beim Prüfen aller Parameter an erster Stelle, was sich meist nur zu Lasten der Performance realisieren lässt.

Kommen wir nun zu den konkreten Schwachstellen, die Context Information Security gefunden hat und die von Microsoft ebenfalls bemängelt wurden.

WebGL – Ein Einfallstor für Angreifer?

Die erste Analyse von WebGL wurde am 9. Mai 2011 von James Forshaw veröffentlicht [2]. Der erste Kritikpunkt ist das bereits oben beschriebene Problem des unkontrollierten Zugriffs auf den Speicher, außerdem wurden DoS-Angriffe und das Ausspähen von Bildern fremder Domains bemängelt. Bevor wir zu den möglichen Angriffen kommen, gibt es zuerst einen vereinfachten Überblick über den Aufbau des 3-D-Grafiksystems.

3-D-Grafik im Überblick

Das 3-D-Grafiksystem basiert auf mehreren Schichten (Tabelle 1). Auf der untersten Schicht befindet sich die Hardware mit dem Grafikprozessor (Graphics Processor/Processing Unit, GPU) selbst. Auf dieser Schicht gibt es kein allgemeines API, sondern nur herstellerspezifische Interfaces, die die auf Programmiersprachenebene benötigten Funktionen bereitstellen. Moderne Grafikhardware enthält i. A. von User-Mode-Prozessen individuell programmierbare Einheiten, die so genannten „Shader".

Über der Hardware befindet sich der Treiber, der meistens im Kernel-Mode des Betriebssystems läuft. Er verbindet die herstellerspezifische Low-Level-Hardware mit einem standardisierten Interface wie beispielsweise dem Windows Display Driver Model (WDDM), über das die anderen Komponenten des Betriebssystems auf die GPU zugreifen können.

Über dem Treiber sitzt das Scheduling, für das es mehrere mögliche Orte für seine Implementierung gibt: Im Kernel-Treiber selbst, im Betriebssystem oder auch komplett im User Mode. Seine Aufgabe ist die Verwaltung der Zugriffe durch die verschiedenen auf dem Rechner laufenden Programme. Bei herkömmlichen Grafikkarten greift normalerweise zu jedem Zeitpunkt nur eine Anwendung, beispielsweise ein Windows-Manager, auf die GPU zu. Im Fall von 3-D-Grafiken müssen mehrere Programme gleichzeitig direkten Zugriff auf die Shader und den Speicher der Grafikkarte haben, was über den darauf spezialisierten Scheduler realisiert wird.

Über dem Scheduler sitzt dann die Interface-Library, über die die Benutzerprozesse auf die Grafikkarte zugreifen. Beispiele hierfür sind Direct3D (die auch einige Kernel-Funktionen enthält) und OpenGL als Cross-Platform-Lösung. Die Interface-Library stellt APIs zum Erstellen der anzuzeigenden 3-D-Grafiken, Compiler zur Umwandlung der in einer allgemeinen Sprache geschriebenen Shader-Programme in GPU-spezifischen Code sowie Funktionen für die Verwaltung der Textur-Informationen für den Grafikkartenspeicher bereit. Die Interface-Library ist die höchste Abstraktionsstufe, auf der soweit wie möglich auf hardwarespezifische Funktionen verzichtet wird.

WebGL und seine Auswirkungen

Normalerweise haben die Inhalte im Webbrowser keinerlei Zugriff auf die Hardware. Soll zum Beispiel eine Bitmap-Grafik angezeigt werden, ist dafür entsprechender Code im Webbrowser selbst zuständig. Und der ruft dafür normalerweise eine dafür zuständige Betriebssystemfunktion auf, die dann die eigentliche Darstellung übernimmt. Selbst beim Einsatz von 2-D-Grafikbeschleunigung hat der Browserinhalt selbst keinen direkten Zugriff auf die GPU. Außerdem sind 2-D-Grafiken recht einfach zu prüfen, relativ schnell zu rendern und enthalten generell nur wenige programmierbare Funktionalitäten, die einen direkten Zugriff auf die Grafikhardware erfordern.

WebGL dagegen erlaubt den direkten Zugriff auf die Grafikhardware. Der Shader-Code wird in den GPU-spezifischen Code kompiliert, in den Speicher der Grafikkarte geladen und von der GPU ausgeführt. Wie lange das Rendering der Daten dauert, lässt sich aus den Rohdaten nur schwer ermitteln. Außerdem sind die 3-D-Grafiken zumindest teilweise nur schwer zu prüfen, sodass Sicherheitsbedingungen kaum durchzusetzen sind.

Läuft der Shader-Code erst einmal auf der GPU, kann er nicht mehr so einfach gestoppt werden. Zumindest nicht ohne Auswirkungen auf das gesamte System. Durch entsprechend präparierte Daten sind daher zumindest DoS-

Mode	Schicht
User Mode	Interface
User Mode oder Kernel Mode	Scheduling
Kernel Mode	Treiber
	Hardware/GPU

Tabelle 1: Das 3-D-Grafiksystem basiert auf mehreren Schichten