Sicherheitsprobleme dynamischer Erweiterbarkeit in E-Learning-Systemen

Christian J. Eibl
Lehrstuhl Didaktik der Informatik und E-Learning
Universität Siegen
eibl@die.informatik.uni-siegen.de

Abstract: Dieser Artikel beschäftigt sich mit möglichen Problemen dynamisch nachzuladender Erweiterungsmodule für Lern-Management-Systeme. Neben Vorteilen für eine gesteigerte Flexibilität im Lehr-/Lernprozess werden Risiken des Einsatzes solcher Module aufgrund konzeptueller Schwachstellen eingeführt und diskutiert. An konkreten Beispielen kürzlich aufgedeckter, kritischer Sicherheitslücken im System „Moodle“ werden diese Schwachstellen aufgegriffen und mit der Praxis in Bezug gesetzt.

1 Motivation

Mit Blick auf die Flexibilität und Einsatzmöglichkeiten verschiedener Methoden im Präsenzunterricht ergibt sich im E-Learning ein analoges Bedürfnis nach weitreichender Freiheit in der Wahl derartiger Unterrichtsgestaltungen. Da es sich bei E-Learning-Systemen jedoch um deterministische Anwendungen handelt, die aufgrund dieser Tatsache nicht die Flexibilität eines menschlichen Lehrers erreichen können, ist zumindest der Ruf nach dynamischer Erweiterbarkeit durch Module mithilfe standardisierter Schnittstellen verständlich und sinnvoll.

Erweiterbarkeit durch Module führt zum einen zu einer gewissen Herstellerunabhängigkeit, da nicht nach Feststellung eines Bedarfs für bestimmte Aktivitäten auf eine entsprechende Integration der Funktionalität durch den Hersteller gewartet werden muss, zum anderen ergibt sich gerade bei Open Source Projekten und offenliegenden Schnittstellen für diese Module in der Regel eine Community, die nach eigenen Bedürfnissen eine Vielzahl von Erweiterungen für die Kernanwendung schreibt und diese anderen Nutzern zur Verfügung stellt. Am Beispiel des Open Source Lern-Management-Systems (LMS) Moodle[1], das uns im weiteren Verlauf dieses Artikels als Beispiel dienen soll, ergibt sich so auf der Webseite dieses LMS eine Sammlung von annähernd 500 (offiziell bekannten) Erweiterungen, die von jeder Organisation, die Moodle einsetzt, ohne weitere Kosten und mit nur geringem Aufwand eingebunden und genutzt werden können.

Trotz gewichtiger Vorteile für die Lehre ergeben sich jedoch Probleme bei der unreflektierten und unvorsichtigen Einbindung von Erweiterungsmodulen. Da die Erweiterbarkeit

[1] URL: http://moodle.org

durch Module, die nicht durch einen verantwortlichen Hersteller selbst ausführlich geprüft wurden, die Sicherheit eines Systems gefährden können, werden in diesem Artikel mögliche Schwachstellen dieses Modulansatzes diskutiert. Trotz der Vorteile der zusätzlichen Funktionsvielfalt sollte bedacht werden, dass Produktivsysteme eine gewisse Verlässlichkeit garantieren müssen, um auf lange Sicht qualitativ gute Lehre sichern zu können.

2 Stand der Forschung

Die Begriffe „Modul“ und „Modularisierung“ werden in der Informatik in verschiedenen Bedeutungen verwendet. Der Softwaretechnik folgend sollte Software immer modular aufgebaut werden, so dass Modularisierung als ein Qualitätskriterium von Software verstanden wird [Ba00, POS05]. Diese Modularisierung bedeutet hier jedoch in erster Linie, dass Funktionen auch programmintern hinter Schnittstellen verborgen sind, um die eigentliche Implementierung einer Funktionalität flexibler austauschen zu können, z.B. um die Effizienz einer Funktion durch bessere Algorithmen zu erhöhen. Dieses Prinzip des „information hiding“ ermöglicht eine Erhöhung der Flexibilität bzgl. des Austauschs von einzelnen Programmteilen und verbessert die Wartbarkeit in diesem Sinne. Im Endergebnis resultiert dies in verbesserter Wiederverwendbarkeit und Kostenersparnis.

Obwohl dieser Ansatz der Modularisierung im E-Learning ebenfalls von Belang ist, sind mit Blick auf Erweiterungsmöglichkeiten mit Modulen vor allem Software-Bausteine gemeint, die an ein Kernsystem, das die Grundfunktionalität sicherstellt, angebunden werden können. Peter Hubwieser rät im Rahmen der „Prinzipien didaktischen Handelns“ dazu, „viele unterschiedliche Lehrmethoden“ einzusetzen [Hu07, S. 22]. Um dies bewerkstelligen zu können mithilfe von E-Learning-Systemen, ist eine ausreichende Funktionsvielfalt notwendig. Dieser Aspekt der Funktionsvielfalt wird ebenfalls von Weippl aufgegriffen mit dem Argument, dass Lehrpersonen Vorbehalte gegen zu starre und restriktive Systeme haben könnten, da sie ihre Freiheit in der Lehre und den anwendbaren Methoden einschränken [We05, S. 22f]. Speziell die Möglichkeit, viele verschiedene Aktionen und Interaktionsmöglichkeiten anzubieten, gewinnt mit Blick auf Theorien zum Konstruktivismus an Wert. Weitere Anpassungsmöglichkeiten ergeben sich jedoch auch fachspezifisch durch Besonderheiten in der Repräsentation von Lerninhalten oder der vorherrschenden und bevorzugten Arbeitsweise mit diesen Inhalten.

Ein weiterer Aspekt, der im Bereich der E-Learning-Systeme bzgl. Modulen diskutiert wurde, ist die Funktionsvielfalt und der Umfang, den ein E-Learning-System tatsächlich anzubieten hat. Hier können wir zwischen monolithischen Systemen und einer „Menge gekoppelter Werkzeuge“ [KNW03, S. 3] unterscheiden. Monolithische Systeme, wie sie durch die meisten LMS gegeben sind, versuchen alle relevanten Bereiche von E-Learning innerhalb eines Systems abzubilden und setzen hierfür auf eine identische technologische Basis. Mit relevanten Bereichen für E-Learning seien hierbei alle Phasen eines Lernprozesses verstanden, d.h. neben einer Motivation und der grundlegenden Erarbeitung von neuem Wissen sollen auch Kommunikationsmöglichkeiten geboten werden, um in Gruppen Wissen zu vertiefen, sowie Anwendungsmöglichkeiten, Übungen und Selbsttests, um mit dem neuen Wissen interaktiv arbeiten und dieses festigen zu können. Die Beschränkung auf

eine identische technologische Basis kann sehr nachteilig sein mit Blick auf reduzierte technische Möglichkeiten, z.B. durch ungeeignete Protokolle wie HTTP für jeglichen Datenaustausch (vgl. [ESS07, Ei08]). Dennoch ist die Verknüpfung von Daten in einem monolithischen, gut aufeinander abgestimmtem System aller Wahrscheinlichkeit nach besser als in einer Sammlung unabhängiger Teilsysteme.

Bezüglich der Dynamik von Erweiterbarkeit und der angestrebten Flexibilität im Umgang mit Modulen ist zu berücksichtigen, dass diese Dynamik mit Sicherheitsproblemen verbunden ist. Mit Blick auf Sicherheit und Rechtemanagement sollte beachtet werden, dass im Zweifel bestimmte Aktionen verboten statt genehmigt werden sollten. Es hat sich in Sicherheitskreisen ein „whitelisting“ gegenüber „blacklisting“ durchgesetzt, d.h. alles ist verboten, was nicht explizit erlaubt ist [An01]. Zudem sollte ein Sicherheitskonzept möglichst erschöpfend die Funktionsvielfalt und vor allem mögliche Benutzereingaben berücksichtigen, um nicht durch unerwartete Benutzerinteraktion Sicherheitslücken zu öffnen [Er03, Ma09, MK07]. Trotz dieser in der Theorie weitreichend bekannten Vorkehrungen und Vorgehensweisen zeigen sich in der Praxis dennoch sehr häufig Schwachstellen, die aufgrund von Unkenntnis, fehlendem Sicherheitsbewusstsein oder einfach Unachtsamkeit in ein System integriert und irgendwann durch Angreifer ausgenutzt werden.

3 Theoretische Schwachstellen bei Verwendung von Modulen

In diesem Abschnitt werden sechs grundlegende, konzeptuelle Schwachstellen von dynamischen Erweiterungsmodulen betrachtet. Es sei hierbei deutlich darauf hingewiesen, dass sich diese Betrachtung im Folgenden nicht speziell auf Moodle beschränkt, sondern allgemein (webbasierte) E-Learning-Systeme adressiert. Moodle wird als Beispiel angeführt, da es mittlerweile einen beeindruckenden Verbreitungsgrad erreicht hat und folglich die aufgeführten Schwachstellen besser mit einem bekannten System in Verbindung gebracht werden können.

3.1 Modulanforderungen unklar für Kernsystem

Auf der Zugriffskontrollebene als zentrale Einrichtung des Sicherheitskonzeptes müssen möglichst alle Aktionen im System berücksichtigt werden, um für reelle Anforderungssituationen bestmögliche Anpassungen vornehmen zu können. Dynamische Erweiterungsmöglichkeiten durch später eingebrachte Module stellen hier eine enorme Herausforderung dar. Es ist im Vorwege nicht abzusehen, welche Funktionalität durch ein solches Erweiterungsmodul bereitgestellt werden soll, folglich sind die darin implementierten Aktionen und benötigten Datensätze nicht vorher bekannt. Dies erlaubt nur noch eine Annäherung an ein benötigtes Sicherheitskonzept durch entsprechend geartete Schnittstellen und allgemeine Restriktionen für sensible Daten.

An dieser Stelle muss unterschieden werden, ob es sich bei dem neuen Modul um eine echte Erweiterung handelt, oder ob vorhandene Funktionen durch die neuen mehr oder weniger ersetzt werden. Bei reinen Erweiterungen, z.B. neuen Interaktionsformen und

Hilfsmitteln für kollaboratives Lernen wie eine Whiteboard-Integration für gleichzeitiges Arbeiten an identischen Repräsentationen, fallen neue Daten an und müssen natürlich auch vor unberechtigtem Zugriff geschützt werden. Ein Zugriff auf die bestehenden Datensätze ist jedoch nicht notwendig. Im Gegensatz dazu sind Module wie die Integration eines Mail-Systems, das ein bestehendes, rudimentäres Nachrichtensystem ersetzen soll, durchaus in der Notwendigkeit vorhandene Stammdaten auszulesen, um Kontaktdaten nicht redundant im System speichern zu müssen und um sofort beispielsweise alle Studierenden des eigenen Kurses über dieses System kontaktieren zu können ohne langwierige Vorarbeiten zur Konfiguration.

Für die Sicherheit bedeutet dies, dass in der Regel jedes Modul selbst ein benötigtes Sicherheitsmodell mitbringen muss. Abhängig von der Art des Moduls ist hierbei besondere Vorsicht vonnöten, um nicht zentral relevante Daten durch ein Modul zu gefährden. Einstellungen des Moduls müssen entsprechend angewandt werden, entweder durch Integration in das zentrale Rechtemanagement oder durch spezielle Konfigurationsseiten für jedes Modul einzeln. Eventuelle Zugriffsbeschränkungen sollten feingranular geregelt und möglicherweise rollenabhängige Rechtesysteme durch „best fit"-Ansätze an systemweite Rollen angegliedert werden. Es ist jedoch zu beachten, dass diese Ansätze lediglich vor Missbrauch durch die Anwender schützen bei entsprechender Umsetzung, die Implementierung und damit das Gefährdungspotential auf technischer Ebene wird dabei nicht berücksichtigt.

3.2 Uneingeschränkte Laufzeitumgebung

Ähnlich eines Quarantäne-Bereichs sollte Software in Umgebungen laufen, die selbst bei Fehlern in der Software nicht zu einem kompletten Systemausfall führen können. Zum einen entspricht dies der Forderung danach, einen Prozess ausschließlich unter einem Account mit eingeschränkten Rechten laufen zu lassen, da in diesem Fall nur der Bereich, der mit den entsprechenden Rechten erreichbar ist, nicht jedoch das System als ganzes gefährdet ist. Auf einem Server wird zudem gefordert, nur mit einem Minimalsystem zu arbeiten, um nicht unnötige Komplexität und möglicherweise für einen Angreifer hilfreiche Werkzeuge ins Spiel zu bringen (vgl. Abschnitt 3.4). Zusätzlich werden Termini wie „Sandbox", „chroot jail" oder „Virtualisierung" gebraucht, um deutlich zu machen, dass auch spezielle Bereiche eines Serversystems von anderen getrennt werden können. Unter „Sandbox" ist hierbei der generelle Ansatz dieser Trennung zu verstehen, wohingegen „chroot" bereits auf eine konkrete Technik hinweist und Virtualisierung einen abstrakten Bereich möglicher Lösungen beschreibt, bei denen virtuelle Maschinen als Gastsysteme auf dem eigentlichen Host laufen und auf dessen Systemdaten nicht zugreifen können.

Das Abschotten des Webservers, auf dem das E-Learning-System läuft, wird auf vielen Produktivsystemen bereits betrieben – wenn auch bei weitem nicht auf allen. Dennoch ist mit Blick auf das Gefährdungspotential von nachgeladenen Modulen zumindest für Module mit unklaren Sicherheitsproblemen ein derartiges Konzept in rekursiver Art und Weise ebenfalls sinnvoll, um Moduldaten vom Kern des E-Learning-Systems zu trennen. Diese Trennung schließt vor allem auch die Verwendung getrennter Datenbanken ein, da die Datenablage für das Kernsystem immens sicherheitskritische Daten verwaltet, was bei

bösartigem Zugriff zu einer Kompromittierung des gesamten Systems führen kann. Bei Betrachtung der angebotenen Erweiterungsmodule für Moodle[2] zeigt sich dieses Problem durch die fast omnipräsente Forderung nach Speicherung von Ergebnissen aus Aktionen dieser Erweiterungsmodule. Eine Sicherheitlücke in einem Modul kann folglich bei gemeinsamer Datenhaltung oder Implementierungsfehlern Auswirkung auf das Gesamtsystem haben, so dass eine Sandboxlösung mit Entkopplung von sicherheitskritischen Bereichen anzustreben ist.

3.3 Individualzugriffssteuerung eingeschränkt

Moodle unterscheidet bei Erweiterungmöglichkeiten verschiedene Kategorien: Aktivitätsmodule, Blöcke und Filter. Aktivitätsmodule lassen sich u.a. direkt in einen Kurs integrieren. Sie ermöglichen die Verwendung neuer Methoden und erlauben verschiedene Formen der Interaktion. Blöcke stellen Information in der Benutzungsschnittstelle bereit, so dass Nutzer des Systems auf ihrer persönlichen Startseite beispielsweise über aktuelle Einträge in Foren informiert werden können oder die anstehenden Termine auf einen Blick präsentiert bekommen. Filter erlauben eine Übersetzung von eingegebenen Texten in eine andere Form, d.h. es werden nicht die Inhalte der Originaleingabe, sondern eine verarbeitete Form an dieser Stelle eingesetzt und den Nutzern präsentiert.

Bezüglich eines Rechtemanagements ist zu beachten, dass Aktivitätsmodule und Blöcke nicht generell verwendet werden und für alle Teilnehmer im System zugänglich sind. Sie müssen von Lehrenden explizit für ihre Kurse eingebunden werden. Ob sich deren Verwendung jedoch an weitere Voraussetzungen binden lässt, um eine individuelle Zugriffssteuerung innerhalb eines Kurses zu ermöglichen, ist vom jeweiligen Modul abhängig. Eine Individualzugriffssteuerung ist nur im Rahmen des vorherrschenden rollenbasierten Rechtesystems möglich, so dass für eventuelle Verbote oder explizite Befugnisse neue Rollen generiert werden müssen, um eine derartige Zugriffssteuerung umzusetzen. Es ist folglich nicht direkt, jedoch über Workarounds möglich, in Moodle angepasste Zugriffsbeschränkungen zu erlauben – für Aktivitätsmodule und darin enthaltene Funktionen.

Textfilter auf der anderen Seite gelten systemweit und lassen in der Regel keine individuellen Beschränkungen zu. Sie sind global vom Administrator freizuschalten und nicht kursgebunden. Da hiervon jedoch jegliche textuelle Eingabe im System betroffen ist, ist im Vorwege klar abzuwägen, ob der Einsatz eines bestimmten Filters sinnvoll erscheint. Aufgrund der fehlenden Anpassbarkeit an die individuelle Rechtesituation einer bestimmten Organisation wird in Abschnitt 4 diese Form der Erweiterung als Beispiel für ein Sicherheitsproblem näher erläutert.

3.4 Verallgemeinerung und Komplexität

Zunehmende Komplexität impliziert ein gesteigertes Risiko, dass Sicherheitslücken existieren und übersehen werden. Trotzdem ist tendenziell bei Entwicklern ein Drang nach komplexeren und allgemeingültigeren Software-Bausteinen zu finden. Dies rührt in der

[2] Erweiterungsmodule für Moodle, URL: `http://moodle.org/mod/data/view.php?id=6009`

Regel daher, dass bei existierenden Basiskomponenten eine Verallgemeinerung des Konzepts und eine Erweiterung um zusätzliche Funktionen geringer erscheinen, als eine alternative Neuimplementierung für die fehlenden Funktionen.

Bezogen auf Moodle und in Vorausschau auf das Beispiel in Abschnitt 4 sei an dieser Stelle die Gegenüberstellung des Programms „mimetex“[3] und einer vollständigen LaTeX-Installation, z.B. TeXlive, für die Verwendung im TeX-Filter erwähnt. Diese Anwendungen werden in Moodle verwendet, um textuelle Eingaben in LaTeX-Formelnotation in ein Bild zu übersetzen, das in den HTML-Code von Kursseiten in Moodle integriert werden kann. Mimetex stellt für mathematische Formeln einen sehr eingeschränkten Sprachraum möglicher LaTeX-Ausdrücke zur Verfügung, der jedoch in den meisten Fällen ausreichen dürfte. Dennoch wird seit Version 1.6 von Moodle die mächtigere LaTeX-Umgebung bevorzugt, was es auch erlaubt, neue Symbole einzubinden, Begriffe umzudefinieren und selbst komplexe Inhaltsstrukturen darzustellen. Das Argument, dass mächtigere Systeme für bestimmte Situationen besser wären, hat hier vermutlich überzeugt trotz der meist nicht notwendigen Funktionsvielfalt und Komplexität einer vollständigen Umgebung einschließlich eigener Erweiterungspakete.

Diese Verallgemeinerungen und Komplexitätssteigerungen resultieren jedoch in teilweise unüberschaubaren Systemen und folglich in Sicherheitslücken aufgrund fehlender Kenntnisse aller relevanten Details. In Abschnitt 4 wird eine kritische Sicherheitslücke in Moodle diskutiert, die sich durch diese Designentscheidung ergeben hat.

3.5 Fehlende Kontrollinstanz

Das Fehlen einer Kontrollinstanz, die gewisse Änderungen und Einträge erst absegnen muss, bevor diese aktiv werden, kann sowohl positiv als auch negativ gesehen werden. Es ist sicherlich vorteilhaft im Sinne einer schnelleren Bereitstellung von Funktionalität, dass nicht erst ein designiertes Gremium von Experten als Qualitätskontrolleure die neuen Funktionen begutachten muss. Auf diese Weise kann viel Flexibilität erreicht werden und Anpassungen an sich verändernde Lehr-/Lernsituationen können schnell und effektiv umgesetzt werden. Auf der anderen Seite fehlen jedoch unabhängig Dritte, die mit der nötigen Expertise auf den Programmcode und die dahinter liegenden Konzepte blicken, um so eine Einschätzung der Sicherheit zu bekommen. Es ist zu beachten, dass die Sicherheit von E-Learning-Systemen als webbasierte Anwendungen aufgrund der möglicherweise immens großen Anzahl potentieller Nutzer, die sich auf das komplette Internet verteilen können, als wichtigen Qualitätsfaktor verstanden werden muss. Folglich sollte vor allem in Produktivsystemen mit entsprechender Verfügbarkeitsrelevanz die Sicherheit über möglichen Flexibilitätsforderungen stehen. Ein flexibles, aber aufgrund von Sicherheitsvorfällen nicht verfügbares System ist nicht zu gebrauchen.

Ein Kompromiss analog zu Linux-Distributionen wäre, Entwickler von Distributoren zu trennen. Das bedeutet, dass Entwickler einen gewissen Stand nach eigener Einschätzung als sicher und stabil freigeben können, die eigentlichen Nutzer bei Verwendung von Distributoren als kontrollierende Mittelsmänner jedoch nur die zuletzt als stabil und sicher

[3] URL: `http://www.forkosh.com`

markierte Version erhalten. So bieten sich verschiedene Vorteile bzgl. der Flexibilität und Sicherheit an, da jeder Nutzer die Wahl hat zwischen stabilen und potentiell instabilen Releases und statt einer starren Kontrollinstanz auf Herstellerseite eine große Community als Distributionskern verwendet werden kann. Die Verantwortlichkeit ist zwar in diesem Fall nicht in demselben Maße gegeben, als wenn ein (kommerzieller) Hersteller die Fehlerfreiheit seines Produktes garantieren muss, aber für die gewonnene Flexibilität kann dieser Nachteil in Kauf genommen werden.

3.6 Zugriff auf Fremddaten

Wie oben bereits erwähnt, verwenden Module in Moodle dieselben Datenhaltungsvorrichtungen, wie die Kernelemente. Da weiterhin häufig ein Einsperren des kompletten Webservers in eine abgesicherte Laufzeitumgebung nicht umgesetzt wird, ergeben sich auch systemweite Sicherheitsprobleme durch die Verwendung dynamischer Skripte, wie sie bei Moodle durch die Verwendung von PHP gegeben sind. Prinzipiell müssen wir zwischen Daten im E-Learning-System selbst und globalen Daten des Servers unterscheiden. In diesem Abschnitt betrachten wir nur Daten im E-Learning-System. Das Beispiel im folgenden Abschnitt wird sich ausführlich mit globalen Datenzugriffen beschäftigen.

Ein Modul in Moodle kann seine Daten in der zentralen Datenbank ablegen. Hierbei wird für das komplette E-Learning-System ein eindeutiger Datenbanknutzer verwendet, um alle Operationen mit der Datenbank durchzuführen. Die Module sehen als Trennung zwischen vorhandenen Daten von Moodle und neuen Daten durch das Modul lediglich verschiedene Tabellen vor, was jedoch auch in erster Linie an bisher unbekannten Datenrelationen und dem Datenkonzept liegt. Durch die fehlende Trennung in verschiedene logische Datenbanken bzw. einer Unterscheidung der Datenbankbenutzer kann es bei Fehlern im Modul dazu kommen, dass der verwendete Datenbankzugang aufgedeckt wird und potentiell alle Daten ausgelesen oder manipuliert werden.

Da die Module an sich direkt mit dem Moodle-Kern verbunden sind, können E-Learning-spezifische Daten vergleichsweise einfach abgerufen und manipuliert werden bei entsprechenden Fehlern. Ein typisches Angriffsmuster und häufige Schwachstelle webbasierter Anwendungen sind sog. SQL-Injection-Angriffe [Ma09], bei denen bösartige SQL-Statements in Eingabefelder von Web-Formularen eingegeben und ungefiltert an die Datenbank gesendet werden. Eine einzige Lücke in einem eingebundenen Modul kann hierbei ausreichen, um alle Daten der Datenbank zu kompromittieren.

4 Beispiel: TeX-Filter in Moodle

Aufgrund eines fehlenden Sandbox-Prinzips und der uneingeschränkten Einbindung in das Kernsystem können Module zu systemweiten Risiken führen. Dies soll am Beispiel des TeX-Filters bei Moodle veranschaulicht werden.

Wie bereits in Abschnitt 3.4 angedeutet, wird unter dem Begriff „TeX-Filter“ ein Eingabefilter verstanden, der dafür sorgt, dass Formeln in TeX-Notation (eingeschlossen in

```
$$ (a + b)^n = \sum_{k=0}^{n} {n \choose k} a^{n-k} \cdot b^{k} $$
```

$$\Downarrow$$

$$(a + b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} \cdot b^{k}$$

Abbildung 1: Beispiel: ordnungsgemäße Formelnotation.

doppelte Dollarzeichen: „$$ ... $$“) im Hintergrund in Bilder umgewandelt werden, die die Formeln korrekt darstellen und einfach in eine HTML-Seite integriert werden können (vgl. Abb. 1). So ist es möglich, dass auch mathematisch orientierte Fächer ihre Inhalte angemessen in Moodle darstellen können. Dieser Filter steht allen Nutzern im System nach Freischaltung durch den Administrator zur Verfügung, eine feinere Einstellung und Freigabe für bestimmte Benutzer bzw. eine Beschränkung auf bestimmte Kurse ist nicht möglich (→ 3.3: Individualzugriffssteuerung), wodurch die Anzahl möglicher Angreifer im Falle von Sicherheitsproblemen unnötig groß gehalten wird.

Innerhalb weniger Monate sind in diesem besagten Filter jedoch zwei kritische Sicherheitslücken entdeckt und gemeldet worden. Die erste Lücke wurde im Oktober 2008 gefunden und im Dezember, nachdem ein Patch verfügbar war, über die Bugtraq-Mailingliste von Security Focus publiziert [POP08]. Sie bezog sich auf einen Parameter zur Pfadangabe in der Datei „texed.php“. Diese Datei hat keine tragende Funktion für den Filter und seine Funktionsweise in Moodle. Die Datei dient Autoren jedoch dazu, die Syntax für ihre Formeleingaben zu kontrollieren und ggf. anzupassen. Die Lücke erforderte in diesem Zusammenhang, dass die Option „register_globals“ im PHP Verarbeitungsmodul des Webservers angestellt ist, was seit Version 4.2 (April 2002) aus Sicherheitsgründen standardmäßig nicht mehr der Fall ist. Diese Sicherheitslücke ist daher von eher theoretischer Bedeutung, erlaubt jedoch im Fall der gegebenen Bedingungen das Einschleusen und Ausführen von beliebigem Code, so dass das Moodle-System als ganzes kompromittiert werden kann und auch das Serversystem ausgekundschaftet und ggf. geschädigt werden kann, z.B. durch Nachladen von Schadprogrammen wie Rootkits.

Eine weitere kritische Sicherheitslücke, auf die im Folgenden genauer eingegangen werden soll, ist durch den Autor Mitte März 2009 gefunden und den Moodle-Entwicklern zeitnah mitgeteilt worden (vgl. Bug MDL-18552). Diese Sicherheitslücke vereint nahezu alle oben genannten konzeptuellen Probleme und eignet sich daher besonders gut für eine exemplarische Darstellung der Notwendigkeit der Beachtung gegebener Anforderungen. Konkret handelt es sich bei der Schwachstelle ebenfalls um eine Sicherheitslücke, die auf fehlender Eingabeüberprüfung beruht. In diesem Fall ist jedoch kein Parameter innerhalb der Programmlogik betroffen, der direkt von den PHP Routinen ausgewertet oder an eine Datenbank weitergeleitet wird, sondern die Verwendung von LaTeX zur Evaluation und Übersetzung der eingegebenen Formeln. Seit Moodle Version 1.6 (Juni 2006) wird eine vollständig LaTeX-Umgebung auf dem Server bevorzugt gegenüber dem mitgelieferten „mimetex“, das nur einen begrenzten Befehlssatz von LaTeX implementiert und daher weniger mächtig und flexibel ist (→ 3.4: Verallgemeinerung). Vor allem diese Verallgemeinerung führt jedoch zu einer enormen Komplexitätssteigerung bezüglich

```
$$ \input{../../../htdocs/lms/moodle/config.php} $$
```

⇓

<?php///MoodleConfigurationFile
unset(CFG);
CFG− > dbtype =′ mysql′;CFG-¿dbhost = 'localhost'; CFG− > dbname =′
moodle182′;CFG-¿dbuser = '[redacted]; CFG− > dbpass =′ [redacted];CFG-¿dbpersist
= false; CFG− > prefix =′ mdl_;
CFG− > wwwroot =′ http : //141.99.50.135/lms/moodle′;CFG-¿dirroot
= '/home/htdocs/lms/moodle'; CFG− > dataroot =′ /var/www/localhost/moodledata182′;CFG-
¿admin = 'admin';
CFG− > directorypermissions = 00777; //try02777onaserverinSafeMode
require_once("CFG-¿dirroot/lib/setup.php"); // MAKE SURE WHEN YOU
EDIT THIS FILE THAT THERE ARE NO SPACES, BLANK LINES, //
RETURNS, OR ANYTHING ELSE AFTER THE TWO CHARACTERS
ON THE NEXT LINE. ?¿

Abbildung 2: Beispiel: Aufdecken der Konfigurationsdatei von Moodle.

der Handhabung dieses Filters, da TeX als Basissystem für die Verwendung in dem Eingabefilter eine Turing-vollständige Sprache darstellt, die auch Dateisystemoperationen in begrenztem Maße unterstützt. Die Folge ist, dass neben der Möglichkeit, vertrauliche Informationen einzulesen und einem Angreifer darzustellen auch Code unter bestimmten Umständen eingeschleust werden kann. Letzteres wurde aufgrund der standardmäßig sehr restriktiven Voreinstellung von LaTeX für die Dateiausgabe (vgl. Option „openout_any“) nicht weiter untersucht.

Die Problematik der Verallgemeinerung und des Drangs nach flexiblen und universell einsetzbaren Modulen hat in diesem Fall die Sicherheitsbedenken ausblenden lassen (→ 3.5: Kontrollinstanz). Auf der Webseite von mimetex wird explizit auf Sicherheitsrisiken hingewiesen, welche in der Implementierung von mimetex berücksichtigt wurden:

> „\input{filename} behaves just like the corresponding LaTeX command [...] Moreover, for security, absolute paths with leading /'s or \'s, and paths with ../'s or ..\'s, are not permitted.“
> (`http://www.forkosh.com/mimetexmanual.html`; 19.03.2009)

Die vollständige LaTeX-Installation, wie in dem Zitat angedeutet, unterstützt die Funktion „\input“ zum Einbinden externer Dateien. Die Sicherheitsmaßnahmen, wie im Zitat erwähnt, sind jedoch bei LaTeX standardmäßig ausgestellt aufgrund üblicherweise anderer Einsatzszenarien im Vergleich zu mimetex. Ein Angreifer kann daher bei ausreichender Kenntnis des anzugreifenden Systems vertrauliche Dateiinhalte anzeigen lassen (→ 3.6: Zugriff auf Fremddaten). Abbildung 2 zeigt das Ergebnis bei Einlesen der vertraulichen Konfigurationsdatei von Moodle. Die Konfiguration zeigt hier deutlich, auf welchem Server sich die entsprechende Datenbank von Moodle finden lässt. Weiterhin werden Benutzername und Kennwort im Klartext angezeigt (in der Abbildung geschwärzt). Dies ermöglicht dem Angreifer die gefundenen Daten für einen Direktzugriff auf den Datenbankserver zu verwenden, wodurch sich alle Einträge einschließlich Benutzer- und Kursdaten vollständig manipulieren oder sogar löschen lassen. Der Angreifer kann sich einen

Benutzer mit Administrationsrechten anlegen, um über das System selbst eine angenehmere Darstellung der Daten zu erlangen. Der einzige Schutz hiergegen ist, die Datenbank nicht nach außen zugänglich zu machen, sondern über eine rein lokale Socketverbindung zwischen Webserver und Datenbanksystem zu arbeiten.

Es ist zu beachten, dass nicht nur die Konfigurationsdatei, sondern alle Dateien auf dem Server, die mit den Rechten des Webservers auslesbar sind, auch über diesen Ansatz aufgedeckt werden können (→ 3.2: uneingeschränkte Laufzeitumgebung). Dies schließt Dateien wie „/etc/passwd“ ein, um herauszufinden, welche Benutzer auf dem Server existieren, und ob diese bei erfolgreichem Brute-Force-Angriff zum Passwortraten eine interaktive Shell erhalten. Zum anderen können aber auch alle Dateien, die innerhalb von Moodle hochgeladen wurden und damit im Datenverzeichnis von Moodle abgelegt sind, über diesen Weg trotz eigentlich fehlender Berechtigung ausgelesen werden, z.B. auch ohne in einen entsprechenden Kurs eingetragen zu sein.

5 Zusammenfassung und Ausblick

In diesem Artikel wurden Ziele und mögliche Schwachstellen bei der Verwendung dynamischer Erweiterungsmodule vorgestellt. Die dynamische Erweiterbarkeit ist von hohem Stellenwert, um Ängsten bzgl. der Einschränkung der Freiheit in der Lehre durch zu restriktive Systeme entgegen zu wirken. Nichtsdestotrotz ergeben sich Sicherheitsprobleme, so dass selbst die Verwendung harmlos erscheinender Eingabefilter die Gesamtsicherheit des Systems beeinträchtigen können.

Am Beispiel von Moodle wurden konkrete Sicherheitsprobleme diskutiert. Es ist hier zu beachten, dass trotz der hohen Gefährdung durch diese Schwachstellen, die Wahrscheinlichkeit, dass diese Lücken in Produktivsystemen zu finden sind, als eher gering eingestuft werden kann. Der erste Fall der Code-Einschleusung und -Ausführung durch fehlerhafte Behandlung eines Parameters erfordert für die tatsächliche Anwendbarkeit, dass eine bereits seit 2002 standardmäßig ausgeschaltete Option in PHP gesetzt ist. Sofern hier nicht ein Administrator entgegen dem expliziten Rat der PHP Programmierer gehandelt hat, ist die Wahrscheinlichkeit der Verwundbarkeit sehr gering.

Die ausführlicher behandelte Lücke ist diesbezüglich interessanter, da die Moodle-Entwickler sich explizit gegen das sicherere Konzept einer eingeschränkten Funktionalität entschieden haben und erst die Standardeinstellung diese Lücke ermöglicht. Dennoch ist in den meisten Fällen davon auszugehen, dass entweder der TeX-Filter aufgrund fehlender Mathematik-Inhalte in Kursen deaktiviert ist, oder zumindest auf den Minimalsatz über mimetex zurückgegriffen wird. Eine vollständige LaTeX-Installation ist weniger von server- als von clientseitiger Sinnhaftigkeit und wird daher vermutlich von kaum einer Organisation auf dem Server angeboten.

Obwohl wir uns in diesem Artikel vornehmlich auf konkrete Sicherheitslücken am Beispiel eines einzelnen LMS abgestützt haben, bestehen diese allgemeinen konzeptuellen Schwachstellen bei Modulen auch generell für andere LMS. Die Verwendung von Dynamik impliziert, dass vor Auslieferung einer Software unmöglich alle zukünftigen Konfigu-

rationen vorausgesehen werden können. Folglich muss mit Annahmen gearbeitet werden, die unmöglich alle Fälle exakt abdecken können. Dieses Maß an Unsicherheit führt zu Risiken, die zumindest durch abgeschottete Laufzeitumgebungen abgemildert werden sollten, was sich in der Praxis jedoch fast nicht finden lässt. Hier ist explizit Aufklärungsbedarf und weitere Forschung bezüglich Flexibilität trotz erhöhter Sicherheit und bezüglich Performance-Bedenken durch zusätzliche Kontrollinstanzen vonnöten.

Literaturverzeichnis

[An01] Anderson, R.J.: Security Enginieering: A Guide to Building Dependable Distributed Systems. Wiley Computer Publishing, New York, 2001.

[Ba00] Balzert, H.: Lehrbuch der Software-Technik, Band 1, Software-Entwicklung. Spektrum Akademischer Verlag, 2.. Auflage, 2000.

[Ei08] Eibl, C.J.: Vertraulichkeit persönlicher Daten in Lern-Management-Systemen. In: Seehusen, S.; Lucke, U.; Fischer, S. (Hrsg.): DeLFI 2008. LNI Band 132, Bonn, September 2008. Köllen Druck+Verlag, Seiten 317–328.

[Er03] Erickson, J.: Hacking: The Art of Exploitation. No Starch Press, Oktober 2003.

[ESS07] Eibl, C.J.; von Solms, S.H.; Schubert, S.: Development and Application of a Proxy Server for Transparently, Digitally Signing E-Learning Content. In: Venter, H.; Eloff, M.; Labuschagne, L.; Eloff, J.; von Solms, R. (Hrsg.): New Approaches for Security, Privacy and Trust in Complex Environments. IFIP TC-11, Proc. of IFIPsec 2007, Springer Science+Business Media, Boston, 2007, Seiten 181–192.

[Hu07] Hubwieser, P.: Didaktik der Informatik. Springer, Berlin, 3. Auflage, 2007.

[KNW03] Kerres, M.; Nattland, A.; Weckmann, H.-D.: Hybride Lernplattformen und integriertes Informationsmanagement an der Hochschule. In: Dittrich, König, Oberweis, Rannenberg, Wahlster (Hrsg.): Informatik 2003. Innovative Informatikanwendungen, Band 2, 2003, Seiten 90–96.

[Ma09] Martin, B.; Brown, M.; Paller, A.; Christley, S.: 2009 CWE/SANS Top 25 most Dangerous Programming Errors. Online: `http://cwe.mitre.org/top25`, Januar 2009.

[MK07] Meucci, M.; Keary, E. (Hrsg.): OWASP Testing Guide 2.0. Open Web Application Security Project (OWASP Foundation), 2007.

[POP08] Parata, A.; Ongaro, F.; Pellerano, G.: Moodle 1.9.3 Remote Code Execution. Security Focus – BugTraq, online: `http://www.securityfocus.com/archive/1/499172/30/840/threaded`, Dezember 2008. [19.03.2009]

[POS05] Pankratius, V.; Oberweis, A.; Stucky, W.: Lernobjekte im E-Learning – Eine kritische Beurteilung zugrunde liegender Konzepte anhand eines Vergleichs mit komponentenbasierter Software-Entwicklung. In: 9. Workshop Multimedia in Bildung und Wirtschaft. TU Ilmenau, September 2005.

[We05] Weippl, E.R.: Security in E-Learning. Springer Science+Business Media, New York, 2005.