Praktikum Anwendungssicherheit
Hochschule der Medien Stuttgart

Protokoll

II: Parametermanipulation

Verfasser:	Benjamin Zaiser
E-Mail:	bz003@hdm-stuttgart.de
Studiengang:	Computer Science and Media
Semester:	2
Datum:	5. November 2008

Inhaltsverzeichnis

1 Zum Aufbau des Protokolls

Das Protokoll ist wie folgt aufgebaut: zu jeder Übung gibt es ein Kapitel. Jedes Kapitel hat nach Möglichkeit folgende Unterpunkte:

Hintergrundinformationen Informationen, die für die Durchführung der Übung, bzw. für das Verständnis der Aufgabenstellung erforderlich sind.

Aufgabenstellung Welche Aufgabe soll durchgeführt werden; was ist das Ziel der Übung.

Lösung/Durchführung Wie wurde die Aufgabe gelöst; welche Probleme traten dabei auf.

Erkenntnis Was lernt man aus der Aufgabe; welche Erkenntnis könnte in die Entwicklung einer Web-Applikation einfließen.

2 WebGoat: Role based access control

2.1 Hintergrundinformationen

In einem System mit einer Benutzerverwaltung kann jeder Nutzer einer bestimmten Gruppe angehören, die ein bestimmtes Set an Rechten besitzt. Dabei kann es jedoch vorkommen, dass die Abfrage, ob der aktuelle Benutzer die gerade ausgeführte Aktion überhaupt durchführen darf, an der falschen Stelle platziert wurde oder womöglich vom Entwickler gar nicht implementiert wurde.

Bei den nun folgenden Aufgaben kann sich ein Benutzer durch ein Login-Formular am System anmelden. Danach kann durch Auswahl eines anderen Benutzers dessen Profil angezeigt werden. Bestimmte Benutzer haben außerdem das Recht, einen markierten Benutzer zu löschen.

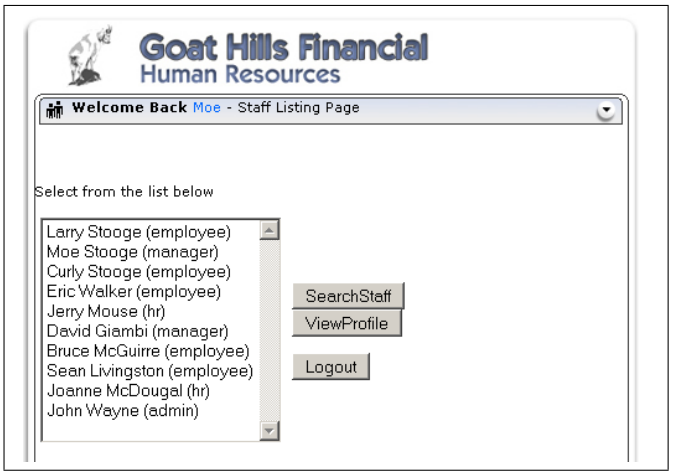

Abbildung 1: Formular, um ein ausgewhhltes Benutzerprofil anzeigen zu lassen

2.2 Aufgabenstellung 1

Ein Benutzer, der eigentlich kein Recht besitzt, einen anderen Benutzer zu löschen, soll diese Aktion nun trotzdem ausführen.

2.2.1 Lösung/Durchführung

Zunächst wurde die Aktion „View Profile“ für einen markierten Benutzer ausgeführt. Dabei wurden die gesendeten POST-Daten mithilfe der Firefox Erweiterung „HackBar“ analysiert.

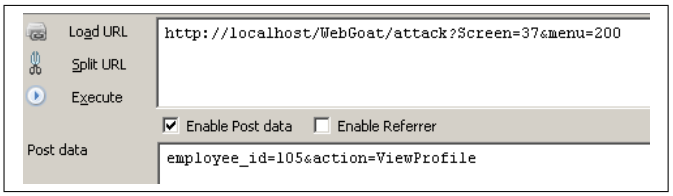

Abbildung 2: POST Daten

Wie man sieht, werden zwei Parameter bei dem Vorgang übertragen: die ID des ausgewählten Benutzers („employee_id“) und die Aktion, die auf den ausgewählten Benutzer ausgeführt werden soll („action“). Durch die Aktions-Bezeichnung „ViewProfile“ liegt es nahe, dass es wahrscheinlich auch eine Aktion gibt, die „DeleteProfile“ oder „DelProfile“ geben könnte. Die POST-Daten werden nun entsprechend geändert und als neue Aktion wird „DeleteProfile“ eingetragen.

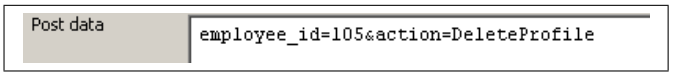

Abbildung 3: Modifizierte POST-Daten

Wenn nun die Abfrage, ob der aktuelle Benutzer die angegebene Aktion ausführen darf fehlerhaft oder gar nicht implementiert ist, müsste die Aktion den Benutzer mit der ID 105 löschen. Nach dem Absenden der modifizierten POST-Daten wird tatsächlich der ausgewählte Benutzer gelöscht.

2.2.2 Erkenntnis

Der „DeleteProfile“ Button wurde korrekt ausgeblendet, da der angemeldete Benutzer nicht das Recht besitzt, andere Benutzer zu löschen. Leider vergaß der Entwickler aber, die Zugriffsrechte bei dem Abarbeiten der POST-Daten nochmals durchzuführen. Selbst wenn die Applikation eine bestimmte Funktion in der GUI versteckt, kann nicht davon ausgegangen werden, dass diese Funktion niemals von dem Benutzer ausgeführt werden kann. Alles was vom Client kommt muss auf seine Gültigkeit hin überprüft werden. Dies beinhaltet auch die Prüfung der Rechte.

2.3 Aufgabenstellung 2

Der offensichtliche Bug aus Aufgabenstellung 1 soll nun behoben werden.

2.3.1 Lösung/Durchführung

Nach dem Einlesen in die Architektur der WebGoat Applikation wurde eine Klasse gefunden, die einen Request entsprechend weiterleitet (Dispatcher). Z.B. wird ein „ViewProfile“ Request an die entsprechende Klasse weitergeleitet, die die Profile eines Benutzer ausgeben kann. Es bietet sich an, gleich zu Beginn des Programmablaufs und an dieser zentralen Stelle, eine Rechteprüfung einzubauen.

```
public void handleRequest(WebSession s)
{
    // Here is where dispatching to the various action handlers happens.
    // It would be a good place verify authorization to use an action.

    // System.out.println("RoleBasedAccessControl.handleRequest()");
    if (s.getLessonSession(this) == null) s.openLessonSession(this);

    String requestedActionName = null;
    try
    {
        requestedActionName = s.getParser().getStringParameter("action");
    } catch (ParameterNotFoundException pnfe)
    {
        // Let them eat login page.
        requestedActionName = LOGIN_ACTION;
    }
    // System.out.println("Requested lesson action: " + requestedActionName);

    try
    {
        DefaultLessonAction action = (DefaultLessonAction) getAction(requestedActionName);
        if (action != null)
        {
            // System.out.println("RoleBasedAccessControl.handleRequest() dispatching to: " +
            // action.getActionName());
            if (!action.requiresAuthentication())
            {
                // Access to Login does not require authentication.
                action.handleRequest(s);
            }
            else
            {
                //***************CODE HERE*************************

                if(action.getActionName().equals(DELETEPROFILE_ACTION) &&
                        !isAuthorized(s, getUserId(s), RoleBasedAccessControl.DELETEPROFILE_ACTION)){
                    throw new UnauthorizedException();
                }
```

Abbildung 4: Eingefügte Rechteprüfung in „handleRequest“

Die Methode „isAuthorized“ überprüft, ob die angegebene Userld das Recht besitzt, die Aktion „DELETEPROFILE_ACTION“ auszuführen. Gibt die Methode „false“ zurück, hat

der Benutzer nicht das Recht. Es wird dann eine „UnauthorizedException“ geworfen und das Programm wird beendet.

Die Überprüfung soll selbstverständlich nur stattfinden, wenn auch ein „Delete“ Request angefordert wurde. Anfangs wurde diese Bedingung nicht beachtet. Dadurch entstand dann das Problem, dass der Benutzer gar keine Aktion mehr durchführen konnte, da er ja das Recht die „DELETEPROFILE_ACTION“ auszuführen nicht besitzt. Dadurch brach die Applikation stets mit der „UnauthorizedException“ ab.

Wichtig ist auch, die korrekte Exception zu werfen. Eine „UnauthenticatedException“ wäre an dieser Stelle semantisch unpassend. Durch die „UnauthorizedException“ wird die korrekte Exception geworfen und die WebGoat Applikation erkennt dann auch die korrekte Lösung der Übungseinheit.

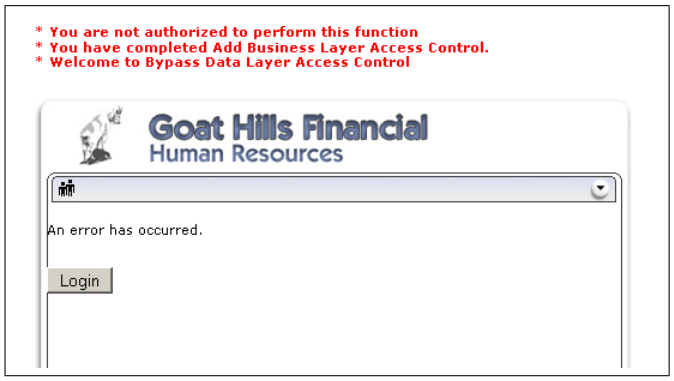

Abbildung 5: Übungseinheit wurde erfolgreich beendet

2.3.2 Erkenntnis

Es bietet sich an, stets zu Beginn der Abarbeitung eines Requests eine Rechteprüfung durchzuführen. Dadurch wird die Sicherheit der Applikation erhöht und bleibt trotzdem noch übersichtlich.

2.4 Aufgabenstellung 3

Ein Benutzer besitzt das Recht, das Profil anderer Benutzer anzuzeigen. Allerdings ist die Rechtevergabe in dem System feingranularer als nur auf Aktionsebene („Business Layer“). Ein Benutzer kann z.B. nur ganz bestimmte andere Benutzer anzeigen lassen („Data Layer“). Es soll nun versucht werden, das Profil eines anderen Benutzers anzeigen zu lassen, auf den eigentlich kein Zugriff bestehen sollte.

2.4.1 Lösung/Durchführung

Aus Aufgabenstellung 1 sind die bei einer Aktion übertragenen POST-Daten bereits bekannt. Mit der „ViewProfile“ Aktion kann der angegebene Benutzer „employee_id“ angezeigt werden. Durch das Eingeben einer passenden employee_id (z.B. durch Inkrementierung) können nun

die Profile anderer Benutzer angezeigt werden, da die Überprüfung der Rechte fehlerhaft implementiert wurde.

2.4.2 Erkenntnis

Wie in Aufgabenstellung 1 schon erkannt, muss die Überprüfung nicht nur auf Aktionsebene stattfinden, sondern auch eine Ebene tiefer, auf der Datenebene.

2.5 Aufgabenstellung 4

Der offensichtlichte Bug aus Aufgabenstellung 3 soll nun behoben werden.

2.5.1 Lösung/Durchführung

Die Rechteprüfung auf Datenebene muss in der ViewProfile Klasse durchgeführt werden. Durch Eingabe von STRG+Leertaste werden durch die Entwicklungsumgebung „Eclipse“ alle möglichen Methoden angezeigt. Interessant ist hierbei die Methode „isAuthorizedForEmployee“, die überprüft, ob der angegebene User für die angegebene employee_id authorisiert ist. Ist dies nicht der Fall, so wird die Abarbeitung des Requests mit einer UnauthorizedException abgebrochen.

```
public ViewProfile(GoatHillsFinancial lesson, String lessonName, String actionName)
{
    super(lesson, lessonName, actionName);
}

public void handleRequest(WebSession s) throws ParameterNotFoundException, UnauthenticatedException,
        UnauthorizedException
{
    getLesson().setCurrentAction(s, getActionName());

    if (isAuthenticated(s))
    {
        int userId = getIntSessionAttribute(s, getLessonName() + "." + RoleBasedAccessControl.USER_ID);
        int employeeId = -1;
        try
        {
            // User selected employee
            employeeId = s.getParser().getIntParameter(RoleBasedAccessControl.EMPLOYEE_ID);
        } catch (ParameterNotFoundException e)
        {
            // May be an internally selected employee
            employeeId = getIntRequestAttribute(s, getLessonName() + "." + RoleBasedAccessControl.EMPLOYEE_ID);
        }

        if(!isAuthorizedForEmployee(s, userId, employeeId)){
            throw new UnauthorizedException();
        }

        Employee employee = getEmployeeProfile(s, userId, employeeId);
        setSessionAttribute(s, getLessonName() + "." + RoleBasedAccessControl.EMPLOYEE_ATTRIBUTE_KEY, employee);


    }
```

Abbildung 6: Rechteprüfung auf Daten-Ebene

2.5.2 Erkenntnis

Je nach dem, wie Granular die Rechtevergabe in einem System sein soll, muss dies auch auf den unterschiedlichen Ebenen, auf denen ein Request bearbeitet wird, beachtet werden. Methoden, die die Prüfung der Rechte geschickt kapseln bieten sich hierbei sehr gut an, da so der Programm-Code sehr sauber und nachvollziehbar gehalten werden kann.

3 Hidden Field Webgoat: Exploit Hidden Fields

3.1 Hintergrundinformationen

Ein Online-Shop System beinhaltet z.B. in der Produkt-Detailansicht ein Eingabeformular, in dem die gewünschte Menge des Artikels bestellt werden kann. Das Formular beinhaltet also eine Input Box, um die Menge eingeben zu können und einen Submit Button, um das Formular an den Server zu senden. Mit HTML ist es auch möglich, sogenannte „hidden-fields“ in ein Formular einzubinden. Diese Felder können vom Benutzer nicht bearbeitet werden, sondern name und value werden vom Entwickler fest definiert. Für den Entwickler können die hidden-fields manchmal hilfreich sein, wenn z.B. ein Formular an ein PHP Skript gesendet wird. Durch ein Hidden-Field kann dann die gewünschte Aktion bearbeitet werden:

```
<input type=hidden name=action value=calcValues />
```

3.2 Aufgabenstellung

In dem gegebenen Bestellformular soll versucht werden, den Preis des Artikels zu verändern. Das Shop-System soll dann den veränderten Preis für die weiteren Berechnungen verwenden (Warenkorb, Rechnungssumme, ...).

Abbildung 7: Bestellformular

3.3 Lösung/Durchführung

Zunächst wird der HTML Quelltext mithilfe der Erweiterung „FireBug“ betrachtet. Dabei fällt auf, dass in dem Formular ein hidden-field verwendet wird mit dem Namen „Price“. Es könnte nun sein, dass das Value Attribut des hidden-fields den Preis des Artikels für den weiteren Bestellvorgang verwendet wird.

Abbildung 8: Hidden-Field mit Preisangabe

Mithilfe der Firebug Erweiterung ist es möglich, ganz einfach den Value zu verändern.

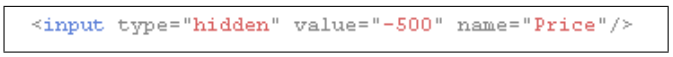

Abbildung 9: Preis Value verändert

Nach dem Senden des Formulars an den Webserver, wird tatsächlich der Preis aus dem hidden-field ausgelesen und für die weitere Verarbeitung (Warenkorbsumme, Bestellsumme, Rechnung, ...) verwendet.

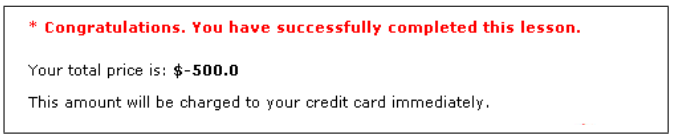

Abbildung 10: Artikel wurde mit verändertem Preis bestellt

3.4 Erkenntnis

Alles was zum Client gesendet wird, kann von diesem auch verändert werden. Im Umkehrschluss: alles was vom Client empfangen wird, muss auf seine Gültigkeit hin überprüft werden. Kurzum sollte der Entwickler für die temporäre Speicherung von Daten, wie z.B. den Preis eines Artikels eine SESSION Variable verwenden, die auf dem Server gehalten wird oder einfach den Preis in einer Datenbank speichern und während des gesamten Bestellvorgangs nur mit IDs arbeiten.