

parseSiContactsDropdown Liefert den HTML-Code eines Dropdown-Feldes zurück, welches alle beteiligten Personen eines konkreten Security Incidents enthält, wobei die Contact-IDs als Wert und die Namen der Personen als Bezeichnung fungieren. Dabei wird der Eintrag, dessen ID mit dem Parameter `$active` übereinstimmt, als `selected` markiert. Dient als Adressbuch der Prozessbeteiligten eines Security Incidents.

updateState Aktualisiert den Status eines Incidents nach Plausibilitätsprüfung des übermittelten Zeitstempels und berechnet die Dauer der aktuellen Prozessphase als Gesamtzeit und während der Servicezeiten unter Anwendung der Funktion `diffBusinessHours`. Die Funktion liefert im Erfolgsfall `TRUE` zurück. Im Fehlerfall erfolgt eine Umleitung zur zuletzt besuchten Seite unter Angabe einer entsprechenden Fehlermeldung.

6.1.4. Cronjobs

Zusätzlich zu den benutzergesteuerten Programmabläufen sind folgende Aktionen regelmäßig und automatisiert auszuführen:

- Synchronisation der `users`-Tabelle mit dem LDAP-Verzeichnisdienst
- Benachrichtigung des SICs, wenn sich ein Incident seit mindestens fünf Tagen in der gleichen Prozessphase befindet
- Benachrichtigung des SICs, wenn sich ein Incident seit mindestens 14 Tagen in der Monitoringphase befindet

Die Aktionen sollen einmal täglich durchgeführt werden. Da der exakte Zeitpunkt der Skriptausführung nicht relevant ist, wird auf die Nutzung der Linux Crontab verzichtet. Stattdessen wird ein einfacher Cron-Dienst innerhalb der Webanwendung implementiert.

Da davon auszugehen ist, dass die Login-Seite der Webanwendung täglich aufgerufen wird, wird das Cron-Skript am Anfang der Datei `index.php` (vgl. Listing 6.2) positioniert. Selbst wenn die Anwendung nicht täglich gestartet werden sollte, so erfolgt die LDAP-Synchronisation spätestens unmittelbar vor dem nächsten Login. Dadurch ist sichergestellt, dass die Personeninformationen stets auf dem aktuellen Stand gehalten werden.

Listing 6.2: Cron-Skript auf der Login-Seite (`/index.php`)

```

1 if (strtotime(getSetting($db, "CRON_LAST_RUN")) < strtotime(date("Y-m-d", time()
   ))) {
2     include($appConfig['path'] . "/inc/cron.php");
3     setSetting($db, "CRON_LAST_RUN", date("Y-m-d", time()));
4 }

```

Beim Aufruf der Datei `index.php` wird in Zeile 1 überprüft, ob das letzte Ausführungsdatum des Cronjobs kleiner als das aktuelle Datum ist. Liefert diese if-Abfrage `TRUE` zurück, so wurde der Cronjob an diesem Tag noch nicht durchgeführt. Folglich wird in Zeile 2 die Cronjob-Datei `/inc/cron.php` geladen und ausgeführt. Diese beinhaltet die Befehle zur Erledigung der oben genannten Aufgaben. Auf die Implementierung der LDAP-Synchronisation wird im nächsten Abschnitt (vgl. Listing 6.5) eingegangen. Abschließend wird in Zeile 3 das in der `settings`-Tabelle gespeicherte Ausführungsdatum aktualisiert. Ergibt die if-Abfrage hingegen `FALSE`, so wurde der Cronjob an diesem Tag bereits ausgeführt und deshalb nun übersprungen.

6.2. Technische Grundlagen

Im Folgenden werden die technischen Grundlagen der Webanwendung hinsichtlich Authentifizierung, IT-Sicherheit, Userinterface, Prozessunterstützung und Kommunikation anhand von Codebeispielen erläutert. Zunächst werden jedoch die Systemvoraussetzungen definiert und die Installation der Webanwendung beschrieben.

6.2.1. Systemvoraussetzungen und Setup

Um den Betrieb der Anwendung auf einem LRZ-Standardwebserver gewährleisten zu können, werden die für den Betrieb der webbasierten Security Incident Management Software erforderlichen technischen Voraussetzungen dementsprechend spezifiziert:

- Linux-Webserver mit PHP Version 5.3.14
- Datenbankserver MySQL Version 5.1.73
- LDAP-Server zur Nutzerauthentifizierung und Kontaktdatenabfrage
- Mailserver zum Mailversand via SMTP

Da die Inbetriebnahme der Webanwendung in wenigen Schritten manuell durch einen Administrator durchgeführt werden kann und keine technischen Spezialkenntnisse erfordert, wird auf die Implementierung eines webbasierten Installationsassistenten verzichtet. Während der Installation der Webanwendung sind zunächst die Datenbanktabellen gemäß Anhang B anzulegen. Anschließend ist die Typentabelle gemäß Anhang C mit den Standarddatensätzen zu initialisieren. Das Datenbank-Setup erfolgt durch den Import von zwei SQL-Dateien, die sich im Ordner `/inc` befinden. Hierfür wird die Verwendung des Administrationstools `phpMyAdmin` empfohlen. Nach der Vorbereitung der Datenbank sind nur noch die Konfigurationsparameter in der Datei `/inc/config.php` anzupassen. Dort sind beispielsweise die Zugangsdaten für Datenbank, LDAP und SMTP einzutragen. Abschließend ist die Webanwendung via FTP auf dem Webserver bereitzustellen.

6.2.2. Benutzerverwaltung, Authentifizierung und Rechtevergabe

Da der LDAP-Verzeichnisdienst des LRZ nicht nur die Kontaktdaten aller LRZ-Mitarbeiter bereitstellt, sondern auch zur Nutzerauthentifizierung verwendet werden kann, wird bei der Implementierung mehrfach auf die Klasse `adLDAP` zurückgegriffen.

Um sich gegenüber der Webanwendung zu authentifizieren, trägt der Benutzer auf der Startseite (`/index.php`) seine LRZ-Kennung und das zugehörige Passwort ein. Mit dem Absenden des Formulars werden die Zugangsdaten via `POST`-Request an das Login-Modul (`/actions/login.php`) übergeben. Dieses prüft zunächst, ob die übermittelten Daten mit den in der Konfigurationsdatei gespeicherten Zugangsdaten des Root-Users übereinstimmen. Trifft dies nicht zu, so wird eine LDAP-Authentifizierung in die Wege geleitet. Dieser Abschnitt des Login-Moduls (vgl. Listing 6.3) wird im Folgenden näher betrachtet:

Listing 6.3: Nutzerauthentifizierung via LDAP (/inc/login.php)

```

1 elseif ($adldap->user()->authenticate($_POST["kennung"].$ldapSuffix, $_POST["
2     pwd"])){
3     $query = "SELECT * FROM ".DBPREFIX."users WHERE kennung = '". $_POST["kennung
4     "]."'";
5     $stmt = $db->query($query);
6     if ($stmt->rowCount() == 1) {
7         $userData = $stmt->fetch(PDO::FETCH_ASSOC);
8         $userid = $userData["id"];
9         writeSyslog($db, $userid, NULL, 'login', 'OK', $_POST["kennung"]. ' login
10        successful ');
11    }
12    else {
13        writeSyslog($db, 0, NULL, 'login', 'ERROR', $_POST["kennung"]. ' LDAP Login
14        fehlgeschlagen. User nicht in DB. ');
15        header("Location: ".$appConfig['url']."/index.php?msg=5");
16        exit;
17    }
18
19    if ($adldap->user()->inGroup($userData["kennung"], $ldapGroupNameCsirt)) {
20        $userRole = "CSIRT";}
21    elseif ($adldap->user()->inGroup($userData["kennung"], $ldapGroupNameUser)) {
22        $userRole = "USER";}
23    else {
24        writeSyslog($db, 0, NULL, 'login', 'ERROR', $_POST["kennung"]. ' LDAP Login
25        fehlgeschlagen. Ungültige Gruppenzugehörigkeit. ');
26        header("Location: ".$appConfig['url']."/index.php?msg=5");
27        exit;
28    }
29
30    $sql = "UPDATE ".DBPREFIX."users SET loginStamp=?, ip=?, userRole=? WHERE id
31    =?";
32    $query = $db->prepare($sql);
33    $query->execute(array($stamp, $ip, $userRole, $userid));
34
35    $auth = TRUE;
36 }

```

In Zeile 1 werden sowohl die Kennung als auch das Passwort des Nutzers an die `authenticate` Methode des `adLDAP`-Objekts übergeben. Diese liefert bei erfolgreicher Authentifizierung `TRUE` zurück, sodass die Anweisungen in den Zeilen 3 bis 28 ausgeführt werden. In den Zeilen 3 und 4 wird anhand der der LRZ-Kennung der passende Datensatz aus der `users`-Tabelle abgerufen. Enthält die SQL-Abfrage genau einen Datensatz, so werden die Nutzerdaten für die weitere Verwendung lokal gespeichert. Der erfolgreiche Login wird durch die `writeSyslog` Funktion im Systemlog dokumentiert. Der Fall, für einen authentifizierten Benutzer in der `users`-Tabelle kein Eintrag existiert (Zeile 10), dürfte aufgrund der nachfolgend beschriebenen Synchronisation nicht auftreten. Dennoch wird dieser Fehler in den Zeilen 11 bis 13 behandelt.

Da die Rechtevergabe innerhalb der Webanwendung statisch auf Basis der Benutzerrolle erfolgt, hätte eine fehlerhafte Zuweisung erhebliche Auswirkungen auf die Informationssicherheit. Daher verlässt sich die Webanwendung nicht auf die in Zeile 3 bereits aus der `users`-Tabelle abgefragte `userRole`. Obwohl eine Manipulation der Benutzerrollen in der

6. Implementierung

Datenbank sehr unwahrscheinlich ist, aber dennoch nicht vollständig ausgeschlossen werden kann, wird die Benutzerrolle in Echtzeit über den LDAP-Verzeichnisdienst ermittelt. Unter Anwendung der `inGroup`-Methode des `adLDAP`-Objekts wird im Rahmen eines `if`-Konstrukts zunächst auf Mitgliedschaft des Benutzers in der CSIRT-LDAP-Gruppe (Zeile 16) und anschließend auf Mitgliedschaft in der LDAP-Gruppe der LRZ-Mitarbeiter (Zeile 17) geprüft. Ist der authentifizierte Benutzer weder ein CSIRT-Mitglied noch ein LRZ-Mitarbeiter, so wird ihm der Zugriff zur Webanwendung verwehrt (Zeilen 19 bis 21).

Abschließend wird in den Zeilen 24 bis 26 die IP-Adresse des Anwenders, seine `userRole` und der Login-Zeitstempel in der `users`-Tabelle aktualisiert. Der erfolgreiche Abschluss der Authentifizierung wird systemintern durch Setzen der `$auth`-Variable repräsentiert (Zeile 28).

Falls die die Authentifizierung fehlgeschlagen ist, so wird der Anwender unter Angabe einer entsprechenden Fehlermeldung zur Login-Seite zurückgeleitet. Zudem wird die Benutzerkennung des gescheiterten Loginversuchs im Systemlog dokumentiert. Ist hingegen das `$auth`-Flag gesetzt, so werden abschließend noch allgemeine Login-Operationen durchgeführt. Dazu zählt die Belegung von Session-Variablen mit den Benutzerdaten, die Generierung des Session-Tokens sowie die Umleitung zum Dashboard.

Beim Aufruf des Dashboards wird durch den Funktionsaufruf `checkUser(USER,CSIRT,ROOT)` sichergestellt, dass entweder ein LRZ-Mitarbeiter, ein CSIRT-Mitglied oder der Root-User eingeloggt ist und seine Sitzung noch nicht abgelaufen ist. Die `checkUser` Funktion schützt sowohl die Benutzeroberfläche als auch die in den Modulen enthaltene Anwendungslogik vor unberechtigten Zugriffen. Jedoch stößt die gruppenbasierte Rechtevergabe bei der Incident-statusseite an ihre Grenzen, da diese neben den CSIRT-Mitgliedern auch von allen an einem Incident beteiligten Personen unabhängig von ihrer Benutzerrolle aufrufbar sein soll. Für diesen Fall wird eine dynamische Rechtevergabe implementiert:

Listing 6.4: Dynamische Rechtevergabe (`/inc/si-overview.php`)

```
1 if (!((getContactRole($db,$_SESSION['userKennung'],$GET['si']) != FALSE) OR
2   checkUser("CSIRT,ROOT"))){
3   showAlert('error','Sie besitzen nicht die erforderlichen Rechte, um den
4     Vorgang auszuführen.');
```

```
5 }
```

Die `if`-Bedingung in Listing 6.4 besteht im Wesentlichen aus der Negation einer `OR`-Verknüpfung von Boole'schen Werten. Im ersten Teil wird anhand der `getContactRole` Funktion geprüft, ob der Anwender am Incident beteiligt ist. Handelt es sich beim Anwender beispielsweise um den Vorfalldmelder, wird der Vergleich `'MELDER' != FALSE` zu `TRUE` ausgewertet, sodass der zweite Teil der `OR`-Verknüpfung übersprungen wird. Damit alle CSIRT-Mitglieder und der Root-User auch unabhängig von ihrer konkreten Prozessbeteiligung die Statusseite einsehen können, liefert die `checkUser` Funktion für diese beiden Benutzergruppen `TRUE` zurück. Da der `if`-Clause den Negativfall – also den Zugriff eines Anwenders, der weder am Vorfall beteiligt noch CSIRT- oder Root-User ist – behandelt, wird der gesamte Ausdruck negiert.

Möchte sich der Anwender vom System abmelden, so ruft er das Logout-Modul (`/actions/logout.php`) auf. Dieses dokumentiert zunächst den Vorgang im Systemlog (`writeSyslog`) und löscht daraufhin mit der Funktion `session_unset` alle Session-Variablen, die gegenwärtig

registriert sind. Nach der Beendigung der Session mit der Funktion `session_destroy` wird der Anwender zur Loginseite umgeleitet. Der Aufruf von geschützten Seiten ist nun mangels Session nicht mehr möglich und würde einen erneuten Login erfordern.

Was die Benutzerverwaltung anbelangt, so wird aus Performancegründen darauf verzichtet, für jeden Zugriff auf Benutzerdaten eine LDAP-Abfrage in Echtzeit durchzuführen. Stattdessen werden die Personen- und Kontaktinformationen von allen LRZ-Mitarbeitern und CSIRT-Mitgliedern nur einmal täglich und in einem Vorgang vom Verzeichnisdienst abgerufen und in der `users`-Tabelle gespeichert. Dadurch wird einerseits die Belastung des LDAP-Dienstes durch die Webanwendung auf ein Minimum reduziert und andererseits stehen dem Root-User bei einem LDAP-Ausfall dennoch die Kontaktdaten aller Kollegen zur Verfügung, die er insbesondere bei der Zusammenstellung eines Security Incident Teams benötigen würde. Die LDAP-Synchronisation erfolgt im Rahmen des auf der Login-Seite implementierten Cronjobs (vgl. Listing 6.5) automatisch maximal einmal täglich. Nach dem Laden der Datei `/inc/ldap.php`, in der das `$adldap`-Objekt erstellt wird, baut die lesende LDAP-Kennung in Zeile 3 eine Verbindung zum Verzeichnisdienst auf. Kann die Verbindung nicht aufgebaut werden, so erfolgt in den Zeilen 6 bis 12 die Fehlerbehandlung. Bei der folgenden Synchronisierung wird insofern auf die Reihenfolge geachtet, als dass zunächst die LRZ-Mitarbeiter und erst dann die CSIRT-Mitglieder aktualisiert werden, weil die für die Berechtigungsvergabe relevante Benutzerrolle anhand der Gruppenzugehörigkeit festgelegt wird. Da jedes CSIRT-Mitglied auch der Gruppe der LRZ-Mitarbeiter angehört, wird in den Zeilen 44 bis 48 zuerst die allgemeinere Gruppe der LRZ-Mitarbeiter und anschließend die speziellere CSIRT-Gruppe aktualisiert.

Zur Synchronisation wird die in den Zeilen 14 bis 42 implementierte Funktion `ldapSync` aufgerufen. Als Eingabeparameter werden neben dem Datenbankhandler und dem `adLDAP`-Objekt die mit der `adLDAP`-Klasse abgefragten Mitglieder der jeweiligen Gruppe sowie die zugehörige Benutzerrolle übergeben. Bei dieser Funktion handelt es sich im Wesentlichen um eine große `foreach`-Schleife (Zeilen 20 bis 40), die über alle übergebenen Gruppenmitglieder iteriert. Dabei werden in Zeile 21 zunächst die Kontaktdaten der jeweils aktuellen Person aus dem LDAP-Verzeichnis abgerufen. Anschließend wird in den Zeilen 23 bis 25 der korrespondierende Eintrag anhand der LRZ-Kennung aus der `users`-Tabelle der Datenbank selektiert. Liefert die Datenbankabfrage kein Ergebnis zurück (Zeile 27), so handelt es sich um einen neuen Mitarbeiter, für den in der `users`-Tabelle ein neuer Datensatz angelegt wird (Zeilen 28 bis 30). Ergibt die Datenbankabfrage ein Ergebnis (Zeile 33), so wird dieser Datensatz mit den LDAP-Werten überschrieben (Zeilen 34 bis 37). Sowohl bei der `INSERT`- als auch bei der `UPDATE`-Operation auf der Datenbank wird die `userRole` gemäß Inputparameter gesetzt. Darüber hinaus wird das Feld `lastLdapUpdate` aktualisiert, sodass die Webanwendung unterscheiden kann, ob es sich um einen aktuellen Datensatz oder um einen ausgeschiedenen Mitarbeiter handelt. Die in der Funktion verwendeten Counter geben am Ende Auskunft über die Gesamtzahl der verarbeiteten Datensätze sowie die Anzahl der neu angelegten und aktualisierten Benutzer. Der Rückgabewert der Funktion `ldapSync` enthält die Counter-Informationen als String und wird im Systemlog gespeichert.

Betrachtet man abschließend ein CSIRT-Mitglied, so wird dieses beim ersten Aufruf der Funktion `ldapSync` als `USER` in der Datenbank hinterlegt, da alle CSIRT-Mitglieder auch der Gruppe der LRZ-Mitarbeiter angehören. Beim zweiten Aufruf der `ldapSync` Funktion wird ihm die speziellere Benutzerrolle `CSIRT` zugewiesen. Dementsprechend ist bei einer späteren Implementierung weiterer Benutzergruppen insbesondere auf die Reihenfolge während der LDAP-Synchronisation zu achten.

Listing 6.5: LDAP-Synchronisation (/inc/cron.php)

```

1 require_once($appConfig['path'].'/inc/ldap.php');
2
3 if($adldap->user()->authenticate($ldapUser.$ldapSuffix,$ldapPwd)) {
4     writeSyslog($db,NULL,NULL,'ldapSync','OK','Login for Sync ok.');
```

```

5 }
6 else {
7     writeSyslog($db,NULL,NULL,'ldapSync','ERROR','Login for Sync failed.');
```

```

8     $adldap->close();
9     $db = null;
10    header("Location: ".$appConfig['url'].'/index.php?msg=5");
11    exit;
12 }
13
14 function ldapSync($db,$adldap,$group,$userRole){
15     $updateCounter = 0;
16     $insertCounter = 0;
17     $generalCounter = 0;
18     $stamp = date("Y-m-d",time());
19
20     foreach($group as $item){
21         $user = $adldap->user()->infoCollection($item, array('cn','sn','givenname',
22             'telephonenumber','mail'));
23
24         $query = "SELECT id FROM ".$DBPREFIX."users WHERE kennung = '$user->cn'";
25         $stmt = $db->query($query);
26         $result = $stmt->fetch(PDO::FETCHASSOC);
27
28         if (!$result) { // Neuer User -> anlegen
29             $sql = "INSERT INTO ".$DBPREFIX."users (userRole,kennung,name,vorname,
30                 email,tel,lastLdapUpdate) VALUES (?,?,?,?,?,?,?)";
31             $query = $db->prepare($sql);
32             $query->execute(array($userRole,$user->cn,$user->sn,$user->givenname,
33                 $user->mail,$user->telephonenumber,$stamp));
34             $insertCounter++;
35         }
36         else { // User existiert -> Update
37             $sql = "UPDATE ".$DBPREFIX."users SET userRole=?, name=?, vorname=?,
38                 email=?, tel=?, lastLdapUpdate=? WHERE kennung=?";
39             $query = $db->prepare($sql);
40             $query->execute(array($userRole,$user->sn,$user->givenname,$user->mail,
41                 $user->telephonenumber,$stamp,$user->cn));
42             $updateCounter++;
43         }
44         $generalCounter++;
45     }
46     return "$userRole / $generalCounter Items / updated: $updateCounter / new:
47         $insertCounter";
48 }
49
50 $syncUsers = ldapSync($db,$adldap,$adldap->group()->members(
51     $ldapGroupNameUser),"USER");
52 writeSyslog($db,NULL,NULL,'ldapSync','OK',$syncUsers);
53
54 $syncCsirt = ldapSync($db,$adldap,$adldap->group()->members(
55     $ldapGroupNameCsirt),"CSIRT");
56 writeSyslog($db,NULL,NULL,'ldapSync','OK',$syncCsirt);
57
58 $adldap->close();

```

6.2.3. IT-Sicherheit

Die im Konzept beschriebenen Aspekte der IT-Sicherheit werden – soweit dies im Rahmen der Entwicklungsumgebung möglich ist – bei der Implementierung berücksichtigt. Es wird beispielsweise grundsätzlich darauf geachtet, dass keine schützenswerten Daten als GET-Parameter in der URL übertragen werden. Da die Implementierung von Benutzerauthentifikation und Sitzungsverwaltung bereits im vorigen Abschnitt ausführlich beschrieben wurde, wird im Folgenden die Umsetzung weiterer Sicherheitsaspekte betrachtet.

Obwohl die Entwicklungsumgebung nicht über ein SSL-Zertifikat verfügt, wird dennoch das HTTPS-Protokoll verwendet. Die Datei `/inc/basics.php` erzwingt eine verschlüsselte Verbindung, sodass HTTP-Aufrufe automatisch in HTTPS-Requests umgewandelt werden. Wie aus Listing 6.6 ersichtlich ist, wird dabei in der ersten Zeile anhand der PHP-Servervariable geprüft, ob *keine* HTTPS-Verbindung besteht. Ergibt die Auswertung der if-Abfrage `FALSE`, besteht bereits eine verschlüsselte Verbindung, sodass diesbezüglich keine Aktionen erforderlich sind. Liefert die if-Abfrage hingegen ein `TRUE` zurück, so ist eine Umleitung der HTTP-Adresse auf die korrespondierende HTTPS-Adresse erforderlich. Hierfür wird in Zeile 3 die HTTPS-URL zusammengesetzt. In den Zeilen 5 bis 7 werden eventuell vorhandene GET-Parameter an die neue URL angehängt. Schließlich wird in Zeile 10 der Aufbau einer verschlüsselten Verbindung durch Umleitung auf die zuvor generierte HTTPS-URL in die Wege geleitet, sofern bislang noch keine anderen Header gesendet wurden.

Listing 6.6: Erzwingung einer HTTPS-Verbindung (`/inc/basics.php`)

```

1  if (!(isset($_SERVER['HTTPS']) && ($_SERVER['HTTPS'] == '1' || strtolower(
2     $_SERVER['HTTPS'])=='on'))){
3
4     $url = 'https://' . $_SERVER['HTTP_HOST'] . $_SERVER['PHP_SELF'];
5
6     if (!empty($_SERVER['QUERY_STRING'])) {
7         $url .= '?' . $_SERVER['QUERY_STRING'];
8     }
9
10    if (!headers_sent()) {
11        header('Location: ' . $url);
12    }

```

Um das Aufrufen von Systemdateien außerhalb des zulässigen Kontexts und das Einbinden dieser in fremde Skripte zu verhindern, werden die Verzeichnisse `/inc`, `/tmp` und `/tpl` mit einem `.htaccess`-Verzeichnisschutz versehen. Dadurch werden externe Direktzugriffe u.a. auf die Konfigurationsdatei, die Funktionsbibliotheken und die HTML-Templates sowie deren temporäre Cachingdateien unterbunden. Listing 6.7 zeigt, wie sich die Verzeichnisinhalte durch den Einsatz von `.htaccess`-Dateien vor externen Zugriffen schützen lassen. Da sich die `DENY FORM ALL` Anweisung nur auf externe Zugriffe bezieht, können interne Skripte der Webanwendung weiterhin auf die geschützten Dateien zugreifen.

Listing 6.7: `.htaccess`-Verzeichnisschutz

```

1  order deny, allow
2  deny from all

```

6. Implementierung

Zur Reduzierung der Sicherheitsrisiken von Cross-Site-Scripting (XSS) und (SQL)-Injections werden alle Benutzereingaben einer serverseitigen Filterung unterzogen. Die Implementierung dieser Sicherheitsmaßnahme (vgl. Listing 6.8) erfolgt vor der Weiterverarbeitung der unbehandelten Daten an zentraler Stelle in der Datei `/inc/basics.php`. Da die Benutzereingaben seitens PHP im `POST`-Array und `GET`-Array gespeichert werden, werden beide Arrays mit der Standard-PHP-Funktion `filter_input_array` rekursiv durchlaufen. Der dabei angewandte `FILTER_SANITIZE_STRING` Filter macht die Benutzereingaben unschädlich, indem er potenziell gefährliche Zeichenfolgen entfernt und Sonderzeichen wie Hochkommata maskiert. Die gefilterten Daten werden in die ursprünglichen Arrays zurückgespeichert und können nun komfortabel und gefahrlos weiterverarbeitet werden.

Listing 6.8: Serverseitige Eingabefilterung

```
1 $_GET = filter_input_array(INPUT_GET, FILTER_SANITIZE_STRING);
2 $_POST = filter_input_array(INPUT_POST, FILTER_SANITIZE_STRING);
```

Darüber hinaus wird zum Schutz der Anwendungslogik jede Aktion mit einem Session-Token autorisiert. Listing 6.9 fasst die relevanten Codeabschnitte zusammen. Da sich der Token von der dem Browser bekannten Session-ID unterscheiden soll, wird er während des Loginvorgangs entsprechend der ersten Zeile als 32-stelliger MD5-Hash einer eindeutigen Zufalls-ID generiert. Er wird in einer Session-Variable gespeichert und bleibt für die Dauer der Session unverändert. Um bei der Übermittlung eines Formulars die Korrektheit des Tokens prüfen zu können, muss dieses mit den Formulardaten übergeben werden. Zu diesem Zweck enthält jedes Formular ein verstecktes Token-Feld entsprechend der Darstellung in Zeile 3. Die Moduldatei, welche die abgesendeten Formulardaten als `POST`-Array empfängt, ruft in Zeile 5 die Funktion `checkToken` auf und übergibt den empfangenen Token als einzigen Parameter an die Funktion. Diese vergleicht in Zeile 8 den übergebenen Token mit dem in einer Session-Variable gespeicherten Token. Stimmen die beiden Token überein, gibt die Funktion `TRUE` zurück. Falls nicht, wird in Zeile 10 mit der Funktion `setAlert` eine Fehlermeldung gesetzt. In Zeile 11 wird der Anwender zur zuletzt aufgerufenen Seite umgeleitet, deren Adresse der Session-Variable `lastUrl` entnommen wird, woraufhin die Ausführung des Skripts beendet wird. Der Token bleibt so lange erhalten, bis die Session abläuft oder die Sitzung während des Logoutvorgangs – wie in den Zeilen 17 und 18 dargestellt – geschlossen wird.

Listing 6.9: Implementierung des Session-Tokens

```
1 $_SESSION['token'] = md5(uniqid(rand(), true));
2
3 <input name="token" value="{ $token }" type="hidden" />
4
5 checkToken($_REQUEST['token']);
6
7 function checkToken($token){
8     if ($token == $_SESSION['token']) {return TRUE;}
9     else {
10         setAlert('error', 'Der Token ist ungültig. ');
11         header("Location: " . $_SESSION['lastUrl']);
12         exit;
13         return FALSE;
14     }
15 }
16
17 session_unset();
18 session_destroy();
```

Da die Webanwendung aus Gründen der Informationssicherheit nicht von mobilen Endgeräten aus bedient werden soll, wird am Anfang der Login-Seite `index.php` (vgl. Listing 6.10) ein Mechanismus zur Erkennung von Smartphones und Tablets implementiert. Hierfür wird zunächst die Klasse `mobileDetect`¹ geladen und ein neues Objekt dieser Klasse erstellt. Die `if`-Abfrage in Zeile 3 prüft, ob der Seitenaufruf durch ein mobiles Gerät erfolgt ist. Liefert diese `TRUE` zurück, so erfolgt in Zeile 4 eine Umleitung ins Verzeichnis „mobile“. Zeile 5 beendet die Ausführung der PHP-Datei. Die am Endgerät dargestellte Datei `/mobile/index.php` informiert den Nutzer über den Grund der Zugriffssperre. Liefert die `if`-Abfrage `FALSE` zurück, wurde kein mobiles Endgerät erkannt, sodass mit der Ausführung der nächsten Skriptzeilen fortgefahren wird.

Listing 6.10: Unterbindung der Nutzung auf mobilen Endgeräten (`/index.php`)

```

1 require_once("inc/mobileDetect/Mobile_Detect.php");
2 $detect = new Mobile_Detect;
3 if ( $detect->isMobile() ) {
4     header("Location: /mobile");
5     exit;
6 }

```

Eine Prüfung des Endgeräts genügt auf der Login-Seite, da diese Seite vom Anwender in jedem Fall besucht werden muss, um eine Nutzersitzung initiieren zu können. Im Falle eines Direktzugriffs beispielsweise auf die Datei `dashboard.php`, stellt die Anwendung fest, dass der Zugriff mangels Session nicht autorisiert ist und leitet den Anwender zur die Login-Seite um. Dort wird dann die Endgeräteprüfung vorgenommen, woraufhin der Nutzer im Falle eines mobilen Zugriffs nicht die Login-Maske, sondern den in Abbildung 6.4 dargestellten Hinweis bekommt.



Abbildung 6.4.: Screenshot: Sperrung von mobilen Endgeräten

¹<http://mobiledetect.net/>

6.2.4. Benutzeroberfläche

Obwohl der Schwerpunkt der Arbeit nicht auf der grafischen Benutzerschnittstelle liegt, so wird dennoch eine professionelle, intuitiv bedienbare Benutzeroberfläche erwartet. Die Implementierung der im Konzept beschriebenen Aspekte erfolgt daher auf Basis einer modernen Designvorlage, für deren Nutzung eine Lizenz erworben wurde. **Ace**² ist ein leichtgewichtiges, auf Bootstrap 3 basierendes GUI-Framework für Webanwendungen, das trotz einer hohen Funktionsvielfalt eine komfortable Nutzung erlaubt. Dieses Rahmenwerk stellt zahlreiche Elemente für den Aufbau und die Gestaltung des Userinterfaces bereit. Dazu zählen neben Tabellen, Widgets und Formularen auch Icons, typografische Elemente und die Navigationsleiste. Die Benutzeroberfläche der Webanwendung besteht insgesamt aus 36 HTML-Dateien. Die Masken enthalten Platzhalter, die beim Aufruf durch die Template-Klasse **rainTPL**³ dynamisch mit Nutzdaten befüllt werden. Abbildung 6.5 zeigt die gerenderte Darstellung des Dashboard Templates (`/tpl/dashboard.html`).

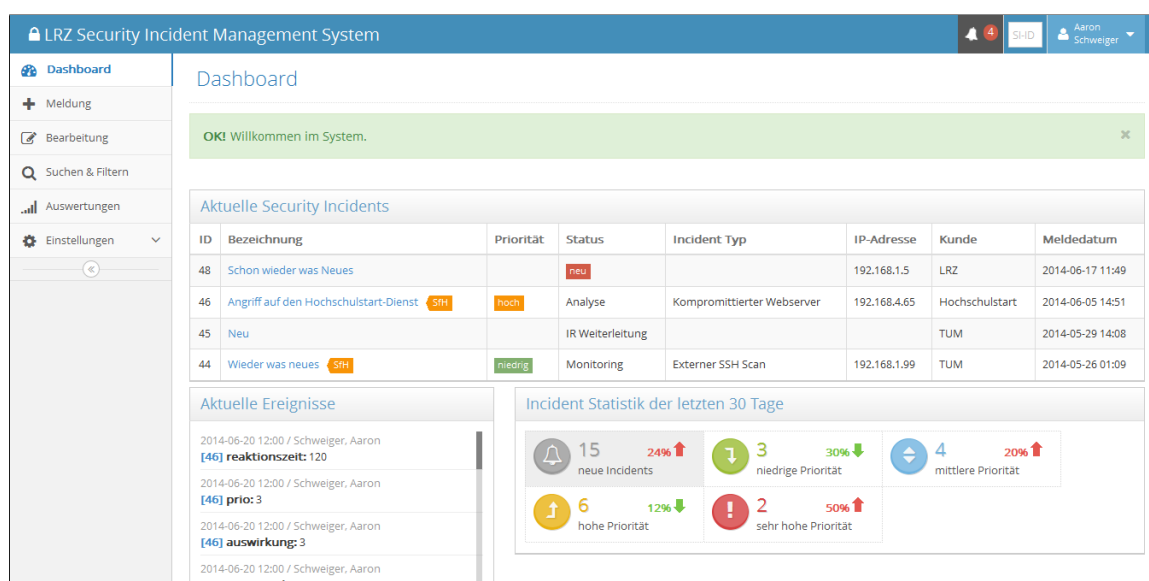


Abbildung 6.5.: Screenshot: Dashboard (`/dashboard.php`)

Wie im Konzept vorgesehen, beinhaltet die mit **Ace**-Elementen kreierte Benutzeroberfläche eine Kopfzeile, eine einklappbare Menüleiste und einen großflächigen variablen Bereich für die individuellen Bestandteile der jeweiligen Maske. Die Navigationsleiste wird in Abhängigkeit von der `userRole` des eingeloggten Anwenders und der aufgerufenen Seite in einer `for`-Schleife (`/inc/tpl.nav.php`) dynamisch erzeugt und als HTML-Code an das Template übergeben. Auch bei der Zusammenstellung der Kopfzeile wird die Benutzerrolle berücksichtigt, da die beiden Schnellzugriff-Elemente nur für die CSIRT-Mitglieder und den Root-User sichtbar sein sollen. Durch Klick auf das Glockensymbol lässt sich eine Übersicht der neuen und gerade in Bearbeitung befindlichen Vorfälle sowie deren Bearbeitungsfortschritt einblenden. Wählt man dort einen Vorfall aus, so wird man vom Incident Router (`/si-router.php`) automatisch auf die zur Prozessphase passenden Seite geleitet. Für die Routingentscheidung wird

²<https://wrapbootstrap.com/theme/ace-responsive-admin-template-WBOB30DGR>

³<http://www.raintpl.com/Documentation/>

der aktuelle Status des Vorfalls aus der Datenbank abgefragt. Mit einer switch-Anweisung wird die zum Status gehörende Zielseite ermittelt. Das in der Kopfzeile angrenzende Schnellzugriffsfeld ermöglicht das gezielte Aufrufen eines Vorfalls durch Eingabe seiner ID. Da diese Funktion zum effizienten Zugriff auf bereits abgeschlossene Vorfälle konzipiert ist, wird statt des Routers direkt die Statusseite aufgerufen.

Um den Anwender über den Erfolg oder Misserfolg der von ihm durchgeführten Aktion zu informieren, ermöglicht die Benutzeroberfläche die Ausgabe von Systemmeldungen. Die in Screenshot 6.5 grün hinterlegte Meldung signalisiert den erfolgreichen Abschluss der Login-Aktion. Derartige Systemmeldungen können durch die in Listing 6.11 abgebildete Funktion `setAlert` erstellt werden. Falls es sich um eine Erfolgsmeldung handelt, so wird in Zeile 2 die Session-Variable `$_SESSION['alert_ok']` mit dem Meldungstext initialisiert. Im Falle einer Fehlermeldung wird mit der Session-Variable `$_SESSION['alert_error']` analog verfahren.

Listing 6.11: Funktion `setAlert (/inc/functions.php)`

```

1 function setAlert($type, $text){
2     if($type == "ok") {$_SESSION['alert_ok'] = $text; return TRUE;}
3     elseif($type == "error") {$_SESSION['alert_error'] = $text; return TRUE;}
4 }

```

Beim Aufruf der nächsten Seite wird gemäß Listing 6.12 zunächst geprüft, ob die Session-Variable `$_SESSION['alert_ok']` gesetzt ist (Zeile 1). Trifft dies zu, so wird in Zeile 2 der HTML-Code mit dem Meldungstext in der lokalen Variable `$alert` gespeichert. Eine eventuelle Fehlermeldung wird in den Zeilen 5 und 6 analog behandelt. Anschließend wird der Inhalt der `$alert` Variable an das Template übergeben (Zeile 9). Abschließend werden in den Zeilen 10 und 11 durch das Rücksetzen der beiden Session-Variablen eventuell vorhandene Meldungen gelöscht, damit eine bereits angezeigte Meldung nach einem Seitenwechsel nicht erneut ausgegeben wird. Die Integration der Systemmeldungen in die Benutzeroberfläche erfolgt durch den Aufruf `{$parsedAlert}` in der Datei `/tpl/header.html`.

Listing 6.12: Ausgabe von Systemmeldungen (`/inc/tpl.alert.php`)

```

1 if (isset($_SESSION["alert_ok"])){
2     $alert .= "<div class='alert alert-block alert-success'><button class='close
3         ' data-dismiss='alert' type='button'><i class='icon-remove'></i></button><
4         strong>OK! </strong>".$_SESSION["alert_ok"]."</div>";
5 }
6 if (isset($_SESSION["alert_error"])){
7     $alert .= "<div class='alert alert-block alert-danger'><button class='close '
8         data-dismiss='alert' type='button'><i class='icon-remove'></i></button><
9         strong>Fehler! </strong>".$_SESSION["alert_error"]."</div>";
10 }
11 $tpl->assign('parsedAlert', $alert);
12 unset($_SESSION["alert_ok"]);
13 unset($_SESSION["alert_error"]);

```

6.2.5. Prozessunterstützung

Im Folgenden wird auf die Umsetzung prozessunterstützender Maßnahmen eingegangen. Diese umfassen die Sicherstellung des ordnungsgemäßen Prozessablaufs, die Zeitmessung während allen Phasen, eine manipulationssichere Dokumentation aller Aktivitäten sowie die Erweiterung der Benutzeroberfläche um eine Incident Menüleiste und eine Sidebar.

Der Wechsel in die nächste Prozessphase wird durch den Anwender ausgelöst. Hierzu sendet der Anwender das in Abbildung 6.6 dargestellte Formular ab. Standardmäßig ist der aktuelle Zeitstempel hinterlegt. Dieser kann jedoch im Falle einer Nachtragung geändert werden.

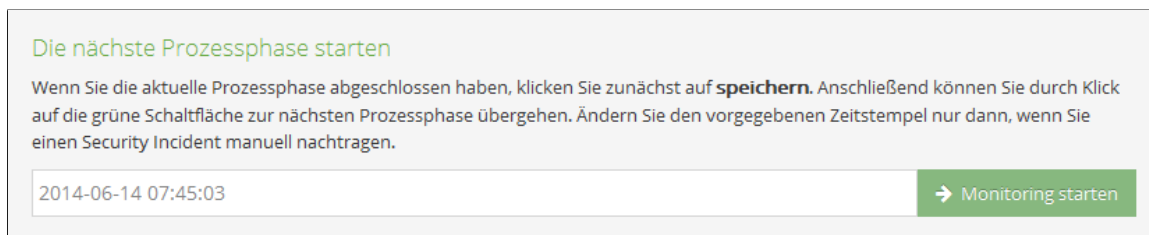


Abbildung 6.6.: Screenshot: Statuswechsel

Durch Klick auf den grünen Button wird der Zeitstempel an die Aktion `updateState` (vgl. Listing 6.13) übergeben. In Zeile 1 wird durch Aufruf der `checkState` Funktion sichergestellt, dass diese Aktion nur ausgeführt werden kann, wenn sich der Vorfall entweder den Status Klassifikation, Analyse, Lösung oder Monitoring aufweist. Anschließend wird in Zeile 2 die `updateState` Funktion aufgerufen, die den Wechsel der Prozessphase durch Änderung des Incidentstatus vornimmt.

Listing 6.13: Aktion `updateState` (`/actions/incidents.php`)

```
1 checkState($db, $_POST[ 'si' ], "CLASS,ANAL,SOL,MON" );  
2 updateState($db, $_POST[ 'si' ], $_POST[ 'stamp' ] );
```

Durch die Auslagerung des Statuswechsels in eine separate Funktion können auch andere Aktionen – wie beispielsweise `classifyIncident`, `closeIncident` und `reopenIncident` – den Übergang in die nächste Prozessphase herbeiführen. Diese für die Anwendungslogik sehr zentrale Funktion `updateState` (vgl. Listing 6.14) erwartet als Inputparameter lediglich den Datenbankhandler, die ID des Vorfalls sowie den Zeitstempel. Um eine potenzielle Manipulation der Anwendungslogik zu verhindern, ist weder die Übergabe des aktuellen noch des Folgestatus möglich. Stattdessen ermittelt die Funktion in den Zeilen 3 und 4 den aktuellen Status und den Beginn der aktuellen Prozessphase durch Anwendung der `getIncidentValue` Funktion aus der Datenbank. In den Zeilen 7 bis 21 findet eine Fehlerbehandlung hinsichtlich des übergebenen Zeitstempels statt. Ist der Zeitstempel formal nicht korrekt oder kleiner als der Zeitstempel zu Beginn der Prozessphase, so wird die Ausführung der Funktion beendet, wobei der Anwender unter Angabe einer Fehlermeldung zur vorigen Seite zurückgeleitet wird. Der Switch-Block (Zeilen 23 bis 36) definiert in Abhängigkeit vom aktuellen Incidentstatus den nächsten Status `$newState` gemäß Prozessmodell, die zugehörige Zielseite `$nextFeature` sowie den Zeitzähler `$saldoId`, dem die aktuelle Prozessphase zuzurechnen ist. Dabei entspricht `saldo1` der Reaktionszeit, `saldo2` der Analysezeit, `saldo3` der Lösungszeit und `saldo4` der Nachlaufzeit.

Listing 6.14: Funktion updateState (/inc/functions.si.php)

```

1 function updateState($db, $si, $stamp){
2     global $appConfig;
3     $currentState = getIncidentValue($db, $si, "currentState");
4     $currentStamp = getIncidentValue($db, $si, "currentStamp");
5     if ($stamp == FALSE){$stamp = date("Y-m-d H:i:s", time());}
6
7     if(strtotime($stamp) == FALSE){
8         $db = null;
9         showAlert('error', "Der Status wurde nicht aktualisiert, da der Zeitstempel
10        ungültig ist.");
11        header("Location: ".$_SESSION['lastUrl']);
12        exit;
13        return FALSE;
14    }
15
16    if (strtotime($stamp) <= strtotime($currentStamp)) {
17        $db = null;
18        showAlert('error', "Der Status wurde nicht aktualisiert, da der Zeitstempel
19        größer als zu Beginn der letzten Prozessphase sein muss.");
20        header("Location: ".$_SESSION['lastUrl']);
21        exit;
22        return FALSE;
23    }
24
25    switch ($currentState){
26        case "NEW": $newState = "CLASS"; $saldoId = "saldo1";
27        $nextFeature = "si-data"; break;
28        case "CLASS": $newState = "ANAL"; $saldoId = "saldo1";
29        $nextFeature = "si-tasks"; break;
30        case "ANAL": $newState = "SOL"; $saldoId = "saldo2";
31        $nextFeature = "si-solution"; break;
32        case "SOL": $newState = "MON"; $saldoId = "saldo3";
33        $nextFeature = "si-monitoring"; break;
34        case "MON": $newState = "CLOSED"; $saldoId = "saldo4";
35        $nextFeature = "si-overview"; break;
36        case "CLOSED": $newState = "ANAL"; $saldoId = "saldo4";
37        $nextFeature = "si-overview"; break;
38    }
39
40    $query = "SELECT currentStamp, $saldoId, ".$saldoId.".bh FROM ".$DBPREFIX."
41    incidents WHERE id = '$si'";
42    $stmt = $db->query($query);
43    $old = $stmt->fetch(PDO::FETCH_ASSOC);
44
45    if($currentState == "CLOSED") {
46        $diff = 0;
47        $diffMin = 0;
48        $newSaldo = $old[$saldoId];
49        $diffBh = 0;
50        $diffMinBh = 0;
51        $newSaldoBh = $old[$saldoId.'.bh'];
52    }
53    else {
54        $diff = strtotime($stamp) - strtotime($old["currentStamp"]);
55        $diffMin = round($diff/60); // in Minuten
56        $newSaldo = $old[$saldoId] + $diffMin; // in Minuten

```

6. Implementierung

```
54
55     $diffBh = diffBusinessHours($old["currentStamp"], $stamp);
56     $diffMinBh = round($diffBh/60); // in Minuten
57     $newSaldoBh = $old[$saldoId.'bh'] + $diffMinBh; // in Minuten
58 }
59
60 $sql = "UPDATE ".$DBPREFIX."incidents SET currentState=?, currentStamp=?,
61     $saldoId=?, ".$saldoId."bh=? WHERE id=?";
62 $query = $db->prepare($sql);
63 $query->execute(array($newState, $stamp, $newSaldo, $newSaldoBh, $si));
64 writeSyslog($db, $_SESSION["userId"], $si, "updateState", "OK", "currentState ->
65     $newState / $saldoId / $old[$saldoId] -> $newSaldo ($diff)");
66
67 insertHistory($db, $si, $_SESSION["userId"], "currentState", $currentState,
68     $newState, $_SESSION["userName"].", ".$_SESSION["userVorname"]);
```

Da das System sowohl die tatsächliche Dauer jeder Prozessphase als auch die Dauer während der Servicezeiten berechnen soll, werden in den Zeilen 38 bis 40 die bisherigen Zeitwerte aus der Datenbank geladen. In den Zeilen 42 bis 58 findet die Berechnung der Zeitdifferenz zwischen dem übergebenen Zeitstempel und dem Zeitpunkt zu Beginn der Prozessphase statt. Falls der Incident aktuell geschlossen ist (Zeile 42), bewirkt die `updateState` Funktion das Wiedereröffnen des Vorfalls und das Zurücksetzen in die Analysephase. In diesem Fall ist jedoch keine Zeitberechnung erforderlich, da der Zeitraum, in dem der Vorfall geschlossen war, für die Performancemessung irrelevant ist, sodass die Zeitdifferenzen auf Null gesetzt werden und der neue Saldo dem alten entspricht. In allen anderen Fällen (Zeilen 50 bis 58) wird zunächst die gesamte Zeitdifferenz in Sekunden (Zeile 51) und Minuten (Zeile 52) berechnet. Der neue Saldo ergibt sich aus der Addition des alten Saldos mit der errechneten Differenz, sodass die Laufzeiten auch bei einem mehrfachen Durchlaufen der Prozessphasen korrekt erfasst werden. Analog dazu erfolgt die Berechnung der Zeitdifferenz während der Servicezeiten, wobei anstelle der Subtraktion die Funktion `diffBusinessHours` aufgerufen wird. Deren Vorgehensweise wird im Anschluss näher beschrieben. Nach der Berechnung der Phasenlaufzeit werden die neuen Werte in die Datenbank geschrieben (Zeilen 60 bis 62) und im Systemlog vermerkt (Zeile 63). Abschließend wird die Statusänderung durch die Funktion `insertHistory` im incidentbezogenen Verlaufsprotokoll dokumentiert.

Nachfolgend wird die Implementierung der zuvor aufgerufenen Funktion `diffBusinessHours` erläutert, die den Unterschied zwischen zwei Zeitstempeln in Sekunden berechnet, wobei nur Zeiten während der Business Hours berücksichtigt werden. Als Servicezeiten gelten die Zeiträume von Montag bis Donnerstag jeweils von 8 bis 18 Uhr und freitags von 8 bis 16 Uhr. Da es sich um ein komplexes Problem handelt, für dessen Lösung keine frei verfügbare PHP-Klasse auffindbar ist, wird ein eigener Algorithmus entwickelt. Abbildung 6.7 visualisiert die ermittelte Reaktionszeit von zwei Stunden für einen Incident, der freitags um 17 Uhr gemeldet und montags um 10 Uhr klassifiziert wird.



Abbildung 6.7.: Visualisierung `diffBusinessHours`

Die grundlegende Idee hinter dem Algorithmus besteht darin, das gesamte Zeitintervall in drei Abschnitte – Starttag, Zwischentage und Endtag – zu unterteilen und deren Dauer getrennt zu berechnen. Schließlich liefert die Addition der Subintervalle das Gesamtergebnis. Der Berechnungsalgorithmus geht allerdings davon aus, dass sowohl der Start- als auch der Endzeitpunkt innerhalb der Servicezeit liegen. Deshalb verschiebt der Algorithmus zu Beginn `$start` und `$end` an die Grenze der Servicezeit, falls dies erforderlich ist. Während der Implementierung wurde der Algorithmus zahlreichen Tests unterzogen, um zu gewährleisten, dass der Algorithmus auch Sonderfälle korrekt berechnet. Dazu zählen Zeitintervalle, die außerhalb der Servicezeit beginnen und enden, ebenso wie mehrwöchige Intervalle, die mehrere Freitage und Wochenenden beinhalten.

Die Funktion `diffBusinessHours` (vgl. Listing 6.15) setzt sich aus zahlreichen verschachtelten if-Anweisungen und einer for-Schleife zusammen. Die in den Zeilen 1 bis 4 anhand der Eingabeparameter ermittelten Wochentage werden benötigt, um den Endzeitpunkt an das Ende der vorigen Servicezeit zu verschieben (Zeilen 7 bis 21). Liegt das Ende auf einem Samstag oder Sonntag, so wird es in Zeile 7 auf den vorigen Freitag um 16 Uhr gesetzt. Nachdem in Zeile 9 die Uhrzeit des Endzeitpunkts in einen sechsstelligen String ohne Trennzeichen umgewandelt wurde, wird in den Zeilen 11 bis 14 der Fall betrachtet, dass das Zeitintervall morgens vor 8 Uhr endet. Handelt es sich um einen Montag, so wird das Ende auf 16 Uhr vor drei Tagen (also Freitag) justiert. An allen anderen Tagen wird das Ende auf 18 Uhr des Vortags korrigiert. Anschließend wird in den Zeilen 16 bis 21 der Fall betrachtet, dass der Zeitpunkt abends nach dem Ende der Servicezeit liegt. Da freitags die Servicezeit bereits um 16 Uhr endet, greift eine wochentagsabhängige Unterscheidung. Liegt ein Zeitpunkt an einem Freitag nach 16 Uhr, so wird der Zeitstempel auf 16 Uhr geändert. Handelt es sich um einen anderen Wochentag nach 18 Uhr, so erfolgt eine Rücksetzung auf 18 Uhr. Die Verschiebung des Startzeitpunkts an den Beginn der nächsten Servicezeit erfolgt analog dazu und ist in den Zeilen 23 bis 34 implementiert. Da sich durch die Justierung der Zeitpunkte die Wochentage geändert haben können, werden diese in den Zeilen 36 und 37 neu bestimmt.

Nach der eventuellen Verschiebung der Zeitpunkte an den Beginn bzw. das Ende der nächstliegenden Servicezeit erfolgt in den Zeilen 39 bis 55 die eigentliche Berechnung der Zeitdifferenz in Sekunden und wird in der Variable `$sec` gespeichert. Zunächst wird in Zeile 39 geprüft, ob das Datum von Start- und Endzeitpunkt identisch ist. Trifft dies zu, so wird die Zeitdifferenz durch eine einfache Subtraktion der Zeitpunkte berechnet. Liegen Start und Ende an verschiedenen Tagen, gestaltet sich die Berechnung aufwändiger: Zunächst wird in Zeile 41 die Anzahl der ganzen Tage zwischen Ende und Anfang berechnet und in der Variablen `$daysBetween` gespeichert. Ist diese positiv, so wird in den Zeilen 43 und 44 die Anzahl der Sekunden des Starttages durch Subtraktion der wochentagsabhängigen Serviceschlusszeit von der individuellen Startzeit ermittelt und in der Variable `$sec` gespeichert. Anschließend wird in den Zeilen 45 bis 51 die Anzahl der Sekunden der zwischen Start und Ende liegenden Tage berechnet und addiert. Die hierfür verwendete for-Schleife iteriert über alle Tage vom zweiten bis zum vorletzten Tag. Die in Zeile 45 mit der zahlenmäßigen Repräsentation des Start-Wochentags belegte Zählvariable `$day` wird bereits zu Beginn der Schleife inkrementiert (Zeile 47), sodass der zuvor berechnete erste Tag nicht mehr betrachtet wird. Verlässt die Zählvariable durch das Inkrementieren den gültigen Wertebereich (0-6), so wird diese in Zeile 48 auf 0 zurückgesetzt, da nach dem sechsten Tag (Samstag) kein siebter, sondern der nullte (Sonntag) folgt. Liegt der aktuell betrachtete Wochentag zwischen Montag (1) und Freitag (5), so wird in den Zeilen 49 und 50 die gesamte Servicezeit des Tages zur Variable `$sec` hinzuaddiert. Die Servicezeit beträgt freitags 8 Stunden (Zeile 50) und an den anderen Werktagen 10 Stunden (Zeile 49).

Listing 6.15: Funktion `diffBusinessHours (/inc/functions.php)`

```

1 function diffBusinessHours($start,$end){
2     $start = strtotime($start);
3     $startDoW = date('w',$start);
4     $end = strtotime($end);
5     $endDoW = date('w',$end);
6
7     if($endDoW == 0 || $endDoW == 6){$end = strtotime('Last Friday 4pm',$end);}
8
9     $time = date('His',$end);
10
11    if($time < "080000"){
12        if($endDoW == 1){$end = strtotime('-3days 4pm',$end);}
13        else {$end = strtotime('-1day 6pm',$end);}
14    }
15
16    if($endDoW == 5){
17        if($time > "160000"){$end = strtotime('4pm',$end);}
18    }
19    else {
20        if($time > "180000"){$end = strtotime('6pm',$end);}
21    }
22
23    if($startDoW == 0 || $startDoW == 6){$start = strtotime('Next Monday 8am',
24        $start);}
25
26    $time = date('His',$start);
27
28    if($time < "080000"){$start = strtotime('8am',$start);}
29
30    if($startDoW == 5){
31        if($time > "160000"){$start = strtotime('Next Monday 8am',$start);}
32    }
33    else {
34        if($time > "180000"){$start = strtotime('+1day 8am',$start);}
35    }
36
37    $endDoW = date('w',$end);
38    $startDoW = date('w',$start);
39
40    if (date("Y-m-d",$start) == date("Y-m-d",$end)) {$sec = $end-$start;}
41    else {
42        $daysBetween = floor(($end-$start)/(24*60*60));
43        if ($daysBetween > 0) {
44            if($startDoW == 5){$sec = strtotime('4pm',$start) - $start;}
45            else {$sec = strtotime('6pm',$start) - $start;}
46            $day = $startDoW;
47            for($i=1; $i < $daysBetween; $i++) {
48                $day++;
49                if ($day > 6){$day = 0;}
50                elseif ($day > 0 && $day < 5) {$sec += (10*60*60);}
51                elseif ($day == 5) {$sec += (8*60*60);}
52            }
53            $sec += $end - strtotime('8am',$end);
54        }
55        else {$sec = 0;}
56    }
57    return $sec;
58 }

```

Schließlich wird in Zeile 52 noch der Anteil des letzten Tages zur Zwischensumme `$sec` addiert, wobei der individuelle Endzeitpunkt von der täglichen Startzeit (8 Uhr) subtrahiert wird. Die `else`-Anweisung in Zeile 54 wird nur dann ausgeführt, wenn beide ursprünglichen Zeitpunkte am gleichen Wochenende liegen. Da die automatische Justierung in diesem Fall den Beginn in die Zukunft (Montag 8 Uhr) und das Ende in die Vergangenheit (Freitag 16 Uhr) verschiebt, sind die im `if`-Clause in Zeile 42 geprüften `$daysBetween` negativ. In diesem Sonderfall ist folglich keine Zeitdifferenz zu berechnen, sodass die Funktion 0 Sekunden zurückliefert.

Eine weitere im Prozesskontext besonders relevante Funktion ist die `checkTeamComplete` Funktion (vgl. Listing 6.16). Bei jedem Aufruf einer incidentbezogenen Seite wird anhand dieser Funktion geprüft, ob für den konkreten Vorfall das Security Incident Team mindestens minimal besetzt ist, also ein Hotliner und ein SIC definiert wurden. Zunächst wird in den Zeilen 2 und 3 die `getIncidentValue` Funktion zur Ermittlung der Klassifikation und des Incidentstatus angewendet. Darüber hinaus wird in Zeile 4 die Anzahl derjenigen Personen aus der Datenbank abgefragt, die als Hotliner oder SIC dem Vorfall zugeordnet sind. Da die Überprüfung der Teambesetzung im Falle eines Duplikats oder eines Information Requests obsolet ist, wird diese nach Prüfung der Klassifikation in Zeile 5 mit dem Return-Wert `TRUE` übersprungen. Da auch in den Prozessphasen `NEW` und `CLASS` die beiden Teampositionen noch nicht zwangsweise festgelegt sind, greift bei einem derartigen Incidentstatus der `else`-Zweig in Zeile 13, der die Funktion mit dem Rückgabewert `TRUE` verlässt. Nur dann, wenn ein Sicherheitsvorfall vorliegt, der sich mindestens in der Analysephase befindet, wird in Zeile 6 die zuvor aus der Datenbank abgerufene Personenzahl (Hotliner und SIC) betrachtet. Ist diese ungleich 2, so wird in Zeile 7 mit `setAlert` eine Aufforderung zur Teamvervollständigung erzeugt, woraufhin der Anwender zur Teamseite des Incidents umgeleitet wird. Da sich ein bereits festgelegter Hotliner oder SIC jederzeit wieder löschen lässt, reicht eine einmalige Teamprüfung nicht aus, weswegen die `checkTeamComplete` in allen Prozessphasen aufgerufen wird. Dadurch wird eine weitere Bearbeitung des Vorfalls unterbunden, solange nicht beide Teampositionen zugeordnet sind.

Listing 6.16: Funktion `checkTeamComplete (/inc/functions.si.php)`

```

1 function checkTeamComplete($db, $si){
2     $klassifikation = getIncidentValue($db, $si, 'klassifikation');
3     $currentState = getIncidentValue($db, $si, 'currentState');
4     $query = "SELECT COUNT(*) FROM ".DBPREFIX."contacts WHERE si=" .$_GET['si']."
5         AND contactRole IN ('HOTLINER', 'SIC)";
6     if ($klassifikation != "INCIDENT") {return TRUE;}
7     elseif ($currentState != "NEW" AND $currentState != "CLASS" AND $db->query(
8         $query)->fetchColumn() != 2) {
9         setAlert('error', "Team unvollständig! Bitte legen Sie den Hotliner und den
10            SIC fest!");
11         $db = null;
12         header("Location: /si-team.php?si=$si");
13         exit;
14     }
15     return FALSE;
16 }
17 else {return TRUE;}
18 }

```

6. Implementierung

Die in Listing 6.17 abgebildete Funktion `insertHistory` ermöglicht eine incidentbezogene Dokumentation aller Einzelschritte des Prozessverlauf. Sie ist durch den Anwender nicht direkt aufrufbar, sondern wird im Kontext anderer Aktionen und Funktionen automatisch aufgerufen. In den Zeilen 2 und 3 wird ein neuer Datensatz in der `history`-Tabelle angelegt, der neben den IDs des Incidents und des Anwenders die Bezeichnung des betroffenen Datenbankfeldes, den Wert des Feldes vor und nach der Änderung sowie den Namen des Anwenders enthält. Im Normalfall liefert die Funktion `TRUE` zurück (Zeile 9). Tritt beim schreibenden Zugriff auf die `history`-Tabelle hingegen ein Fehler auf, so wird dieser in den Zeilen 5 bis 8 behandelt. Dabei wird zunächst mit `writeSyslog` ein Eintrag im Systemlog erzeugt und anschließend die Funktion mit dem Return-Wert `FALSE` verlassen.

Listing 6.17: Funktion `insertHistory` (`/inc/functions.si.php`)

```
1 function insertHistory ($db, $si, $user, $field, $old, $new, $userName) {
2     $sql = "INSERT INTO ".DBPREFIX." history (si, user, field, old, new, userName)
3         VALUES ('$si', '$user', '$field', '$old', '$new', '$userName)";
4     $query = $db->query($sql);
5
6     if (!$query) {
7         writeSyslog($db, $user, $si, 'insertHistory', 'ERROR', 'DB: '.var_dump($db->
8             errorInfo()));
9         return FALSE;
10    }
11    else {return TRUE;}
12 }
```

Dadurch, dass der Aufruf der Funktion `insertHistory` fest im Code der incidentbezogenen Aktionen bzw. Funktionen verankert ist und nicht durch das Setzen spezieller Parameter umgangen werden kann, ist sichergestellt, dass jeder Vorgang zuverlässig und unverfälscht anhand der Originaldaten dokumentiert wird. Da die Webanwendung weder Funktionen zum Löschen und Ändern noch zum direkten, ereignisunabhängigen Hinzufügen von Historyeinträgen bereitstellt, ist die Manipulationssicherheit des Verlaufsprotokolls gewährleistet.

Um den Prozessverlauf für die CSIRT-Mitglieder möglichst komfortabel und effizient zu gestalten, wird die Benutzeroberfläche im Bereich der Vorfallobarbeitung um eine zusätzliche Incident Menüleiste sowie eine Sidebar erweitert (vgl. Abbildung 6.8). Die technische Umsetzung richtet sich nach den Vorgaben des Konzepts und kann in den Dateien `/inc/tpl.si.nav.php` bzw. `/tpl/si-sidebar.html` eingesehen werden. Die auf dem Screenshot abgebildete Incidentstatusseite (`/si-overview.php`) fasst alle zu einem Vorfall vorliegenden Informationen auf einer Seite übersichtlich zusammen und beinhaltet auch Metainformationen wie die Laufzeiten der einzelnen Prozessphasen. Die Statusseite ist flexibel implementiert, sodass auch Duplikate und Information Requests dargestellt werden können. Zudem ist bei Sicherheitsvorfällen neben der reduzierten Standardversion eine erweiterte Fassung abrufbar.

The screenshot displays the 'LRZ Security Incident Management System' interface. The main header shows the incident title 'SI-46: Angriff auf den Hochschulstart-Dienst'. Below this, there are navigation tabs: 'Übersicht', 'Vorfall', 'Team', 'Maßnahmen', 'Lösung', 'Monitoring', 'Review', 'Kommunikation', and 'Verlauf'. The 'Übersicht' tab is active.

The 'Incident-Daten' section contains a table with the following information:

ID	46
Titel	Angriff auf den Hochschulstart-Dienst
Beschreibung	Der Hochschulstartdienst wurde kompromittiert. Möglicherweise wurden vertrauliche Daten abgegriffen.
Vorfallzeitpunkt	zwischen 09.06. und 11.06.2014
Kunde	Hochschulstart
Betriebssystem	Linux
Hostname	hochschulstart.lrz.de
IP-Adresse	192.168.4.65

The 'Security Incident' summary on the right includes: 'Analyse hoch SFH', 'Typ: Kompromittierter Webserver', 'Kunde: Hochschulstart', 'System: Linux', 'Host: hochschulstart.lrz.de', and 'IP: 192.168.4.65'. Below this, 'Empfehlungen' and 'Aktionen' are listed.

The 'Meta-Informationen' section at the bottom shows a table with columns: 'Art des Vorfalls', 'Meldezeitpunkt', 'Aktueller Status', and 'Abgeschlossen'.

Abbildung 6.8.: Screenshot: Incidentstatusseite (/si-overview.php)

6.2.6. Kommunikation

Da eine reibungslose Kommunikation die effiziente Incidentbearbeitung begünstigt, stellt die Webanwendung ein voll integriertes Kommunikationsmodul bereit. Dieses unterstützt einerseits verschiedene Arten des Mailversands (/si-com-mail.php), wobei die Kontaktdaten der Teammitglieder als Adressbuch zur Verfügung stehen, und andererseits die Erfassung von Gesprächsnotizen und eingegangenen E-Mails (/si-com-memo.php). Dabei werden sämtliche Vorgänge mit der Funktion `addComItem` automatisch in einem incidentbezogenen Kommunikationsprotokoll (/si-com.php) dokumentiert.

Für den tatsächlichen Versand von E-Mails baut die Klasse `PHPMailer`⁴ eine SSL/TLS verschlüsselte Verbindung zu einem SMTP-Server auf, sodass die Übertragung der sensiblen Daten über einen gesicherten Kommunikationskanal gewährleistet ist. Die Webanwendung unterscheidet im Bereich des Mailversands drei Anwendungsfälle:

- Ist beispielsweise eine Anfrage weiterer Informationen an den Vorfallmelder zu richten, da die gemeldeten Daten unvollständig sind, so wird für den Mailversand die Funktion `mailtoContact` verwendet. Diese Funktion eignet sich auch für E-Mails an ein anderes Mitglied des Security Incident Teams, das aus dem adressbuchartigen Dropdown-Feld (`parseSiContactsDropdown`) komfortabel ausgewählt werden kann. Anhand der selektierten `contactId` werden die für den Mailversand benötigten Empfängerdaten aus der `contacts`-Tabelle abgerufen.
- Mit der Funktion `mailtoSomebody` können E-Mails an einen beliebigen Empfänger – beispielsweise an einen externen IT-Forensiker – gesendet werden. In diesem Fall werden die Empfängerdaten als Inputparameter an die Funktion übergeben.
- Die dritte Funktion `mailtoGroup` ermöglicht das Senden einer E-Mail an mehrere Empfänger, beispielsweise an alle Mitglieder des Security Incident Teams. Hierbei sind die Empfängerdaten in Form eines mehrdimensionalen Arrays, das alle Namen und E-Mail-Adressen enthält, an die Funktion zu übergeben.

⁴<https://github.com/PHPMailer/PHPMailer>

6. Implementierung

Ereignisbedingte E-Mails wie die Benachrichtigung des Vorfalle Melders über den Abschluss seines Incidents werden durch Aufruf einer der genannten Mailfunktionen innerhalb des jeweiligen Aktionskontextes (hier: `closeIncident`) ausgelöst. Der Versand zeitlich gesteuerter E-Mails wird hingegen durch einen Cronjob veranlasst.

Für den Versand individueller E-Mails über die Webanwendung ist die Seite (`/si-com-mail.php`) aufzurufen. Nach der Auswahl des Empfängers (Adressbuch, Gruppe, manuelle Eingabe) sind der Betreff und der E-Mail-Inhalt einzugeben. Außerdem besteht die Möglichkeit, durch Checkbox-Auswahl verschiedene Vorfalldaten anzufügen: Basisdaten, Kontaktdaten, Analyse, Lösung, Monitoring, Review. Mit dem Absenden des Formulars wird die Aktion `prepareMail` aufgerufen, welche die ausgewählten Informationen aus der Datenbank abfragt und am Ende der Nachricht einfügt. Der Anwender kann die angezeigte E-Mail-Vorschau vor dem endgültigen Mailversand (`sendPreparedMail`) editieren, um beispielsweise einzelne Daten zu entfernen, die für den Empfänger nicht relevant sind. Welche der drei Mailfunktionen anzuwenden ist, entscheidet das System zur Laufzeit in Abhängigkeit vom Empfängertyp.

Darüber hinaus ermöglicht das Kommunikationsmodul auch die Meldung eines Incidents an die Allianz für Cyber-Sicherheit. Die Seite `/si-com-acs.php` enthält eine Nachbildung des offiziellen Meldeformulars, das bereits soweit wie möglich vorausgefüllt ist. Dieses Formular umfasst acht Fragestellungen, die im Wesentlichen durch das Anklicken von Checkboxes zu beantworten sind. Im Vergleich zur Nutzung des externen Meldeformulars reduziert die anwendungsinterne Meldung den Umfang manuell einzugebender Informationen und somit den Zeitaufwand. Darüber hinaus ist eine automatische Dokumentation der Meldung im Kommunikationsprotokoll gewährleistet. Auf technischer Ebene betrachtet, werden die Formulareingaben an die Aktion `sendAcsMail` übermittelt. Diese generiert anhand der Eingaben den Mailinhalt und übergibt diesen an die Funktion `mailToSomebody`, wobei die Empfängerinformationen aus der Konfigurationsdatei stammen.

Die Vorgehensweise der Mailfunktionen wird am Beispiel `mailToSomebody` (vgl. Listing 6.18) vorgestellt. Für den Versand der E-Mail via `PHPMailer` und die Dokumentation des Vorgangs in der `communications`-Tabelle erwartet die Funktion neun Eingabeparameter:

- `$db` – Handler für Zugriff auf die `communications`-Tabelle
- `$si` – ID des Sicherheitsvorfalls zur Zuordnung der versendeten E-Mail
- `$name` – Name des Empfängers
- `$email` – E-Mail-Adresse des Empfängers
- `$contactRole` – Funktion des Empfängers
- `$subject` – Betreff der E-Mail
- `$body` – Inhalt der E-Mail
- `$senderData` – Absender: `SYSTEM` oder `SESSION`. Ist als Absender `SYSTEM` eingetragen, so wird der in der Konfigurationsdatei festgelegte Standardabsender verwendet. Die Option `SESSION` bewirkt, dass die in der Session gespeicherten Daten des agierenden Benutzers als Absenderdaten verwendet werden.
- `$autoGreeting` – Flag: Gibt an, ob eine automatische Grußformel an den Mailtext angehängt werden soll. Dieses Flag ist standardmäßig gesetzt und muss daher nicht zwingend an die Funktion übergeben werden.

Zunächst wird in Zeile 3 ein neues `simsMailer` Objekt der Klasse `PHPMailer` erzeugt. Anschließend werden die Absenderdaten und die Grußformel festgelegt. Falls das System als Absender fungieren soll (Zeile 5), gelten automatisch die in der Konfigurationsdatei hinterlegten Daten, sodass in Zeile 6 nur die Grußzeile definiert wird. Soll hingegen der agierende Nutzer als Absender auftreten (Zeile 8), beinhaltet die Grußformel dessen Vor- und Nachnamen (Zeile 9). In den Zeilen 10 und 11 werden die E-Mailadresse und der Name des Anwenders – sofern vorhanden – als Absenderdaten an das `simsMailer` Objekt übergeben. Ist der Name oder die E-Mail-Adresse des Anwenders nicht verfügbar, so werden automatisch die Standarddaten aus der Konfigurationsdatei verwendet.

In Zeile 14 wird der übergebene Betreff um ein Präfix (ID des Vorfalls) ergänzt. Dadurch wird sichergestellt, dass die E-Mail auf den ersten Blick einem Vorfall zugeordnet werden kann. Falls das `$autoGreeting` Flag gesetzt ist, wird in Zeile 15 der Mailbody zusammengesetzt, wobei eine neutrale Begrüßung vorangestellt und die zuvor definierte Grußformel angehängt wird. Andernfalls bleibt der E-Mail-Inhalt unverändert (Zeile 16). Anschließend werden die E-Mail-Bestandteile Empfänger, Betreff und Inhalt in den Zeilen 17 bis 19 an das `simsMailer` Objekt übergeben. Der E-Mail-Versand erfolgt in Zeile 21. Ist dieser erfolgreich, so wird die ausgehende E-Mail mit dem aktuellen Zeitstempel mittels `addComItem` incidentbezogen in der `communications`-Tabelle gespeichert (Zeile 22) und ist somit für alle CSIRT-Mitglieder im Kommunikationsprotokoll des Vorfalls einsehbar. Im Erfolgsfall liefert die Funktion `TRUE`, im Fehlerfall `FALSE` zurück, wobei die Fehlerbehandlung in der aufrufenden Aktion erfolgt.

Listing 6.18: Funktion `mailtoSomebody (/inc/functions.si.php)`

```

1 function mailtoSomebody($db, $si, $name, $email, $contactRole, $subject, $body,
2     $senderData, $autoGreeting=TRUE){
3     global $appConfig;
4     $mail = new simsMailer;
5
6     if($senderData == "SYSTEM") {
7         $greeting = "\n\nBeste Grüße von Ihrem\n\nSecurity Incident Management
8         System";
9     }
10    else {
11        $greeting = "\n\nBeste Grüße\n\n" . $_SESSION['userVorname'] . " " . $_SESSION['
12        userName'] . "\nComputer Security Incident Response Team (CSIRT)";
13        if ($_SESSION["userEmail"] != ""){$mail->From = $_SESSION["userEmail"];}
14        if ($_SESSION["userName"] != ""){$mail->FromName = $_SESSION["userVorname"]
15        ]. " " . $_SESSION["userName"] . " (" . $appConfig['sysname'] . ")";}
16    }
17
18    $newSubject = "[SI-$si] " . $subject;
19    if($autoGreeting == TRUE) {$newBody = "Hallo, \n\n" . $body . $greeting;}
20    else {$newBody = $body;}
21    $mail->addAddress($email, $name);
22    $mail->Subject = $newSubject;
23    $mail->Body = $newBody;
24
25    if($mail->send()){
26        addComItem($db, $si, $senderData, date("Y-m-d H:i:s", time()), "MAIL", "OUT",
27        $newSubject, $newBody, $name, null, $email, $contactRole);
28        return TRUE;
29    }
30    else {return FALSE;}
31 }

```

6.3. Erfassung von Sicherheitsvorfällen

Für die Erfassung eines neuen Sicherheitsvorfalls wird in der Datei `/new.php` ein zweigeteiltes Formular bereitgestellt. Während im oberen Teil Informationen zum Vorfall einzutragen sind, umfasst der untere Teil die Kontaktdaten des Vorfalle Melders. Das Formular besteht aus einzeiligen Textfeldern, mehrzeiligen Textareas sowie Dropdown-Feldern. Letztere dienen der Auswahl des betroffenen Kunden und des Betriebssystems. Die Listen der Kunden und Betriebssysteme werden durch die Funktion `getTypesArray` ermittelt und an die Funktion `parseDropDown` zur dynamischen Generierung der Dropdown-Felder übergeben.

Aufgrund der LDAP-Synchronisation stehen die Kontaktdaten des Vorfalle Melders in der `users`-Tabelle bereit. Folglich werden die Kontaktinformationen zur Vorbelegung der Formularfelder aus der Datenbank abgefragt und an das Template übergeben. Dadurch, dass nur noch optionale Angaben zur Handynummer und Erreichbarkeit zu machen sind, reduziert sich der manuelle Eingabeaufwand von Personeninformationen erheblich. Pflichtfelder wie der Titel und die Beschreibung des Vorfalls werden im HTML-Code mit dem Parameter `required` versehen. Dies bewirkt eine browserseitige Prüfung dieser Felder beim Absenden des Formulars. Der Meldezeitpunkt und die Benutzerkennung sind durch die Template-Anweisung `{if = '$userRole == USER' } readonly {/if}` für LRZ-Mitarbeiter nicht editierbar. CSIRT-Mitglieder hingegen können beispielsweise bei der nachträglichen Erfassung eines telefonisch gemeldeten Vorfalls einen abweichenden Meldezeitpunkt und die Kennung des meldenden Kollegen eintragen.

Durch das Absenden des Formulars werden die eingegebenen Daten mittels `POST`-Request an die Aktion `newIncident` (vgl. Listing 6.19) übergeben. Zunächst wird in Zeile 1 eine serverseitige Prüfung der Pflichtfelder vorgenommen. Enthält eines dieser Felder keine Daten, so werden die übermittelten Daten im Rahmen einer `foreach`-Schleife (Zeile 2) in der Session-Variable `formerror` zwischengespeichert (Zeile 3). Mit `setAlert` wird in Zeile 6 eine Fehlermeldung generiert, woraufhin der Anwender in Zeile 7 zum Meldeformular zurückgeleitet wird. Der hierbei übergebene `GET`-Parameter `formerror=1` sorgt dafür, dass die Eingaben aus der Session-Variablen abgerufen und zur Vorbelegung der Formularfelder verwendet werden. Durch die Zwischenspeicherung gehen die zuvor gemachten Angaben nicht verloren, sodass eine Vervollständigung auf Basis des bisherigen Datenbestands möglich ist.

Ist die Prüfung der Pflichtfelder erfolgreich, so wird anhand der erfassten Daten in den Zeilen 11 bis 18 ein neuer Datensatz mit dem Status `NEW` in der `incidents`-Tabelle erzeugt. Ein eventueller Datenbankfehler wird in den Zeilen 14 bis 17 behandelt, wobei der Anwender zum Meldeformular zurückgeleitet wird. Im Standardfall werden nun die Incident Daten in die `history`-Tabelle eingetragen. Da die Dokumentation feldweise atomar erfolgen soll, ruft die `foreach`-Schleife für jedes Feld die Funktion `insertHistory` auf (Zeilen 20 und 21). Anschließend erfolgt in den Zeilen 24 bis 26 die Speicherung der Kontaktdaten des Vorfalle Melders in der `contacts`-Tabelle. Zuletzt werden die CSIRT-Mitglieder und der Vorfalle Melder über den Eingang eines neuen Incidents per E-Mail benachrichtigt. Für den Versand der E-Mail an das CSIRT in den Zeilen 29 bis 31 wird die Funktion `mailtoSomebody` verwendet, wobei die Empfängerdaten aus dem Array `$csirtRecipient` der Konfigurationsdatei stammen. Die Benachrichtigung des Vorfalle Melders (Zeilen 33 bis 35) erfolgt hingegen mit der `mailtoContact` Funktion, wobei die Empfängerdaten anhand der `$contactId` aus der `contacts`-Tabelle abgefragt werden. Abschließend wird eine Erfolgsmeldung gesetzt (Zeile 38), woraufhin der Anwender in den Zeilen 39 bis 42 in Abhängigkeit von seiner `userRole` entweder zum Dashboard (LRZ-Mitarbeiter) oder zur Incidentstatusseite (CSIRT-Mitglieder) weitergeleitet wird.

Listing 6.19: Aktion newIncident (/actions/new.php)

```

1  if ($_POST['title'] == "" || $_POST['description'] == "" || $_POST['
    createdStamp'] == "" || $_POST['vorname'] == "" || $_POST['name'] == "" ||
    $_POST['email'] == "" || $_POST['tel'] == "" || $_POST['kennung'] == ""){
2  foreach(array("title","description","vorfallzeitpunkt","kunde","host","ip","
    os","createdStamp","kennung","name","vorname","email","tel","mobil","
    comment") as $key){
3      $_SESSION['formerror'][$key] = $_POST[$key];
4  }
5  $db = null;
6  showAlert('error','Es wurden nicht alle Pflichtfelder ausgefüllt.');
```

7 header("Location: ".\$appConfig['url']."/new.php?formerror=1"); exit;

8 }

9 \$_POST['currentState'] = "NEW";

10 \$sql = "INSERT INTO ".DBPREFIX."incidents (user,createdStamp,currentStamp,
 currentState,title,description,vorfallzeitpunkt,kunde,host,ip,os) VALUES
 (?,?,?,?,?,?,?,?,?,?,?)";

12 \$query = \$db->prepare(\$sql);

13 \$query->execute(array(\$_SESSION['userId'],\$_POST['createdStamp'],\$_POST['
 createdStamp'],\$_POST['currentState'],\$_POST['title'],\$_POST['description']
 ,\$_POST['vorfallzeitpunkt'],\$_POST['kunde'],\$_POST['host'],\$_POST['ip']
 ,\$_POST['os']));

14 if(!\$query){

15 showAlert('error','Der Security Incident konnte nicht angelegt werden.');

16 header("Location: ".\$_SESSION['lastUrl']); exit;

17 }

18 \$si = \$db->lastInsertId();

19

20 foreach(array("createdStamp","currentState","title","description","
 vorfallzeitpunkt","kunde","host","ip","os") as \$key){

21 insertHistory(\$db,\$si,\$_SESSION['userId'],\$key,NULL,\$_POST[\$key],\$_SESSION[
 'userName']."", \$_SESSION['userVorname']);

22 }

23

24 \$sql = "INSERT INTO ".DBPREFIX."contacts (si,contactRole,kennung,name,vorname,
 anrede,email,tel,mobil,comment) VALUES (?,?,?,?,?,?,?,?,?,?,?)";

25 \$query = \$db->prepare(\$sql);

26 \$query->execute(array(\$si,'MELDER',\$_POST['kennung'],\$_POST['name'],\$_POST[
 'vorname'],\$_POST['anrede'],\$_POST['email'],\$_POST['tel'],\$_POST['mobil']
 ,\$_POST['comment']));

27 \$contactId = \$db->lastInsertId();

28

29 \$subject = "Neuer Security Incident eingegangen";

30 \$body = "..."; // Mailinhalt zur besseren Übersichtlichkeit entfernt

31 mailToSomebody(\$db,\$si,\$csirtRecipient['name'],\$csirtRecipient['email'],
 "CSIRT", \$subject,\$body,"SYSTEM");

32

33 \$subject = "Neuer Security Incident eingegangen";

34 \$body = "..."; // Mailinhalt zur besseren Übersichtlichkeit entfernt

35 mailToContact(\$db,\$si,\$contactId,\$subject,\$body,"SYSTEM");

36

37 \$db = null;

38 showAlert('ok','Der Security Incident mit der ID '.\$si.' wurde erfasst.');

39 if(\$_SESSION['userRole'] == "USER"){

40 header("Location: ".\$appConfig['url']."/dashboard.php"); exit;

41 }

42 else {header("Location: ".\$appConfig['url']."/si.php"); exit;}

6.4. Prozesskonforme Bearbeitung von Sicherheitsvorfällen

Auf Basis der in Abschnitt 6.2.5 beschriebenen Prozessunterstützung wird im Folgenden auf die Implementierung des Bearbeitungsprozesses eingegangen. Dieser beginnt mit der initialen Klassifikation und führt – nach einem Exkurs zu Duplikaten und Information Requests – über die Ermittlung der Priorität zur Teambzusammenstellung. In der Analyse-, Lösungs- und Monitoringphase liegt der Fokus auf der Aktion `updateIncidentValues`. Daraufhin werden die Aktionen zum Schließen, Wiedereröffnen und Abbrechen eines Vorfalls beschrieben. Die Durchführung eines Post Incident Reviews rundet das Kapitel ab.

6.4.1. Initiale Klassifikation

Ein neuer Vorfall ist vor der weiteren Bearbeitung durch den CSIRT-Hotliner initial zu klassifizieren. Hierzu werden auf der Seite `/si-init.php` die Incident Informationen und die Kontaktdaten des Vorfalle Melders tabellarisch dargestellt. Auf Grundlage dieser Daten entscheidet der Hotliner, ob es sich um einen neuen Sicherheitsvorfall, um ein Duplikat eines bereits gemeldeten Incidents oder um einen Information Request handelt. Zur übersichtlichen Darstellung der Möglichkeiten werden drei parallel angeordnete Widgets eingesetzt, die jeweils ein Dropdown-Feld und einen Submit-Button enthalten (vgl. Screenshot 6.9).

The screenshot shows a web interface titled "Bitte entscheiden Sie" (Please decide). It contains three distinct widgets for classification:

- Neuer Security Incident:** A box with a title bar. Below the title, it says "Es handelt es sich um einen neuen, noch nicht erfassten Sicherheitsvorfall vom Typ:". Below this is a dropdown menu with "Kompromittierter Web" selected and a green "weiter" button.
- Incident Duplikat:** A box with a title bar. Below the title, it says "Es handelt sich um ein Duplikat. Dieser Vofall wurde bereits gemeldet:". Below this is a dropdown menu and a green "weiter" button.
- Information Request:** A box with a title bar. Below the title, it says "Es handelt sich um eine Anfrage, die beantwortet oder weitergeleitet werden soll.". Below this is a dropdown menu and a green "weiter" button.

Abbildung 6.9.: Screenshot: Initiale Klassifikation (`/si-init.php`)

Liegt ein neuer Sicherheitsvorfall vor, so wählt der Hotliner im Rahmen der initialen Klassifikation bereits den Incidenttyp aus. Das Dropdown-Feld wird flexibel durch die Funktion `parseDropDown` generiert, wobei neben den vier Standard Security Incidents auch die Auswahl der Restklasse „Kein Standard Security Incident“ möglich ist. Das Absenden des Formulars bewirkt eine Übermittlung des Incidenttyps an die Aktion `classifyIncident` (vgl. Listing 6.20). Da die Klassifikation nur im Status `NEW` zulässig ist, erfolgt in Zeile 1 eine entsprechende Überprüfung mittels `checkState`. Anschließend wird in den Zeilen 3 bis 7 sichergestellt, dass der Vorfall nicht bereits klassifiziert wurde. Ist das Feld `klassifikation` nicht `NULL`, so wird der Hotliner zur Statusseite des Incidents geleitet und über den Fehler informiert. Nachdem die Klassifikation und der Incidenttyp in der Datenbank gespeichert wurden (Zeilen 9 und 10), wird die Klassifikation in Zeile 12 mit einer `insertHistory`-Anweisung dokumentiert. Sofern der übermittelte Incidenttyp nicht der Restklasse entspricht (Zeile 14), wird die anschließend vorgestellte Funktion `applySsiTemplate` aufgerufen, sodass bereits zu Beginn die Daten der Vorlage auf den aktuellen Vorfall übertragen werden. Schließlich wird die reguläre `updateState` Funktion aufgerufen, welche die üblichen Operationen zum Statuswechsel ausführt (vgl. Listing 6.14 in Abschnitt 6.2.5). Als dritter Parameter wird anstelle eines Zeitstempels der Wert `FALSE` übergeben. Dieser sorgt dafür, dass die Funktion den aktuellen Zeitstempel annimmt.

Listing 6.20: Aktion `classifyIncident (/actions/incidents.php)`

```

1 checkState($db,$POST['si'],'NEW');
2
3 if(!is_null(getIncidentValue($db,$POST['si'],'klassifikation'))){
4     showAlert('error','Der Vorgang SI-' . $POST['si'] . ' wurde bereit klassifiziert
5     .');
6     header("Location: " . $appConfig['url'] . "/si-overview.php?si=" . $POST['si']);
7     exit;
8 }
9 $sql = "UPDATE " . DBPREFIX . "incidents SET klassifikation = 'INCIDENT',
10     incidentType=" . $POST['classifyIncident'] . " WHERE id=" . $POST['si'];
11 $query = $db->query($sql);
12 insertHistory($db,$POST['si'],$SESSION['userId'],'klassifikation',"",
13     "INCIDENT",$SESSION["userName"],",",$SESSION["userVorname"]);
14
15 if ($POST['classifyIncident'] != "SONST") {
16     applySsiTemplate($db,$POST['si'],$POST['classifyIncident']);
17 }
18 updateState($db,$POST['si'],FALSE);

```

Die soeben erwähnte Funktion `applySsiTemplate` (vgl. Listing 6.21) erwartet als Eingabeparameter einen Datenbankhandler, die ID des Vorfalls und die ID des Templates, das auf den Vorfall angewendet werden soll. Zunächst wird der Templatedatensatz anhand dessen ID aus der `ssiTpl`-Tabelle abgerufen und in der lokalen Variable `$template` gespeichert. Da die Spaltenbezeichnungen der Vorlagentabelle mit denen der `incidents`-Tabelle übereinstimmen, kann die Datenübernahme effizient abgewickelt werden. Zu deren Vorbereitung wird im Rahmen einer `foreach`-Schleife (Zeilen 6 bis 10) ein SQL-tauglicher `$updateString` generiert, der mit Ausnahme des `id`-Feldes (Zeile 7) alle Bezeichnungen und Werte des `$template`-Datensatzes umfasst. Am Ende des Strings befindet sich schleifenbedingt ein Komma, das in Zeile 12 entfernt wird, um die Korrektheit der SQL-Syntax zu gewährleisten. Daraufhin wird in den Zeilen 13 und 14 der `UPDATE`-Befehl, der den zuvor generierten und bereinigten `$updateString` enthält, an die Datenbank übermittelt. Schlägt diese Anfrage fehl, so wird in den Zeilen 15 bis 18 eine Fehlermeldung gesetzt und die Funktion wird mit dem Return-Wert `FALSE` verlassen. Im Normalfall (Zeile 19 bis 22) erzeugt die Funktion eine Erfolgsmeldung und gibt `TRUE` zurück.

Listing 6.21: Funktion `applySsiTemplate (/inc/functions.si.php)`

```

1 function applySsiTemplate($db,$si,$templateId){
2     $query = "SELECT * FROM " . DBPREFIX . "ssiTpl WHERE id=" . $templateId . " ";
3     $stmt = $db->query($query);
4     $template = $stmt->fetch(PDO::FETCH_ASSOC);
5
6     foreach($template as $key => $value){
7         if ($key != "id"){
8             $updateString .= $key . "=" . $value . ", ";
9         }
10    }
11
12    $updateString = rtrim($updateString, ", ");
13    $sql = "UPDATE " . DBPREFIX . "incidents SET $updateString WHERE id=$si ";
14    $query = $db->query($sql);

```

6. Implementierung

```
15  if (!$query){
16      showAlert("error","Die Felder konnten nicht mit dem SSI Template
    $templateId vorbelegt werden.");
17      return FALSE;
18  }
19  else {
20      showAlert("ok","Die Felder wurden mit dem SSI Template $templateId
    vorbelegt.");
21      return TRUE;
22  }
23 }
```

6.4.2. Umgang mit Duplikaten und Information Requests

Stellt der Hotliner im Rahmen der initialen Klassifikation fest, dass der gemeldete Vorfall bereits von einem anderen Kollegen gemeldet wurde, so handelt es sich um ein Duplikat, das nicht weiter zu bearbeiten ist. In diesem Fall stehen im Dropdown-Feld des mittleren Widgets (vgl. Screenshot 6.9) alle in Bearbeitung befindlichen oder abgeschlossenen Sicherheitsvorfälle zur Auswahl. Auch dieses Feld wird dynamisch generiert, wobei die Funktion `parseIncidentsDropdown` zum Einsatz kommt. Nach der Festlegung des zugehörigen Eltern-Incidents wird dessen ID an die Aktion `classifyDuplicate (/actions/incidents.php)` übergeben. Wegen der strukturellen Ähnlichkeit zur `classifyIncident` Aktion (vgl. Listing 6.20), wird hier nur auf die wesentlichen Unterschiede eingegangen:

- `klassifikation: DUPLICATE`
- `currentState: CLOSED`
- `closedState: SI_DUPLICATE`

Auch hier wird die Klassifikation mittels `insertHistory` dokumentiert. Um das Duplikat mit dem Original zu verknüpfen, wird im Feld `duplicateId` des Duplikatdatensatzes die ID des originalen Incidents gespeichert. Da der Vorfall bereits durch das Datensatzupdate geschlossen wurde, ist ein Statuswechsel mit der Funktion `updateState` nicht mehr erforderlich. Schließlich wird der Vorfallmelder per `mailtoContact` darüber informiert, dass der Vorfall bereits bekannt ist und als Duplikat abgeschlossen wurde. Darüber hinaus wird ihm die ID des originalen Incidents mitgeteilt, um gezielte Rückfragen zu ermöglichen.

Handelt es sich hingegen um einen Information Request, so wählt der Vorfallmelder im rechten Widget (vgl. Screenshot 6.9) aus, ob er den Vorfall selbst beantworten oder an die zuständige Bearbeitergruppe weiterleiten wird. Dementsprechend speichert die Aktion `classifyRequest (/actions/incidents.php)` die Klassifikation in der Datenbank:

- `klassifikation: REQUEST`
- `currentState: IR_AW / IR_FWD`

Nach der Dokumentation der Datensatzänderung mittels `insertHistory` Funktion wird dem Nutzer ein Mailformular zur Beantwortung (`/si-com-request-aw.php`) bzw. Weiterleitung (`/si-com-request-fwd.php`) angezeigt. Während das Antwortformular mit einem Muster-text vorausgefüllt ist, beinhaltet das Weiterleitungsformular alle zum Vorfall bekannten Informationen. Nach der Ergänzung individueller Hinweise wird der Mailinhalt an die Aktion `sendAwMail` bzw. `sendFwdMail` des Requestmoduls (`/actions/request.php`) übergeben.

Da sich die beiden Aktionen nur minimal unterscheiden, wird im Folgenden auf die Aktion `sendAwMail` (vgl. Listing 6.22) eingegangen: Der Mailversand an den Vorfallesteller wird in Zeile 1 mit der Funktion `mailtoContact` realisiert, die für die automatische Dokumentation der versendeten E-Mail mittels `addComItem` sorgt. Anschließend wird der Vorfall durch ein Datensatzupdate geschlossen (Zeilen 3 und 4). Diese Änderungen werden in den Zeilen 6 und 7 mittels `insertHistory` im Verlaufsprotokoll gespeichert. Zuletzt wird der Anwender zum Kommunikationsprotokoll dieses Vorfalls geleitet (Zeile 10) und über den erfolgreichen Versand der E-Mail informiert (Zeile 9).

Listing 6.22: Aktion `sendAwMail (/actions/request.php)`

```

1 mailtoContact($db,$_POST['si'],FALSE,$_POST['subject'],$_POST['content'],'
  SESSION',"MELDER");
2
3 $sql = "UPDATE ".$DBPREFIX."incidents SET currentState='CLOSED', closedState='
  IR_AW', currentStamp='$stamp', closedStamp='$stamp' WHERE id=".$_POST['si'
  ];
4 $query = $db->query($sql);
5
6 insertHistory($db,$_POST['si'],$SESSION['userId'], "currentState", "", "CLOSED",
  $SESSION["userName"].", ".$SESSION["userVorname"]);
7 insertHistory($db,$_POST['si'],$SESSION['userId'], "closedState", "", "IR_AW",
  $SESSION["userName"].", ".$SESSION["userVorname"]);
8
9 showAlert("ok","Die E-Mail an den Vorfallesteller wurde versendet. Der Vorfall
  wurde geschlossen.");
10 header("Location: ../si-com.php?si=".$_POST["si"]);

```

Die Aktion `sendFwdMail` nutzt hingegen die Funktion `mailtoSomebody`, um die Anfrage an eine Mitarbeitergruppe weiterzuleiten. Zudem wird der Vorfallesteller mittels `mailtoContact` automatisch über die Weiterleitung seiner Anfrage und die Kontaktdaten der zuständigen Gruppe informiert, sodass Rückfragen ohne Umweg über das CSIRT direkt erfolgen können.

6.4.3. Weitere Klassifikation und Priorisierung

Im Folgenden wird nun davon ausgegangen, dass ein tatsächlicher Sicherheitsvorfall vorliegt. Als nächster Prozessschritt ist die Ergänzung der Vorfalleinformationen (Vorfallezeitpunkt, Kunde, Betriebssystem, Hostname, IP-Adresse) vorgesehen. Zu diesem Zweck stellt die Seite `/si-data.php` im oberen Teil ein Formular zur Bearbeitung der übermittelten Vorfalleinformationen bereit. Die per `parseDropDown` erzeugten Formularfelder des unteren Bereichs dienen der weiteren Klassifikation und Priorisierung des Incidents. Diese sind bereits mit den Daten eines SSI-Templates vorausgefüllt, sofern im Rahmen der initialen Klassifikation ein Standard Security Incident ausgewählt wurde.

Während der aktuellen Prozessphase, die intern durch die Status-ID `CLASS` repräsentiert wird, ist beispielsweise festzulegen, ob ein Major Incident zu erwarten ist und welche Erstmaßnahmen zu treffen sind. Das in der Prozessbeschreibung des LRZ definierte Verfahren zur Ermittlung der Auswirkung, Priorität und Reaktionszeit des Vorfalls ist ebenfalls Bestandteil dieser Phase. Die fünf hierfür erforderlichen Angaben (Standort des Quell- und Zielsystems, Art der betroffenen Dienste und Informationen sowie die Anzahl der betroffenen Systeme) werden aus dynamisch generierten Dropdown-Feldern ausgewählt, wobei jede Auswahloption zum Zweck der Weiterverarbeitung mit einem Zahlenwert zwischen 1 und 3 codiert ist. Am Beispiel der Dienste entspricht der Wert 1 dem Fall, dass „keine kritischen

6. Implementierung

Dienste“ betroffen sind. Sollten hingegen „wichtige LRZ-Dienste“ betroffen sein, so wäre der Wert 3 auszuwählen.

Die Webanwendung stellt innerhalb der Aktion `updateIncidentValues` (vgl. Listing 6.23) einen Automatismus bereit, der die Auswirkung, Priorität und Reaktionszeit anhand der in der SIR-Prozessbeschreibung des LRZ [Met13] definierten Vorgehensweise ermittelt. Sofern der Anwender die Automatik aktiviert hat (Zeile 1), wird in Zeile 2 die Summe der fünf Inputparameter gebildet. Da diese bereits als Zahlenwerte vorliegen, ist an dieser Stelle keine Übersetzung mehr nötig. Das sich über die Zeilen 4 bis 23 erstreckende if-Konstrukt setzt die Variablen `$auswirkung`, `$prio` und `$reaktionszeit` in Abhängigkeit vom Ergebnis der vorherigen Summenbildung. Sowohl Auswirkung als auch Priorität werden als Zahlenwerte zwischen 1 (niedrig) und 4 (sehr hoch) codiert. Die Reaktionszeit wird hingegen in Minuten gespeichert. Diese beträgt beispielsweise für einen niedrig priorisierten Vorfall *einen* Arbeitstag (10 Stunden) und weist dementsprechend den Wert 600 auf.

Ist die in Zeile 2 berechnete Summe kleiner fünf oder größer fünfzehn, so liegt das Ergebnis außerhalb des zulässigen Wertebereichs. Dieser Fehlerfall wird in den Zeilen 24 bis 29 behandelt. Dabei erfolgt zunächst ein Eintrag ins Systemlog. Anschließend wird eine Fehlermeldung gesetzt, woraufhin der Anwender zum Formular zurückgeleitet wird und dort seine Eingaben korrigieren kann.

Listing 6.23: Automatische Ermittlung von Auswirkung, Priorität und Reaktionszeit in der Aktion `updateIncidentValues`

```
1 if (($POST['feature'] == "si-data") AND ($POST['automatik'] == "1")){
2   $sum = $newValues['matrixZielsys'] + $newValues['matrixDienste'] +
3     $newValues['matrixInfos'] + $newValues['matrixAnzahl'] + $newValues['
4     matrixQuellsys'];
5
6   if (($sum >= 5) AND ($sum <= 6)) {
7     $newValues['auswirkung'] = "1"; // niedrig
8     $newValues['prio'] = "1"; // niedrig
9     $newValues['reaktionszeit'] = "600"; // 10 Stunden
10  }
11  elseif (($sum >= 7) AND ($sum <= 9)) {
12    $newValues['auswirkung'] = "2"; // mittel
13    $newValues['prio'] = "2"; // mittel
14    $newValues['reaktionszeit'] = "240"; // 4 Stunden
15  }
16  elseif (($sum >= 10) AND ($sum <= 12)) {
17    $newValues['auswirkung'] = "3"; // hoch
18    $newValues['prio'] = "3"; // hoch
19    $newValues['reaktionszeit'] = "120"; // 2 Stunden
20  }
21  elseif (($sum >= 13) AND ($sum <= 15)) {
22    $newValues['auswirkung'] = "4"; // sehr hoch
23    $newValues['prio'] = "4"; // sehr hoch
24    $newValues['reaktionszeit'] = "15"; // 15 Minuten
25  }
26  else {
27    writeSyslog($db, $_SESSION["userId"], $POST['si'], "automatik", "ERROR", "
28    Ungültiger Punktwert: $sum");
29    setAlert('error', 'Automatik-Parameter unvollständig!');
30    header("Location: ".$_SESSION['lastUrl']); exit;
31  }
32 }
```

6.4.4. Zusammenstellung des Security Incident Teams

Die Zusammenstellung des Security Incident Teams erfolgt ebenfalls in der Klassifikationsphase, aus Gründen der Übersichtlichkeit jedoch auf einer separaten Seite (`/si-team.php`). Der obere Bereich der Seite gliedert sich in drei nebeneinander angeordnete Widgets und ist für die obligatorischen Teammitglieder reserviert. Während das linke Widget die Kontaktdaten des Vorfalle Melders beinhaltet, kann im mittleren der Hotliner und im rechten der SIC aus einem Dropdown-Feld ausgewählt werden. Da für diese Teampositionen nur CSIRT-Mitglieder in Frage kommen, werden die beiden Felder mittels `parseCsirtDropdown` erzeugt. Der untere Bereich beinhaltet eine tabellarische Auflistung aller weiteren Teammitgliedern und ihrer Kontaktdaten.

Um ein Teammitglied hinzuzufügen, wird dessen Kennung und Rolle (`contactRole`) an die Aktion `addContact` (vgl. Listing 6.24) übergeben. Zunächst werden in den Zeilen 1 bis 3 die Kontaktdaten der Person anhand ihrer Kennung aus der `users`-Tabelle abgefragt und anschließend (Zeilen 5 bis 7) unter Zuordnung der Incident-ID und der `contactRole` als neuer Datensatz in der `contacts`-Tabelle gespeichert. Im Falle eines fehlerhaften Schreibzugriffs auf die Datenbank (Zeile 8 bis 12) wird auf der Teamseite eine Fehlermeldung angezeigt. Im Normalfall (Zeilen 13 bis 18) wird jedoch das Hinzufügen des Teammitglieds mittels `insertHistory` im Verlaufsprotokoll dokumentiert. Daraufhin wird der Anwender mit einer Erfolgsmeldung zur Teamseite zurückgeleitet.

Listing 6.24: Aktion `addContact` (`/actions/contacts.php`)

```

1 $query = "SELECT * FROM ".DBPREFIX." users WHERE kennung='". $_POST['kennung'] ."'
  ' LIMIT 1";
2 $stmt = $db->query($query);
3 $user = $stmt->fetch(PDO::FETCH_ASSOC);
4
5 $sql = "INSERT INTO ".DBPREFIX." contacts (si, contactRole, kennung, name, vorname,
  anrede, email, tel, mobil, comment) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
6 $query = $db->prepare($sql);
7 $query->execute(array($_POST["si"], $_POST["contactRole"], $user["kennung"],
  $user["name"], $user["vorname"], $user["anrede"], $user["email"], $user["tel"],
  $user["mobil"], $user["comment"]));
8 if (!$query) {
9     showAlert('error', 'Das Teammitglied konnte nicht angelegt werden. ');
10    header("Location: " . $_SESSION['lastUrl']);
11    exit;
12 }
13 else {
14    insertHistory($db, $_POST['si'], $_SESSION['userId'], "TEAM", "", $user["vorname"]
  $user["name"] wurde als $_POST["contactRole"] hinzugefügt.", $_SESSION['
  userName'], $_SESSION['userVorname']);
15    showAlert("ok", $user["vorname"] $user["name"] wurde zum SI-$_POST["si"]-Team als
  $_POST["contactRole"] hinzugefügt.);
16    header("Location: " . $_SESSION['lastUrl']);
17    exit;
18 }

```

Die Aktionen zum Ändern von Kontaktdaten (`editContact`) und Löschen von Teammitgliedern (`deleteContact`) sind auf ähnliche Weise implementiert. Beim Löschen wird zusätzlich darauf geachtet, dass es sich beim zu löschenden Kontakt nicht um den Vorfalle Melder handelt. Da sich alle anderen Teampositionen im Prozessverlauf ändern können, gilt die Ausnahme nur für die statische Position des Vorfalle Melders.

6. Implementierung

Um eine effiziente Kommunikation mit den Teammitgliedern zu ermöglichen, sind alle Einträge dieses incidentbezogenen Adressverzeichnisses mit einem Telefon- und Mailsymbol versehen. Möchte der Hotliner eine E-Mail an ein bestimmtes Teammitglied senden oder eine Gesprächsnotiz erfassen, genügt ein Klick auf das entsprechende Symbol, wobei der ausgewählte Mailempfänger bzw. Gesprächspartner via `GET`-Parameter an das Kommunikationsmodul übermittelt wird.

Nach Abschluss der Teamzusammenstellung kann mit der Analyse des Vorfalles begonnen werden. Hierfür ist es erforderlich, in der Webanwendung die nächste Prozessphase zu starten. Dies erfolgt anhand der `updateState` Aktion, deren Funktionsweise bereits im Abschnitt 6.2.5 erläutert wurde.

6.4.5. Analyse, Lösung und Monitoring

Während der Analysephase sind auf der Seite `/si-tasks.php` die durchzuführenden Maßnahmen und die Analyseergebnisse in zwei Textareas einzutragen, wobei beliebig viele Ergänzungen und Änderungen möglich sind. Zur Speicherung der erfassten Daten wird die Aktion `updateIncidentValues` (vgl. Listing 6.25) aufgerufen. Diese schreibt die Änderungen in die Datenbank und dokumentiert sie zugleich im Verlaufsprotokoll des Incidents.

Die Aktion `updateIncidentValues` ist entsprechend flexibel implementiert, sodass sie nicht nur während der Incident Analyse, sondern auch in allen anderen Prozessphasen sowie zur Erfassung der Post Incident Review Ergebnisse eingesetzt werden kann. Aus diesem Grund werden im `switch`-Block (Zeilen 1 bis 11) für jede Datei (`$feature`) die dort vorhandenen Felder (`$fieldsArray`) definiert. Darüber hinaus werden in der Variablen `$allowedStates` die Prozessphasen festgelegt, in denen die zuvor bestimmten Felder editiert werden dürfen. Durch die `check` Funktion in Zeile 13 wird also sichergestellt, dass nur solche Felder geändert werden, die in der aktuellen Prozessphase zulässig sind. So wird beispielsweise ein Schreibzugriff auf das Feld `monDauer` (Zeile 5) unterbunden, wenn sich der Vorfall noch in der Analysephase befindet. Eine Änderung der Incidentpriorität (Zeile 2) ist hingegen in allen Prozessphasen möglich, solange der Vorfall noch nicht abgeschlossen wurde.

Zur Vorbereitung der nächsten Schritte wird in Zeile 15 das `$fieldsArray` in einen kommagetrennten String konvertiert. Das `implode`-bedingte Komma am Ende des Strings ist unerwünscht und wird daher in Zeile 16 entfernt. Zur Weiterverarbeitung der im `POST`-Array übergebenen Daten werden in einer `foreach`-Schleife (Zeilen 18 bis 20) nur diejenigen Felder in das `newValues`-Array überführt, die im `$fieldsArray` für die aktuelle Phase als zulässig definiert wurden.

Falls die Aktion von der Seite `/si-data.php` aus aufgerufen und der Anwender den Automatismus aktiviert hat, wird ab Zeile 22 die Ermittlung von Auswirkung, Priorität und Reaktionszeit durchgeführt. Da diese bereits in Listing 6.23 separat vorgestellt wurde, ist das Listing der Aktion `updateIncidentValues` entsprechend gekürzt.

Anhand des in den Zeilen 15 und 16 vorbereiteten Strings werden in den Zeilen 24 bis 26 die alten Werte vor der Änderung aus der Datenbank abgerufen. Die verschachtelte `foreach-if`-Konstruktion in den Zeilen 28 bis 34 ermittelt alle geänderten Feldinhalte. Dabei vergleicht sie alle neuen Werte (`$newValues`) mit den alten (`$old`). Unterscheidet sich ein Wertepaar, so wurde dieses Feld geändert.

Listing 6.25: Aktion updateIncidentValues (/actions/incidents.php)

```

1 switch ($_POST['feature']){
2   case "si-data": $fieldsArray = array("title","description","vorfallzeitpunkt",
3     "kunde","os","host","ip","majorExpected","hochschulstart","matrixZielsys",
4     "matrixDienste","matrixInfos","matrixAnzahl","matrixQuellsys","auswirkung",
5     "prio","reaktionszeit","firstaid"); $allowedStates="CLASS,ANAL,SOL,MON";
6     break;
7   case "si-tasks": $fieldsArray = array("vorgehensweise","analyseerg");
8     $allowedStates="ANAL,SOL,MON"; break;
9   case "si-solution": $fieldsArray = array("regelbetrieb","lsgType","
10    lsgDetails"); $allowedStates="SOL,MON"; break;
11  case "si-monitoring": $fieldsArray = array("monDauer","monIntervall","
12    monAufgabe","monAuffaellig"); $allowedStates="MON"; break;
13  case "si-review": $fieldsArray = array("revPositiv","revNegativ","
14    revOptimierung"); $allowedStates="CLOSED"; break;
15  default:
16    showAlert('error','Feature Parameter fehlt!');
17    header("Location: ".$appConfig['url']."/dashboard.php");
18    exit;
19 }
20
21 check("Status",$_POST['si'],getIncidentValue($db,$_POST['si'],"currentState"),
22   $allowedStates);
23
24 $fieldsString = implode(', ', $fieldsArray);
25 $fieldsString = rtrim($fieldsString, ', ');
26
27 foreach ($fieldsArray as $item){
28   $newValue[$item] = $_POST[$item];
29 }
30
31 // Automatische Ermittlung von Auswirkung, Priorität und Reaktionszeit
32
33 $query = "SELECT $fieldsString FROM ".$DBPREFIX."incidents WHERE id = '".$_POST
34   ['si']."'";
35 $stmt = $db->query($query);
36 $old = $stmt->fetch(PDO::FETCH_ASSOC);
37
38 foreach($newValue as $key => $value){
39   if ($value != $old[$key]){
40     $doUpdate = TRUE;
41     $updateString .= $key."=".$value." ";
42     insertHistory($db,$_POST['si'],$SESSION['userId'],$key,$old[$key],$value,
43       $SESSION['userName']." ".$SESSION['userVorname']);
44   }
45 }
46
47 if ($doUpdate) {
48   $updateString = rtrim($updateString, ' ');
49   $sql = "UPDATE ".$DBPREFIX."incidents SET $updateString WHERE id = '".$_POST[
50     'si']."'";
51   $query = $db->query($sql);
52   if (!$query){showAlert('error','Fehler beim Speichern der Änderungen.')}
53   else {showAlert('ok','Die Änderungen wurden gespeichert.')}
54 }
55 else {showAlert('error','Es lagen keine Änderungen vor.')}
56
57 header("Location: ".$SESSION['lastUrl']);

```

6. Implementierung

In diesem Fall wird das Flag `$doUpdate` gesetzt (Zeile 30), das geänderte Feld wird in `$updateString` aufgenommen (Zeile 31) und die Änderung wird mittels `insertHistory` im Verlaufsprotokoll dokumentiert, wobei der alte und der neue Wert des geänderten Feldes gespeichert werden. Ist nach Ende der Schleife das Flag `$doUpdate` *nicht* gesetzt, so wurde die Aktion aufgerufen, obwohl die Daten nicht geändert wurden. In diesem Fall (Zeile 43) wird dem Anwender mittels `setAlert` ein Hinweis gegeben.

Im Normalfall wurde jedoch mindestens eine Datenänderung vorgenommen, sodass die `if`-Bedingung in Zeile 36 den Wert `TRUE` ergibt. Nach der Bereinigung des `$updateStrings` in Zeile 37 analog zu Zeile 16 erfolgt schließlich die tatsächliche Änderung in der `incidents`-Tabelle (Zeilen 38 und 39). In Abhängigkeit vom Rückgabewert der Datenbankoperation wird in den Zeilen 40 bis 42 eine Fehler- bzw. Erfolgsmeldung gesetzt. Zuletzt wird der Anwender auf die Ursprungsseite zurückgeleitet (Zeile 45).

Nach dem Abschluss der Analyse und deren Dokumentation kann die Messung der Analysezeit beendet und die Lösungsphase gestartet werden. Beides erfolgt anhand der `updateState` Aktion, deren Funktionsweise bereits im Abschnitt 6.2.5 erläutert wurde.

Während der Lösungsphase sind auf der Seite `/si-solution.php` die Maßnahmen zur Wiederherstellung des Regelbetriebs und die im Rahmen der Incidentlösung durchgeführten Schritte in zwei Textareas zu dokumentieren. Darüber hinaus ist der Lösungstyp aus einem mittels `parseDropDown` Funktion generierten Feld auszuwählen. Die Speicherung der Daten erfolgt auch hier durch die universelle Aktion `updateIncidentValues`. Nach der Wiederherstellung des Regelbetriebs wird mit der `updateState` Aktion die Messung der Lösungszeit beendet und die Monitoringphase eingeleitet. Auf der entsprechenden Seite `/si-monitoring.php` sind neben den Monitoringaufgaben und Verantwortlichkeiten auch die Dauer und Frequenz der Überwachungsaktivitäten festzulegen. Darüber hinaus sind eventuelle Auffälligkeiten zu dokumentieren. Auch in dieser Phase werden die Eingaben durch die Aktion `updateIncidentValues` in der Datenbank gespeichert. Wie bereits beschrieben, werden die vorgenommenen Änderungen im Rahmen dieser Aktion automatisch in der `history`-Tabelle dokumentiert. Da nach dem Monitoring keine weitere Prozessphase folgt, wird der Vorfall – wie im folgenden Abschnitt beschrieben – in den Status `CLOSED` überführt.

6.4.6. Abschließen, Wiedereröffnen und Abbrechen von Security Incidents

Wurden während der Monitoringphase keine weiteren Auffälligkeiten erkannt, so kann der Vorfall geschlossen werden. Die Vorfallinformationen werden dabei direkt in der `incidents`-Tabelle durch Setzen der Felder `closedState` und `closedStamp` archiviert und der Vorfallmelder wird per E-Mail über den Abschluss des Sicherheitsvorfalls informiert. Aufgrund der zusätzlichen Schritte ist die universelle Aktion `updateState` für diesen Vorgang nicht ausreichend. Folglich wird eine separate Aktion `closeIncident` implementiert (vgl. Listing 6.26). Während in Zeile 1 durch die Funktion `checkState` sichergestellt wird, dass der Incident nur aus der Monitoringphase heraus geschlossen werden kann, wird in den Zeilen 3 bis 6 u.a. der Abschlussstatus (z.B. `SI_SUCCESS` für den erfolgreichen Incident Abschluss) in der `incidents`-Tabelle gespeichert und mit der Funktion `insertHistory` dokumentiert. Anschließend wird der Vorfallmelder durch Aufruf der `mailtoContact` Funktion über den Abschluss des Vorfalls informiert (Zeilen 8 bis 10). Zuletzt wird die reguläre `updateState` Funktion aufgerufen, welche die üblichen Operationen zum Statuswechsel ausführt und die Dauer der Incidentnachsorge berechnet (vgl. Listing 6.14 in Abschnitt 6.2.5).

Listing 6.26: Aktion `closeIncident (/actions/incidents.php)`

```

1 checkState($db,$_POST['si'],'MON');
2
3 $sql = "UPDATE ".DBPREFIX."incidents SET closedState=".$_POST['closedState'].
4       ", closedStamp=".$_POST['stamp']."' WHERE id=".$_POST['si'];
5 $query = $db->query($sql);
6
7 insertHistory($db,$_POST['si'],$SESSION['userId'],'closedState',"NONE",$_POST
8   ['closedState'],$SESSION["userName"].", ".$SESSION["userVorname"]);
9
10 $subject = "wurde geschlossen";
11 $body = "wir freuen uns, Ihnen mitteilen zu dürfen, dass der Incident [SI-".
12        $_POST['si']."] inzwischen gelöst, überwacht und soeben geschlossen wurde.
13        Für Rückfragen stehen wir gerne zur Verfügung.";
14 mailToContact($db,$_POST['si'],FALSE,$subject,$body,"SESSION","MELDER");
15
16 updateState($db,$_POST['si'],$_POST['stamp']);

```

Die für das Wiedereröffnen eines Vorfalls implementierte Aktion `reopenIncident` wird benötigt, wenn nach dem Abschließen eines Sicherheitsvorfalls weitere Auffälligkeiten bekannt werden, sodass der Vorfall erneut zu bearbeiten ist. Sie ist strukturell mit der soeben beschriebenen Aktion `closeIncident` identisch. Unterschiede bestehen hinsichtlich des Mailinhalts und der aktualisierten Datenbankfelder, wobei das `reopened`-Flag gesetzt und der `closedState` zurückgesetzt wird.

Stellt der Hotliner im Laufe der Incidentbearbeitung fest, dass es sich um einen „Fehlalarm“ handelt, kann er den Vorfall abbrechen. Dies geschieht auf ähnliche Weise wie der zuvor beschriebene Abschluss eines Vorfalls. Die Aktion `cancelIncident` kann in allen Prozessphasen der Incidentbearbeitung (Klassifikation, Analyse, Lösung und Monitoring) aufgerufen werden. Sie verzichtet jedoch auf die Anwendung der `updateState` Funktion und erledigt stattdessen alle notwendigen Vorgänge selbst.

6.4.7. Post Incident Review

Nach Abschluss eines Vorfalls kann ein Post Incident Review durchgeführt werden, das die Optimierung des SIR-Prozesses bezweckt. Als Datenbasis können folgende Quellen betrachtet werden:

- Statusseite (`/si-overview.php`)
- Verlaufsprotokoll (`/si-history.php`)
- Kommunikationsprotokoll (`/si-com.php`)

Die Statusseite fasst alle zum Vorfall final vorliegenden Informationen zusammen und gibt Auskunft über die Laufzeiten der einzelnen Phasen. Anhand des Verlaufsprotokolls ist es möglich, die Vorgehensweise chronologisch exakt nachzuvollziehen, um eventuelle Defizite im Ablauf identifizieren zu können. Darüber hinaus kann eine Analyse des Kommunikationsprotokolls Hinweise auf Hindernisse im Informationsfluss geben.

Die Webanwendung stellt für die Dokumentation der Reviewergebnisse eine eigene Seite (`/si-review.php`) bereit, die über die horizontale Incidentmenüleiste aufgerufen werden

kann. Sie beinhaltet drei Textareas zur Erfassung von positiven und negativen Erfahrungen sowie von Optimierungsvorschlägen. Da die Reviewinformationen Teil des Incidentdatensatzes sind, erfolgt die Speicherung der Eingaben durch die universelle `updateIncidentValues` Aktion. Dadurch ist sichergestellt, dass auch die Durchführung des Post Incident Reviews im Verlaufsprotokoll des Vorfalls dokumentiert ist.

6.5. Vielfältige Nutzung des Datenbestands

Die strukturierte Erfassung der Sicherheitsvorfälle innerhalb der Webanwendung bietet vielfältige Möglichkeiten zur Nutzung des Datenbestands. Zu diesem Zweck stellt die Software eine Filterkomponente mit integrierter Suchfunktion bereit. Darüber hinaus können mit wenigen Klicks kundenabhängige Auswertungen generiert werden, die über das Incidentaufkommen und die Prozessperformance während des Auswertungszeitraums informieren. Schließlich ermöglichen die Exportfunktionen (CSV und PDF) eine flexible Weiterverarbeitung der Incidentdaten in anderen Kontexten außerhalb der Webanwendung.

6.5.1. Filterung und Suche

Die im Screenshot 6.10 abgebildete Seite ermöglicht die Filterung des Datenbestandes und eine anschließende Suche in der Ergebnisliste. Im oberen Bereich der Seite können insgesamt 15 Filterparameter festgelegt werden. Zur besseren Übersicht sind die Auswahlfelder auf drei Gruppen verteilt. Während als Basisparameter sowohl der Filterzeitraum, als auch der Kunde und die Klassifikation festgelegt werden können, ermöglicht die Gruppe der erweiterten Parameter beispielsweise eine Filterung nach dem Betriebssystem oder dem Incidenttyp, der die Standard Security Incidents abbildet. Die dritte Gruppe enthält die für die Priorisierung des Vorfalls verwendeten Parameter wie die Anzahl der Zielsysteme und den Standort des Quellsystems. Auch hier werden die Dropdown-Felder mittels `parseDropdown` erst zur Laufzeit generiert. Um die weitere Verarbeitung effizient zu gestalten, werden die Bezeichnungen der Filterfelder so gewählt, dass diese mit den korrespondierenden Spaltennamen der `incidents`-Tabelle übereinstimmen.

Mit dem Absenden des Formulars werden die Parameter an die Aktion `setFilter` (vgl. Listing 6.27, Zeilen 6 bis 18) übermittelt. Diese prüft zunächst in den Zeilen 7 bis 11 die formale Korrektheit des zu filternden Zeitraums und verwendet im Fehlerfall das aktuelle Jahr. Anschließend werden die mittels `POST`-Request übergebenen Filterparameter in einer `foreach`-Schleife in das Session-Array `$filter` gespeichert (Zeile 13). Das hierbei verwendete Array `$filterFields` stammt aus der Datei `/inc/basics.php` und beinhaltet die Bezeichnungen aller Filterfelder. Da diese mehrfach benötigt werden, sind sie an zentraler Stelle definiert (Zeilen 1 bis 4) und stehen somit in allen Dateien der Filterkomponente zur Verfügung. Schließlich wird in den Zeilen 15 bis 18 der in dieser Aktion nicht benötigte Datenbankhandler geschlossen, woraufhin der Anwender mit einem Erfolgshinweis zur Ursprungsseite (`/filter.php`) geleitet wird.

Dort werden die in der Session-Variablen `$filter` zwischengespeicherten Parameter abgerufen und mit einer `UND`-Verknüpfung in die SQL-Abfrage integriert. Dabei iteriert eine `foreach`-Schleife (Zeile 21) über alle Filterfelder und ergänzt den jeweiligen Parameter – sofern

er nicht leer ist – im String `$whereFilter` (Zeile 22). Nach dem letzten Schleifendurchlauf enthält dieser String alle Filterbedingungen mit Ausnahme des Zeitraums in valider SQL-Syntax. In Zeile 25 wird die Datenbankabfrage vorbereitet, wobei sich die `WHERE`-Bedingung aus dem Filterzeitraum und den übrigen – im String `$whereFilter` enthaltenen – Parameter zusammensetzt. Schließlich werden in den Zeilen 26 und 27 die selektierten Datensätze aus der `incidents`-Tabelle abgerufen und im Array `$result` zur Weiterverarbeitung gespeichert.

In der unteren Hälfte des Screenshots 6.10 werden die Ergebnisse der Filterung tabellarisch dargestellt, wobei das jQuery Plugin `DataTables`⁵ auf die Tabelle angewendet wird. Dies ermöglicht neben der spaltenbasierten Sortierung der Einträge auch die Suche innerhalb der Tabelle.

Die Filterparameter bleiben so lange im Session-Kontext gespeichert, bis die Aktion `resetFilter` aufgerufen oder die Sitzung geschlossen wird. Verlässt der Anwender die Filterseite, um beispielsweise ein Ergebnis detailliert zu betrachten, so kann er anschließend ohne weitere Eingaben direkt zu der vorher gefilterten Ergebnisliste zurückkehren. Die selektierten Vorfälle lassen sich sowohl im PDF als auch im CSV-Format exportieren. Auf die Implementierung der Exportmodule wird im übernächsten Abschnitt 6.5.3 eingegangen.

The screenshot shows the 'Suchen & Filtern' interface of the LRZ Security Incident Management System. It features three filter panels: 'Basis-Parameter' (date range, customer, classification), 'Erweiterte Parameter' (incident type, priority, operating system, resolution, status), and 'Incident Eigenschaften' (location, services, information, count). Below the filters is a table of search results with 7 entries. The table has columns for ID, Bezeichnung, Incident Typ, Priorität, Status, MI, SfH, Kunde, Host, IP-Adresse, Betriebssystem, and Meldedatum. The first entry is 'Das ist der erste schöne Incident' with ID 1, Incident Typ 'Externer SSH Scan', and Status 'abgeschlossen'.

ID	Bezeichnung	Incident Typ	Priorität	Status	MI	SfH	Kunde	Host	IP-Adresse	Betriebssystem	Meldedatum
1	Das ist der erste schöne Incident	Externer SSH Scan	niedrig	abgeschlossen	<input checked="" type="checkbox"/>	<input type="checkbox"/>	LRZ	testhost.lrz.local	192.168.1.54	Linux	2014-04-08
14	Musterincident	Externer SSH Scan	niedrig	abgeschlossen	<input type="checkbox"/>	<input type="checkbox"/>	LRZ	linux.lrz.local	192.168.1.5	Linux	2014-05-07
15	Test vom CSIRT-User XX	Kompromittierter Webserver	mittel	abgeschlossen	<input type="checkbox"/>	<input type="checkbox"/>	LRZ	linux.lrz.local	192.168.1.5	Linux	2014-05-08
17	Test 4451	Kompromittierung virt. Server	mittel	abgeschlossen	<input type="checkbox"/>	<input type="checkbox"/>	LRZ	linux.lrz.local	192.168.4.65	Linux	2014-05-11
18	abloggingtest hacked	Kein Standard SI	hoch	abgeschlossen	<input type="checkbox"/>	<input type="checkbox"/>	LRZ	abloggingtest.netz.lrz.de	129.187.10.178	Linux	2014-05-13
39	Wieder was anderes	Kompromittierter Webserver	niedrig	abgeschlossen	<input type="checkbox"/>	<input type="checkbox"/>	LRZ	linux.lrz.local	192.168.4.65	Linux	2014-05-18
40	Business Hours Test Incident	Viren-infiziertes LRZ-System	sehr hoch	abgeschlossen	<input checked="" type="checkbox"/>	<input type="checkbox"/>	LRZ	linux.lrz.local	192.168.1.5	Linux	2014-05-15

Abbildung 6.10.: Screenshot: Suchen & Filtern (`/filter.php`)

⁵<http://www.datatables.net/>

Listing 6.27: Implementierung der Incidentfilterung

```

1 // Zentrale Definition der Filterfelder (/inc/basics.php)
2 if ($feature == "filter") {
3     $filterFields = array('kunde', 'klassifikation', 'incidentType', 'prio', 'os', '
4         lsgType', 'currentState', 'closedState', 'matrixZielsys', 'matrixQuellsys', '
5         matrixDienste', 'matrixAnzahl', 'matrixInfos');
6 }
7 // setFilter Aktion (/actions/filter.php)
8 if (strtotime($_POST['from']) == FALSE){$_SESSION['filter']['from'] = date("Y
9     ", time())."-01-01";}
10 else {$_SESSION['filter']['from'] = date("Y-m-d", strtotime($_POST['from']));}
11 if (strtotime($_POST['to']) == FALSE){$_SESSION['filter']['to'] = date("Y",
12     time())."-12-31";}
13 else {$_SESSION['filter']['to'] = date("Y-m-d", strtotime($_POST['to']));}
14 foreach($filterFields as $item){$_SESSION['filter'][$item] = $_POST[$item];}
15 $db = null;
16 setAlert('ok', 'Die Einträge wurden gemäß der Vorgaben gefiltert. ');
17 header("Location: ".$appConfig['url']."/filter.php");
18 exit;
19
20 // Abfrage der gefilterten Datensätze (/filter.php)
21 foreach($filterFields as $item){
22     if($_SESSION['filter'][$item] != "") {$whereFilter .= "AND $item = ".
23         $_SESSION['filter'][$item]."";}
24 }
25 $query = "SELECT id, title, description, incidentType, prio, currentState,
26     klassifikation, majorExpected, kunde, host, ip, os, createdStamp FROM ".DBPREFIX
27     ."incidents WHERE createdStamp BETWEEN '".$_SESSION['filter']['from']."'
28     AND '".$_SESSION['filter']['to']."' $whereFilter ORDER BY id";
29 $stmt = $db->query($query);
30 $result = $stmt->fetchAll(PDO::FETCH_ASSOC);

```

6.5.2. Auswertungen

Die Webanwendung stellt auf der Seite `/stats.php` einfache statistische Auswertungen bereit. Damit bietet sie dem CSIRT einen zahlenmäßigen Überblick über das Incidentaufkommen und den Leistungsgrad des SIR-Prozesses in Form von Key-Performance-Indikatoren. Wie auf dem Screenshot 6.11 zu erkennen ist, orientiert sich die Struktur der Auswertungskomponente exakt an den Vorgaben des Konzepts (vgl. Abschnitt 5.4.2).

Standardmäßig erstreckt sich die Auswertung auf alle über die Webanwendung erfassten Vorfälle das laufenden Kalenderjahres, wobei eine die Festlegung eines abweichenden Auswertungszeitraums und die Auswahl eines Kunden möglich ist. Diese Eingrenzung der Datenbasis wird technisch über die `setFilter` Aktion des `stats`-Moduls realisiert. Die Funktionsweise dieser Aktion entspricht im Wesentlichen ihrem Äquivalent im `filter`-Modul und wird daher nicht genauer beschrieben. Stattdessen wird auf ausgewählte Datenbankabfragen eingegangen, um zu demonstrieren, wie die auf der Auswertungsseite gezeigten Zahlen zu Stande kommen:

Listing 6.28: Überwachung der Reaktionszeiten (`/stats.php`)

```

1 SELECT
2   COUNT(*) AS anzahl, MIN(differenz) AS min,
3   MAX(differenz) AS max, AVG(differenz) AS avg
4 FROM
5   ( SELECT saldo1bh - reaktionszeit AS differenz
6     FROM incidents
7     WHERE
8       createdStamp BETWEEN '".$_SESSION['filter']['from']."' AND
9       '".$_SESSION['filter']['to']."' AND closedState='SLSUCCESS'
10    $whereFilter AND saldo1bh > reaktionszeit
11   ) sub;

```

Das in Listing 6.28 abgedruckte SQL-Statement ermittelt die absolute Anzahl der Vorfälle, bei denen die vorgegebene Reaktionszeit überschritten wurde sowie die Dauer der kleinsten, größten und durchschnittlichen Reaktionszeitüberschreitung. Dabei wird zunächst die Subquery namens `sub` in den Zeilen 5 bis 11 ausgeführt. Diese selektiert für alle erfolgreich abgeschlossenen Vorfälle, die innerhalb des gewählten Zeitraums erfasst wurden (Zeilen 8 und 9) und zudem eine Überschreitung der Reaktionszeit aufweisen (Zeile 10) die Differenz der Spalten `saldo1bh` und `reaktionszeit` (Zeile 5), also den Unterschied zwischen der tatsächlichen Reaktionszeit während der Servicezeiten und der Vorgabe in Minuten. Falls eine kundenspezifische Auswertung erstellt werden soll, enthält der String `$whereFilter` in Zeile 10 die hierfür erforderliche Ergänzung der `WHERE`-Bedingung. Ist kein Kunde gewählt, bleibt die Variable leer, sodass die Vorfälle aller Kunden berücksichtigt werden.

Die Hauptquery wendet in den Zeilen 2 und 3 mehrere SQL-Funktionen auf das Ergebnis der Subquery an und ermittelt dadurch die Anzahl der Einträge (`COUNT`), die kürzeste Dauer (`MIN`), die größte Dauer (`MAX`) und den Durchschnitt aller Zeitwerte (`AVG`). Anschließend werden die Minutenangaben in Sekunden umgerechnet und mit der `secondsToString` Funktion in einen lesbaren String konvertiert. Die Darstellung der Zahlen erfolgt in der Tabelle „Überwachung der Reaktionszeiten“ und ist auf dem Screenshot 6.11 in der rechten Spalte der zweiten Reihe zu sehen.

Listing 6.29: Auswertung nach Prioritäten (/stats.php)

```

1 SELECT
2   sub1.prio AS prio , anzahl , vorgabe , saldo1bh , saldo2bh , saldo3bh , saldo4bh
3 FROM
4   ( SELECT value AS prio
5     FROM types
6     WHERE category='prio'
7   ) sub1
8 LEFT JOIN
9   ( SELECT
10    prio , COUNT(*) AS anzahl , ROUND(AVG(reaktionszeit)) AS vorgabe ,
11    ROUND(AVG(saldo1bh)) AS saldo1bh , ROUND(AVG(saldo2bh)) AS saldo2bh ,
12    ROUND(AVG(saldo3bh)) AS saldo3bh , ROUND(AVG(saldo4bh)) AS saldo4bh
13  FROM incidents
14  WHERE
15    createdStamp BETWEEN '".$_SESSION['filter']['from'].'" AND
16    '".$_SESSION['filter']['to'].'" AND closedState='SLSUCCESS'
17    $whereFilter GROUP BY prio
18  ) sub2
19 ON sub1.prio = sub2.prio ;

```

Die ebenfalls dort in der dritten Reihe abgebildete Tabelle listet für jede Incidentpriorität die Anzahl der Vorfälle, deren durchschnittliche Reaktionszeit sowie die absolute und prozentuale Abweichung von der für die jeweilige Priorität vorgegebenen Reaktionszeit auf. Zusätzlich werden die Durchschnittswerte für Analysezeit, Lösungszeit und Nachlaufzeit für jede Priorität angegeben. Die zugehörige Datenbankabfrage besteht aus einer Hauptquery und zwei mittels `LEFT JOIN` verbundenen Subqueries. Das SQL-Statement ist in Listing 6.29 abgebildet. Während die erste Subquery `sub1` (Zeilen 4 bis 7) eine Liste aller Prioritäten aus der Typentabelle zurückliefert, werden in der zweiten Subquery `sub2` (Zeilen 9 bis 18) alle innerhalb des gewählten Zeitraums erfassten und erfolgreich abgeschlossenen Vorfälle selektiert (Zeilen 15 und 16), wobei mittels `$whereFilter` (Zeile 17) eine kundenspezifische Eingrenzung erfolgen kann. Aus diesen Datensätzen werden durch die `GROUP BY` Anweisung Prioritätsklassen gebildet (Zeile 17). Diese werden durch die Anwendung verschiedener Aggregatsfunktionen zu einem einzigen Datensatz je Prioritätsklasse zusammengefasst. So wird in den Zeilen 10 bis 12 für jede Priorität die Anzahl der Datensätze (`COUNT(*)`), die im Durchschnitt vorgegebene Reaktionszeit (`AVG(reaktionszeit)`) sowie die durchschnittliche Dauer der tatsächlichen Reaktionszeit (`AVG(saldo1bh)`), Analysezeit (`AVG(saldo2bh)`), Lösungszeit (`AVG(saldo3bh)`) und Nachlaufzeit (`AVG(saldo4bh)`) ermittelt.

Der `LEFT JOIN` der beiden Subqueries in den Zeilen 8 und 19 bewirkt eine Zusammenfassung dieser Relationen. Die `JOIN`-Relation enthält dadurch für alle `sub1`-Prioritätswerte die übereinstimmenden `sub2`-Tupel. Folglich gewährleistet der `LEFT JOIN`, dass die Ergebnisrelation auch diejenigen Prioritäten beinhaltet, die noch nicht verwendet wurden und daher in `sub2` nicht vorkommen. Die Hauptquery selektiert schließlich in Zeile 2 die Attribute der `JOIN`-Relation.

Nach der Datenbankabfrage werden die Ergebnisse für die Darstellung im Browser optimiert. Dabei werden die als Zahl codierten Prioritäten (1-4) mit der Funktion `getTypeName` in eine Bezeichnung wie „sehr hoch“ übersetzt. Darüber hinaus werden alle Zeitwerte mittels `secondsToString` in einen verständlichen String (Tage, Stunden, Minuten) konvertiert. Die Abweichung der tatsächlichen Reaktionszeit von der Vorgabe wird durch die Funkti-

on calcAbweichung als absoluter und prozentualer Wert berechnet. Eine positive Differenz bedeutet eine Überschreitung der Reaktionszeit, eine negative eine Unterschreitung. Die Funktion gibt einen formatierten HTML-String zurück, wobei zur besseren Übersicht alle Überschreitungen rot sowie Unterschreitungen grün eingefärbt sind.

Zur professionellen Berichterstattung gegenüber der LRZ-Leitung und den Kunden ist es möglich, die Auswertungsdaten gesamt und kundenspezifisch im PDF-Format zu exportieren. Daher wird im nächsten Abschnitt auf die Implementierung der Exportmodule eingegangen.

The screenshot displays the 'Auswertungen' (Reports) section of the LRZ Security Incident Management System. It includes a navigation sidebar on the left with options like 'Dashboard', 'Meldung', 'Bearbeitung', 'Suchen & Filtern', 'Auswertungen', and 'Einstellungen'. The main content area features a header with the title 'Auswertungen' and filters for date range (from 2014-01-01 to 2014-12-31) and customer (LRZ). Below this, there are two summary tables: 'Incident Klassen, Typen und Laufzeiten' and 'Überwachung der Reaktionszeiten'. The 'Incident Klassen, Typen und Laufzeiten' table is divided into two sub-tables: 'Klassifikation' and 'Incident Typ'. The 'Klassifikation' table shows counts for 'SI erfolgreich' (13), 'SI abgebrochen' (2), 'SI Duplikat' (2), 'Request beantwortet', and 'Request weitergeleitet' (1). The 'Incident Typ' table lists types like 'Externer SSH Scan' (3), 'Kompromittierter Webserver' (2), 'Kompromittierung virt. Server' (3), 'Viren-infiziertes LRZ-System' (3), and 'Kein Standard SI' (2). The 'Überwachung der Reaktionszeiten' table shows 'Reaktionszeit-Überschreitungen' (6 / 46%), 'Durchschnittliche Überschreitung' (1 Tag 0 Std. 32 Min.), 'Kleinste Überschreitung' (1 Std. 55 Min.), and 'Größte Überschreitung' (3 Tage 12 Std. 38 Min.). Below these are two more tables: 'Incident Prioritäten' and 'Incident Typen'. The 'Incident Prioritäten' table shows metrics for 'niedrig', 'mittel', 'hoch', and 'sehr hoch' priorities, including incident counts, reaction times, and deviations. The 'Incident Typen' table provides detailed metrics for each incident type, including incident counts, reaction times, analysis times, solution times, and follow-up times.

Klassifikation	Anzahl
SI erfolgreich	13
SI abgebrochen	2
SI Duplikat	2
Request beantwortet	
Request weitergeleitet	1

Incident Typ	Anzahl
Externer SSH Scan	3
Kompromittierter Webserver	2
Kompromittierung virt. Server	3
Viren-infiziertes LRZ-System	3
Kein Standard SI	2

Durchschnittliche Laufzeiten	
Reaktionszeit	13 Std. 41 Min.
Analysezeit	11 Std. 57 Min.
Lösungszeit	1 Tag 0 Std. 39 Min.
Nachlaufzeit	1 Tag 3 Std. 22 Min.

Überwachung der Reaktionszeiten	
Reaktionszeit-Überschreitungen	6 / 46%
Durchschnittliche Überschreitung	1 Tag 0 Std. 32 Min.
Kleinste Überschreitung	1 Std. 55 Min.
Größte Überschreitung	3 Tage 12 Std. 38 Min.

Priorität	Incidents	Reaktionszeit	RZ-Abweichung	Analysezeit	Lösungszeit	Nachlaufzeit
niedrig	4	13 Std. 13 Min.	+ 3 Std. 13 Min. / 32.17%	10 Std. 33 Min.	1 Tag 22 Std. 26 Min.	13 Std. 52 Min.
mittel	4	3 Std. 41 Min.	- 19 Min. / -7.92%	17 Std. 43 Min.	3 Std. 31 Min.	2 Tage 0 Std. 11 Min.
hoch	1	8 Min.	- 1 Std. 52 Min. / -93.33%	9 Std. 50 Min.	10 Std. 20 Min.	1 Tag 1 Std. 12 Min.
sehr hoch	4	1 Tag 3 Std. 36 Min.	+ 1 Tag 3 Std. 21 Min. / 10940%	8 Std. 9 Min.	1 Tag 3 Std. 35 Min.	20 Std. 37 Min.

Incident Typ	Incidents	Reaktionszeit	Analysezeit	Lösungszeit	Nachlaufzeit
Externer SSH Scan	3	15 Std. 2 Min.	7 Std. 27 Min.	1 Tag 5 Std. 8 Min.	7 Std. 25 Min.
Kompromittierter Webserver	2	4 Std. 58 Min.	14 Std. 50 Min.	2 Tage 6 Std. 20 Min.	1 Tag 5 Std. 12 Min.
Kompromittierung virt. Server	3	4 Std. 11 Min.	20 Std. 20 Min.	1 Std. 15 Min.	2 Tage 7 Std. 51 Min.
Viren-infiziertes LRZ-System	3	1 Tag 12 Std. 4 Min.	7 Std. 36 Min.	1 Tag 9 Std. 20 Min.	19 Std. 6 Min.
Kein Standard SI	2	1 Std. 9 Min.	9 Std. 50 Min.	10 Std. 20 Min.	1 Tag 1 Std. 12 Min.

Abbildung 6.11.: Screenshot: Auswertungen (/stats.php)

6.5.3. Exportmodule

Zur flexiblen Weiterverarbeitung des Datenmaterials außerhalb der Webanwendung werden zwei Exportmodule für die im Konzept vorgesehenen Dateiformate CSV und PDF implementiert. Während die Erstellung von CSV-Dateien mit der Standard-PHP-Funktion `fputcsv` erfolgt, wird für die Generierung von PDF-Dokumenten auf die frei verfügbare Klasse `TCPDF`⁶ zurückgegriffen.

Da beide Module ähnlich aufgebaut sind, richtet sich der Fokus zunächst auf die Aktionen des CSV-Moduls:

- `si-details` – CSV-Export eines Incidentdatensatzes
- `si-team` – CSV-Export der Teamdaten eines Incidents
- `si-history` – CSV-Export der Verlaufsprotokoll eines Incidents
- `si-collection` – CSV-Export mehrerer Incidentdatensätze

Die Implementierung des CSV-Moduls geht aus Listing 6.30 hervor und wird am Beispiel der Aktion `si-collection` beschrieben. Der Export mehrerer Vorfalldatensätze wird im Filtermodul benötigt. Dort werden die IDs der selektierten Datensätze in der Session-Variablen `$collection` gespeichert. Wird nun die Aktion `si-collection` in der Datei `/actions/export-csv.php` aufgerufen, prüft diese zuerst in den Zeilen 1 und 2, ob der Anwender zur Nutzung dieses Moduls berechtigt ist und ob der übergebene Token mit dem Sessiontoken übereinstimmt. Nach den Sicherheitsprüfungen springt die Anwendung anhand der `$action`-Variable innerhalb des Switch-Blocks zum passenden Case (Zeilen 18 bis 23). Anhand der im Session-Kontext zwischengespeicherten Incident-IDs werden die selektierten Datensätze aus der `incidents`-Tabelle abgerufen (Zeilen 19 bis 21) und im Array `$rows` lokal gespeichert. Darüber hinaus wird in Zeile 22 der Dateiname der exportierten Datei definiert. Dieser setzt sich aus der Bezeichnung der Aktion und dem aktuellen Zeitstempel zusammen.

Während der Switch-Block die für die jeweilige Aktion benötigte Datenbankabfrage enthält, dienen die darauffolgenden Anweisungen (Zeilen 26 bis 35) der Erzeugung der CSV-Datei. Dieser Vorgang ist für alle Aktionen des Moduls identisch und wird daher universell implementiert. Zunächst wird in den Zeilen 26 und 27 ein CSV-Header gesendet, der den zuvor festgelegten Dateinamen beinhaltet und durch die Angabe der `Content-Disposition` den Browser dazu anweist, den Dateiinhalte nicht im Browserfenster darzustellen, sondern als Download zu behandeln. Damit Umlaute in der CSV-Datei korrekt dargestellt werden, wird der Zeichensatz `UTF-8` verwendet. Zudem wird in Zeile 28 ein Byte Order Mark zur Definition der Byte-Reihenfolge gesendet. Der PHP-Outputstream wird in Zeile 30 erzeugt. Anschließend werden die Spaltenbezeichnungen in Form einer Kopfzeile mit der Standard-PHP-Funktion `fputcsv` an den Outputstream übergeben (Zeile 31). Da die Spaltennamen jedoch nicht als separater Datensatz vorliegen, wird dieser mit der PHP-Funktion `array_keys` aus den Schlüsselbezeichnungen des ersten Datensatzarrays `$rows[0]` erzeugt.

Abschließend läuft eine `foreach`-Schleife (Zeile 33 bis 35) über alle Einträge des Datensatzarrays und fügt die Datensätze mit der Funktion `fputcsv` zeilenweise dem Outputstream hinzu (Zeile 34). Diese Funktion erwartet als ersten Parameter einen Dateihandler bzw. den

⁶<http://sourceforge.net/projects/tcpdf/>

Outputstream, an zweiter Position das Datenarray und als dritten Parameter ein Feldtrennzeichen (hier: Semikolon). Da der Datensatz `$row` vom CSV-Format nicht unterstützte Zeilenumbrüche enthalten kann, werden mit der PHP-Funktion `str_replace` alle Tab-, Newline- und Return-Befehle durch Leerzeichen ersetzt, bevor der Datensatz in die CSV-Datei geschrieben wird. Nach dem letzten Schleifendurchlauf wird mit dem Beenden des PHP-Skripts auch der Outputstream geschlossen. Daraufhin kann der Anwender die CSV-Datei auf seinem lokalen System speichern und beispielsweise mit Microsoft Excel verarbeiten.

Listing 6.30: Modul `export-csv (/actions/export-csv.php)`

```

1 checkUser("CSIRT,ROOT");
2 checkToken($_REQUEST['token']);
3
4 switch($_REQUEST['action']){
5
6     case "si-details":
7         // gekürzt
8         break;
9
10    case "si-team":
11        // gekürzt
12        break;
13
14    case "si-history":
15        // gekürzt
16        break;
17
18    case "si-collection":
19        $query = "SELECT * FROM ".DBPREFIX."incidents WHERE id IN (".$_SESSION["
20            collection"].")";
21        $stmt = $db->query($query);
22        $rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
23        $csvFileName = 'si-collection-'.date("Ymd-His",time()).'.csv';
24    break;
25}
26 header("Content-Type: text/csv; charset=utf-8");
27 header("Content-Disposition: attachment; filename=$csvFileName");
28 echo "\xEF\xBB\xBF"; // UTF-8 BOM
29
30 $output = fopen('php://output', 'w');
31 fputcsv($output, array_keys($rows[0]), ";");
32
33 foreach ($rows as $row) {
34     fputcsv($output, str_replace(array("\t", "\n", "\r"), array(" ", " ", " "), $row),
35         ";");
36 }

```

Im Gegensatz zum CSV-Format, das u.a. für den Datenimport in ein anderes Softwaresystem geeignet ist, ist das PDF-Format immer dann zu bevorzugen, wenn druckfähig formatierte Dokumente benötigt werden. Die Webanwendung unterstützt sowohl den PDF-Einzelexport eines Vorfalls (`si-details`) auf Basis der Incidentstatusseite, als auch den PDF-Massenexport von selektierten Incidents (`si-collection`) im Filtermodul. Darüber hinaus lassen sich auch Auswertungen (`stats`) im PDF-Format exportieren. Obwohl es sich beim *Portable Document Format* um ein verhältnismäßig komplexes Dateiformat handelt und zudem eine Formatierung der Rohdaten notwendig ist, weist das PDF-Modul eine grundlegende Ähnlichkeit zum CSV-Äquivalent auf.

6. Implementierung

Durch den Einsatz der Klasse TCPDF rückt die Komplexität des Dateiformats jedoch in den Hintergrund. Da diese Klasse HTML-Code in PDF-Syntax konvertieren kann, werden zur Formatierung der Rohdaten zwei HTML/CSS-Vorlagen für Incidents und Auswertungen erstellt. Anhand dieser Templates generiert die bereits eingesetzte Klasse `rainTPL` zur Laufzeit den HTML-Code und übergibt diesen zur Konvertierung an die PDF-Klasse. Bei der Erstellung der Template-Dateien ist zu berücksichtigen, dass der HTML- und CSS-Befehlssatz des TCPDF-Parsers stark eingeschränkt ist. Dennoch ist einfach strukturiertes, aber professionell wirkendes Dokumentenlayout realisierbar.

Anhand der zum vorherigen CSV-Beispiel korrespondierenden PDF-Aktion `si-collection` (vgl. Listing 6.31) wird auf Gemeinsamkeiten und Unterschiede der beiden Exportmodule eingegangen: Während die CSV-Aktion (vgl. Listing 6.30, Zeilen 18 bis 23) nur die Datensätze aus der Datenbank ausliest und den Dateinamen festlegt, generiert die PDF-Aktion darüber hinaus fertig formatierte Dokumentseiten aus den Rohdaten und fügt diese dem PDF-Objekt hinzu. Da die PDF-Aktion `si-collection` ein Dokument mit den Incidentstatusseiten der selektierten Vorfälle erzeugen soll, werden zunächst die als String übergebenen Incident IDs in ein Array konvertiert (Zeile 3) und anschließend an die `foreach`-Schleife (Zeile 5 bis 9) übergeben.

Die Abfrage der Incidentdaten aus der Datenbank und die Erzeugung des HTML-Codes mit der Templateklasse `rainTPL` ist in die lokale Hilfsfunktion `prepareIncidentForPdf` ausgelagert. Diese wird in Zeile 6 – also innerhalb der Schleife – aufgerufen. Anschließend wird dem `$pdf`-Objekt in Zeile 7 eine neue Seite hinzugefügt, woraufhin in Zeile 8 mit der `writeHTML`-Methode der Klasse TCPDF der zuvor generierte HTML-Code in PDF-Befehle konvertiert und in das `$pdf`-Objekt geschrieben wird.

Nach dem letzten Schleifendurchlauf wird in Zeile 11 analog zur CSV-Aktion der Dateiname festgelegt. Im aktionsübergreifenden Teil des PDF-Moduls wird schließlich mit dem Befehl `$pdf->Output($pdfFileName, 'D');` die PDF-Datei erzeugt und an den Browser übergeben.

Listing 6.31: Aktion `si-collection` im Modul `export-pdf` (`/actions/export-pdf.php`)

```
1 case "si-collection":
2
3     $collection = explode(',', $_SESSION['collection']);
4
5     foreach ($collection as $item){
6         $pdfContent = prepareIncidentForPdf($db, $tpl, $item);
7         $pdf->AddPage();
8         $pdf->writeHTML($pdfContent, true, false, true, false, '');
9     }
10
11     $pdfFileName = 'si-collection-'.date("Ymd-His", time()).'.pdf';
12
13 break;
```

Die TCPDF-Klasse ist derart konfiguriert, dass allen Dokumenten automatisch eine Kopf- und Fußzeile hinzugefügt wird. Während sich die Kopfzeile aus dem Titel des Reports, dem Erstellungsdatum und dem Autor zusammensetzt, beinhaltet die Fußzeile die aktuelle sowie die Gesamtseitenzahl. Darüber hinaus ist im Sinne eines einheitlichen Erscheinungsbildes auf jedem Report das LRZ-Logo abgebildet. Ein Musterdokument ist im Anhang D abgebildet.

7. Prototypische Anwendung der Software

Nach Abschluss der Implementierung wird nun die Funktionsweise der Software durch prototypische Anwendung anhand eines fiktiven Security Incidents demonstriert. In diesem Beispiel hat ein Mitarbeiter des LRZ festgestellt, dass die von ihm betreute WordPress Umgebung gehackt wurde. Daraufhin ruft er die im Intranet verlinkte Security Incident Managementsoftware auf und authentifiziert sich mit seiner Kennung. Nun navigiert er zum Meldeformular (Screenshot 7.1) und trägt dort die ihm bekannten Informationen zum Sicherheitsvorfall ein. Darüber hinaus ergänzt der Incidentmelder die zumeist vorausgefüllten Kontaktfelder um fehlende Angaben wie seine Handynummer. Durch das Absenden des Formulars wird ein neuer Incident angelegt und die Messung der Reaktionszeit beginnt.

The screenshot shows a web interface for reporting a security incident. The header includes the system name 'LRZ Security Incident Management System' and the user 'Markus Incidentmelder'. The main content area is titled 'Meldung eines Security Incidents' and contains a section for 'Informationen zum Vorfall'. Below this, there are several form fields: 'Titel*' (Gehackte WordPress Installation), 'Beschreibung*' (Die WordPress basierte Website des Projekts XYZ wurde gehackt...), 'Vorfallzeitpunkt' (zwischen 09.06. und 11.06.2014), 'Kunde' (LMU), 'Betriebssystem' (Linux), and 'Hostname' (wordpress.lmu.de). A second section, 'Kontaktdaten des Vorfallmelders', contains fields for 'Vorname*' (Markus), 'Nachname*' (Incidentmelder), 'E-Mail*' (mail@aaron-schweiger.de), 'Telefon*' (+49-89-35831-5678), 'Mobil' (+49-173-9977213), 'Anmerkungen' (Falls ich nicht erreichbar bin, weiß auch Herr Mustermann Bescheid.), and 'Kennung*' (di29len). At the bottom, there are two buttons: 'abbrechen' and 'Incident melden'.

Abbildung 7.1.: Screenshot: Meldung eines Security Incidents

7. Prototypische Anwendung der Software

Sobald der Incident in der Datenbank gespeichert wurde, versendet die Webanwendung automatisch eine Bestätigungsmail an den Vorfallemitter (Abbildung 7.2) und benachrichtigt das CSIRT (Abbildung 7.3) über den Eingang eines neuen Incidents.

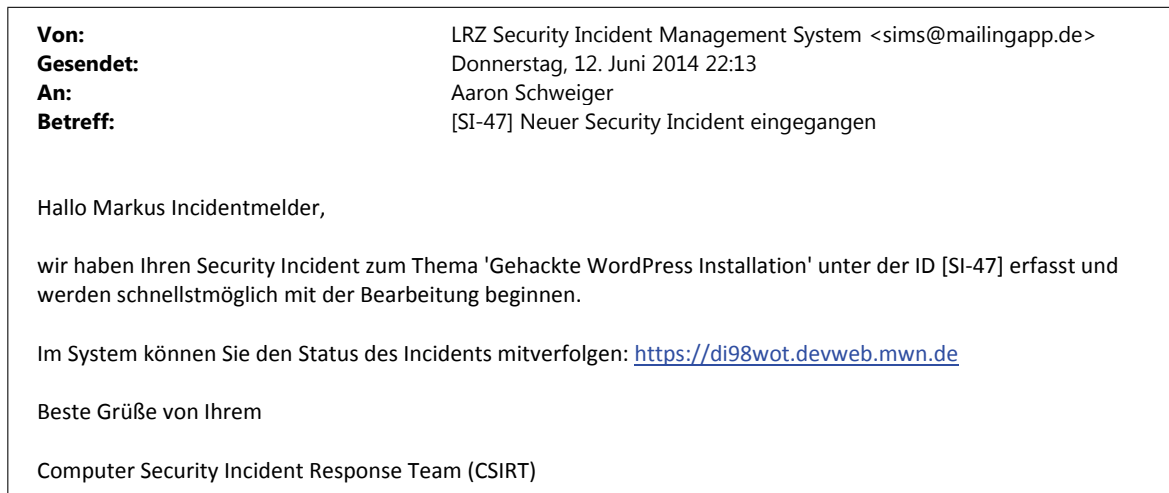


Abbildung 7.2.: Bestätigungsmail an den Vorfallemitter



The screenshot displays the LRZ Security Incident Management System interface. On the left is a navigation sidebar with options like 'Dashboard', 'Meldung', 'Bearbeitung', 'Suchen & Filtern', 'Auswertungen', and 'Einstellungen'. The main area is titled 'Dashboard' and contains three primary sections:

- Aktuelle Security Incidents:** A table listing incidents with columns for ID, Bezeichnung, Priorität, Status, and Incident Typ. Incident 47 is highlighted in red with the status 'neu'.
- Aktuelle Ereignisse:** A list of recent events with details such as 'currentState: MON', 'currentState: SOL', 'os: LINUX', and 'host: wordpress.lmu.de'.
- Incident Statistik der letzten 30 Tage:** A summary card showing metrics for new incidents, low, high, and very high priority counts, along with percentage changes.

A dropdown menu is open over the incident table, showing progress bars for four incidents: SI-47 (0%), SI-46 (40%), SI-45 (40%), and SI-44 (80%).

Abbildung 7.4.: Screenshot: Dashboard aus CSIRT-Sicht

Nach dem Erhalt der E-Mail meldet sich der Hotliner in der Webanwendung an und wird vom System als CSIRT-Mitglied erkannt. Daher stehen ihm alle Funktionen für die Incidentbearbeitung und darüber hinaus ein erweitertes Dashboard (Screenshot 7.4) zur Verfügung. Dieses informiert über die neuen und in Bearbeitung befindlichen Vorfälle sowie über aktuelle Ereignisse und gibt einen Überblick über das Incidentaufkommen der letzten 30 Tage. Der Zugriff auf den neu gemeldeten Incident kann sowohl über die Dashboardtabelle, in der der neue Incident rot markiert ist, als auch über den Schnellzugriff in der Kopfleiste erfolgen.

7. Prototypische Anwendung der Software

The screenshot displays the LRZ Security Incident Management System interface. The top navigation bar includes a home icon, the system name, a notification bell with a red '4', a 'SID' button, and a user profile for 'Aaron Schweiger'. The left sidebar contains menu items: 'Dashboard', 'Meldung', 'Bearbeitung' (highlighted), 'Suchen & Filtern', 'Auswertungen', and 'Einstellungen'. The main content area is titled 'SI-47: Gehackte WordPress Installation' and 'Neuen Vorfall prüfen'. It features two tables: one for incident details and another for contact information.

ID	47
Titel	Gehackte WordPress Installation
Beschreibung	Die WordPress basierte Website des Projekts XYZ wurde gehackt. Im Ordner "uploads" wurden Dateien gelöscht. Darüber hinaus wurden Seiteninhalte manipuliert. Außerdem wurde der WordPress-Admin-Account manipuliert, weswegen aktuell kein Zugriff auf das Backend möglich ist.
Vorfallzeitpunkt	zwischen 09.06. und 11.06.2014
Kunde	LMU
Betriebssystem	Linux
Hostname	wordpress.lmu.de
IP-Adresse	
Meldezeitpunkt	2014-06-12 16:36:08

Name	Markus Incidentmelder
E-Mail	mail@aaron-schweiger.de
Telefon	+49-89-35831-5678
Mobil	+49-173-9977213
Anmerkungen	Falls ich nicht erreichbar bin, weiß auch Herr Mustermann Bescheid.
Kennung	di29len

Below the tables, there are three classification options under the heading 'Bitte entscheiden Sie':

- Neuer Security Incident**: Es handelt es sich um einen neuen, noch nicht erfassten Sicherheitsvorfall vom Typ:
- Incident Duplikat**: Es handelt sich um ein Duplikat. Dieser Vorfall wurde bereits gemeldet:
- Information Request**: Es handelt sich um eine Anfrage, die beantwortet oder weitergeleitet werden soll.

Abbildung 7.5.: Screenshot: Initiale Klassifikation

Der neue Vorfall ist zunächst initial zu klassifizieren (Screenshot 7.5). Auf Grundlage der gemeldeten Informationen entscheidet der Hotliner, ob es sich um einen noch nicht erfassten Sicherheitsvorfall, um einen bereits gemeldeten Incident oder um einen Information Request handelt. Im hier betrachteten Beispiel liegt ein neuer Security Incident vor. Der Hotliner wählt den Incidenttyp „Kompromittierter Webserver“.

Für diese Vorfallart existiert ein Standard Security Incident Template. Dieses wird nun auf den aktuellen Vorfall angewendet, sodass im Folgenden zahlreiche Formularfelder bereits mit den Standardwerten des SSI-Templates vorbelegt sind. Bei normalen, individuell zu behandelnden Sicherheitsvorfällen sind die Felder anfangs leer und im Prozessverlauf manuell auszufüllen.

Im Rahmen der weiteren Klassifikation (Screenshot 7.6) kann der Hotliner fehlende Angaben – zum Beispiel die IP-Adresse – ergänzen. Darüber hinaus wird festgelegt, ob ein Major Incident zu erwarten und ob der Hochschulstart-Dienst betroffen ist. Die für die automatische Ermittlung der Priorität benötigten Inputparameter sind hier aufgrund des SSI-Templates bereits ausgewählt. Wird die Automatik durch den Hotliner deaktiviert, so kann er Auswirkung, Priorität und Reaktionszeit auch manuell bestimmen. Nach der Festlegung der Erstmaßnahmen speichert der Hotliner seine Eingaben, wobei sämtliche Änderungen automatisch im Verlaufsprotokoll dokumentiert werden. Anschließend kann mit der Zusammenstellung des Security Incident Teams begonnen werden.

LRZ Security Incident Management System

SI-ID: Aaron Schweiger

SI-47: Gehackte WordPress Installation

OK! Neuer Status: CLASS

Übersicht **Vorfall** Team Maßnahmen Lösung Monitoring Review Kommunikation Verlauf

Informationen zum Vorfall

Titel: Gehackte WordPress Installation

Beschreibung: Die WordPress basierte Website des Projekts XYZ wurde gehackt. Im Ordner "uploads" wurden Dateien gelöscht. Darüber hinaus wurden Seiteninhalte manipuliert. Außerdem wurde der WordPress-Admin-Account manipuliert, weswegen aktuell kein Zugriff auf das Backend möglich ist.

Vorfallzeitpunkt: zwischen 09.06. und 11.06.2014

Kunde: LMU

Betriebssystem: Linux

Hostname: wordpress.lmu.de

IP-Adresse: 123.124.125.126

Klassifizieren Sie den Sicherheitsvorfall

Incident Typ: Kompromittierter Webserver

Major Incident zu erwarten?: Nein

Hochschulstart betroffen?: Nein

Standort Zielsystem: extern

Standort Quellsystem: extern

Betroffene Dienste: keine kritischen Dienste

Betroffene Informationen: keine vertraulichen Infos

Anzahl der betroffenen Systeme: 1

Automatik: Ja! Auswirkung, Priorität und Reaktionszeit automatisch berechnen und ändern.

Auswirkung: gering

Priorität: niedrig

Reaktionszeit: 10 Std.

Legen Sie die Erstmaßnahmen fest

Erstmaßnahmen:

- Seitenbetreiber benachrichtigen, falls der Hinweis nicht von diesem kam
- Webserver deaktivieren, falls dies notwendig ist, um weiteren Schaden abzuwenden
- httdocs-Ordner + DB-Dump archivieren
- Kennungs- und DB-Passwort zurücksetzen

speichern → Team zusammenstellen

Security Incident

Klassifikation: niedrig

Typ: Kompromittierter Webserver

Kunde: LMU

System: Linux

Host: wordpress.lmu.de

Empfehlungen:

> Beteiligte auf dem Laufenden halten!

Aktionen:

- E-Mail senden
- Gesprächsnotiz erstellen
- PDF Export
- CSV Export (SI-Daten)
- CSV Export (Team-Daten)
- CSV Export (Incident-History)
- Status ändern
- Incident abbrechen

Abbildung 7.6.: Screenshot: Weitere Klassifikation und Priorisierung

7. Prototypische Anwendung der Software

Auf der Teamseite (Screenshots 7.7 und 7.8) sind zunächst die Rollen „Hotliner“ und „Security Incident Coordinator“ zu besetzen. Hierfür stehen alle CSIRT-Mitglieder in Dropdown-Feldern zur Auswahl. Darüber hinaus kann das Team gemäß Screenshot 7.9 um zusätzliche LRZ-Mitarbeiter – beispielsweise Administratoren oder Abteilungsleiter – erweitert werden. Für jedes Teammitglied stehen anhand selbsterklärender Symbole Funktionen zum Senden einer E-Mail, zur Erfassung einer Gesprächsnotiz, zum Editieren der Kontaktdaten und zum Löschen des Teammitglieds bereit. Änderungen an der Teamzusammensetzung werden ebenfalls automatisch im Verlaufsprotokoll dokumentiert. Wurden zumindest Hotliner und SIC festgelegt, kann die nächste Prozessphase gestartet werden (Screenshot 7.10).

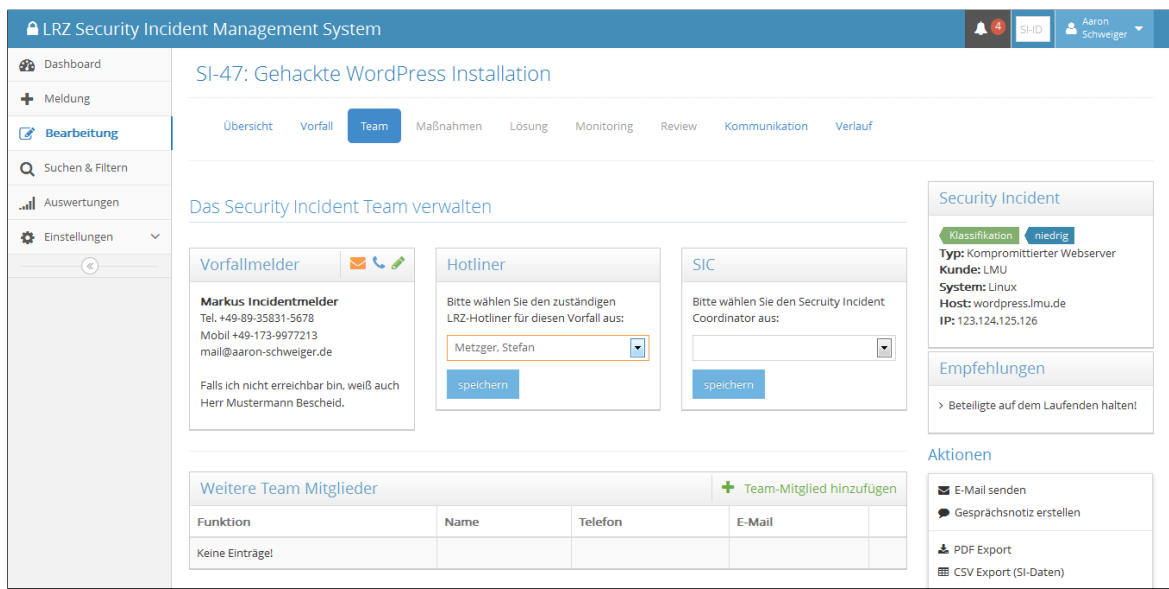


Abbildung 7.7.: Screenshot: Auswahl des Hotliners

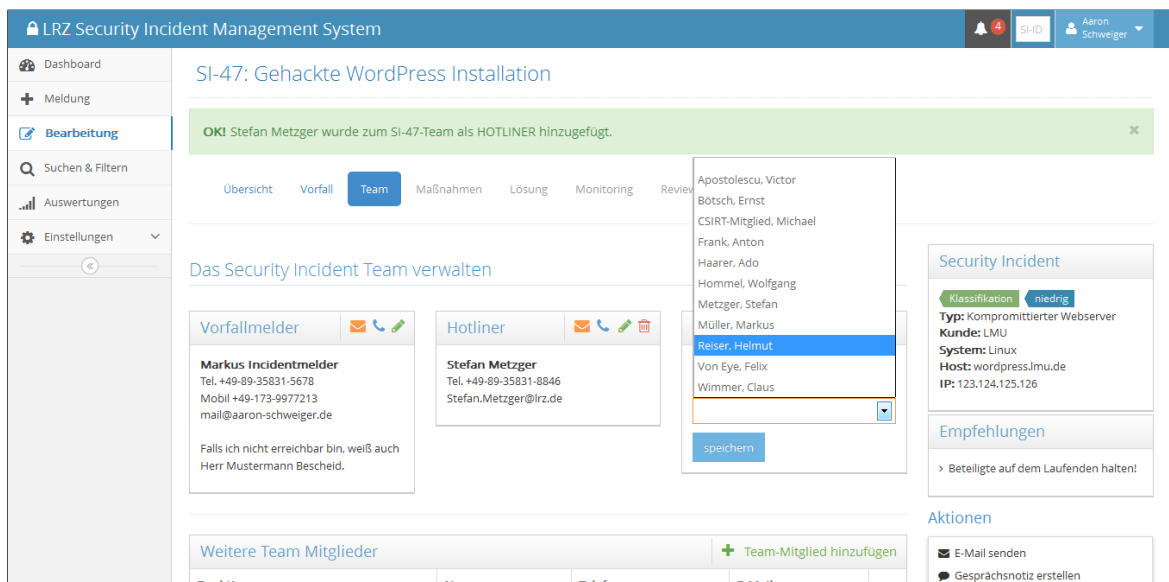


Abbildung 7.8.: Screenshot: Auswahl des Security Incident Coordinators

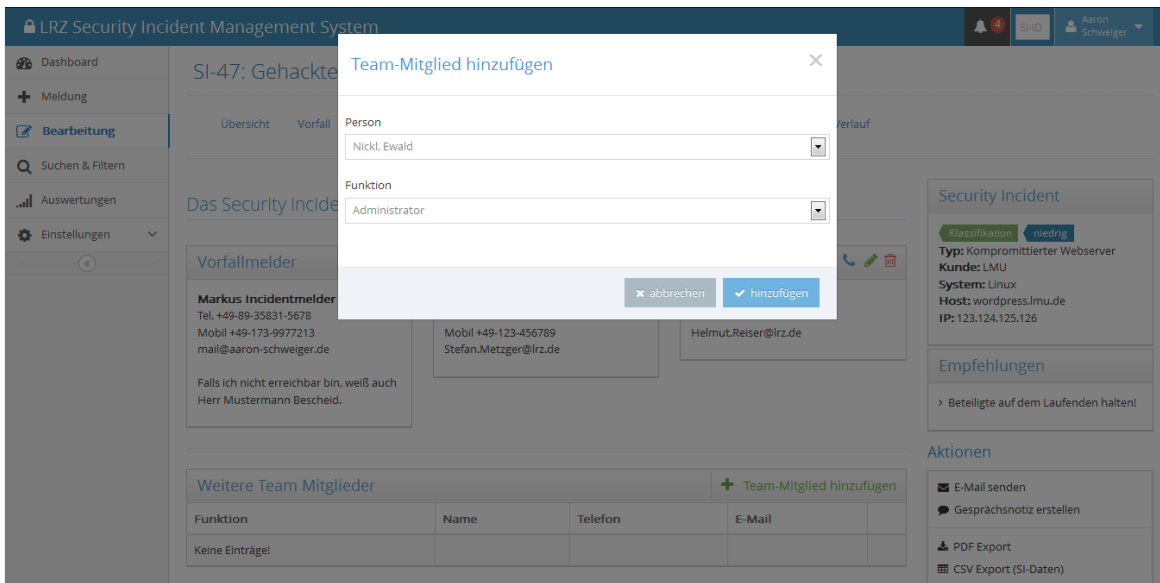


Abbildung 7.9.: Screenshot: Hinzufügen weiterer Teammitglieder

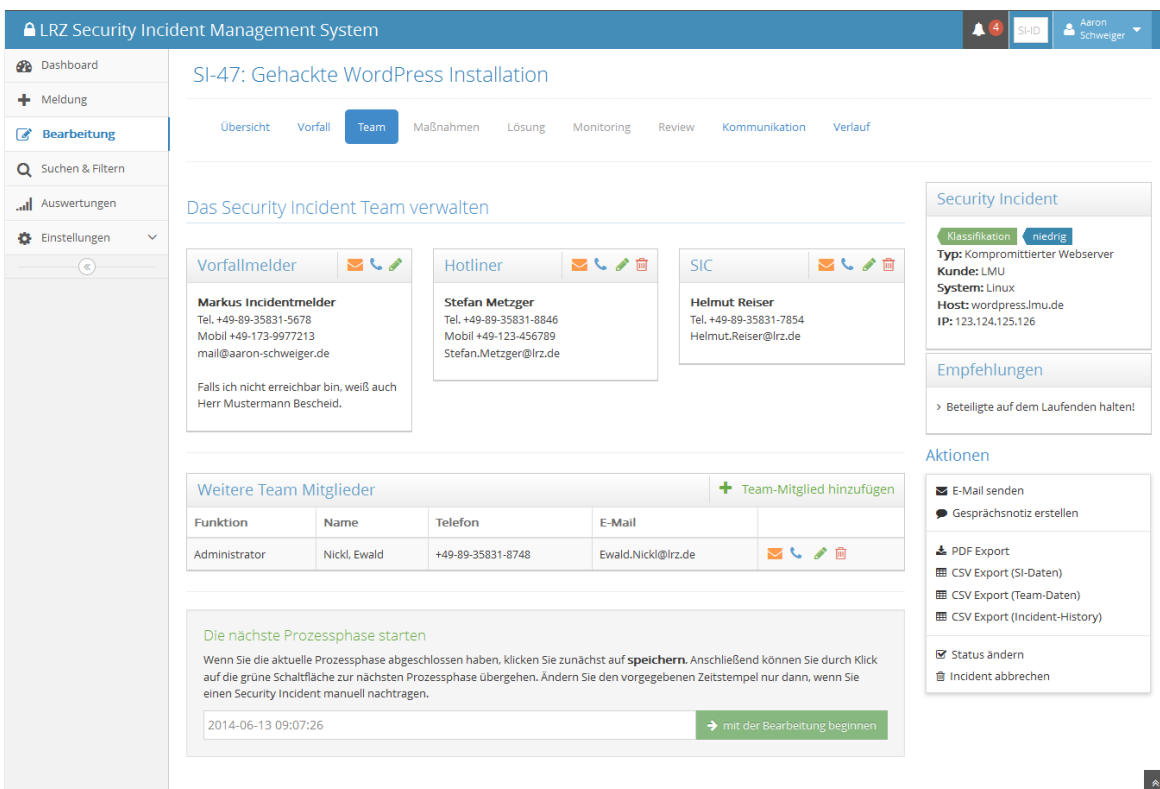


Abbildung 7.10.: Screenshot: Verwaltung des Security Incident Teams

Mit dem Beginn der Analysephase endet die Messung der Reaktionszeit. Zu diesem Zeitpunkt soll der Hotliner sowohl die Vorfalldaten, als auch die Teamzusammensetzung und die festgelegten Erstmaßnahmen per E-Mail an die CSIRT-Mitglieder senden. Hierfür bietet es sich an, die Mailfunktion (Screenshot 7.11) des Kommunikationsmoduls zu verwenden.

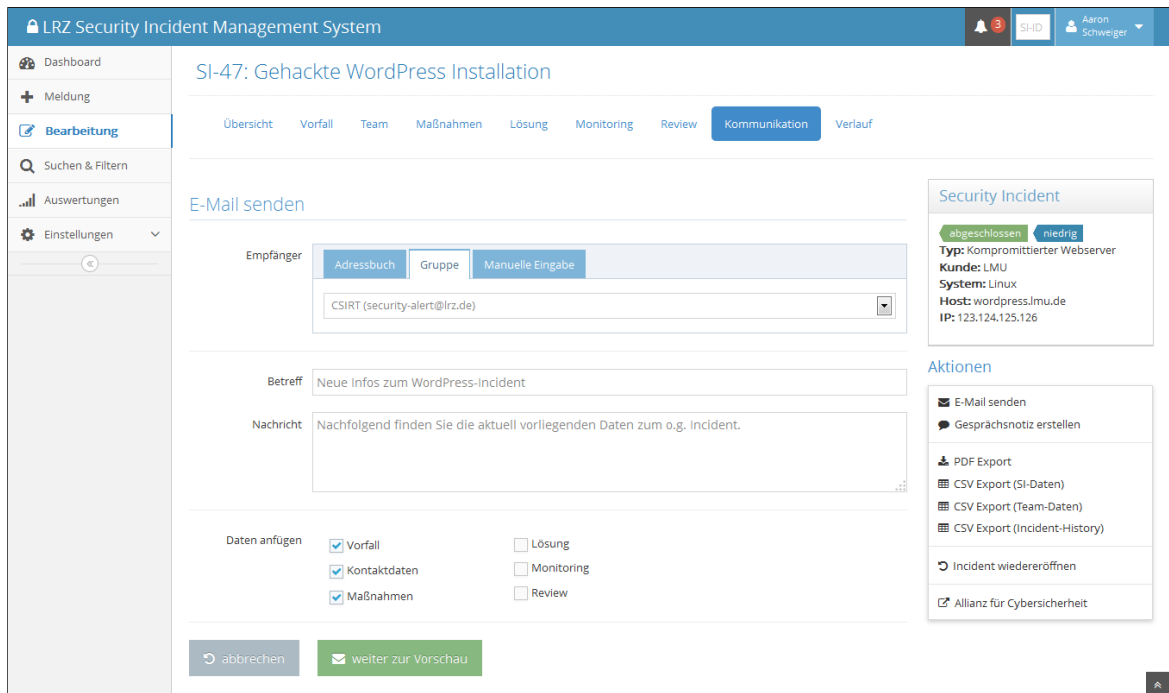


Abbildung 7.11.: Screenshot: Mailfunktion

Während der Analysephase sind zunächst die durchzuführenden Maßnahmen festzulegen und anschließend die Analyseergebnisse zu dokumentieren (Screenshot 7.12). Da es sich bei dem Beispiel um einen Standard Security Incident handelt, sind die für diesen Fall üblichen Analysemaßnahmen bereits voreingestellt. Die Eingaben lassen sich beliebig oft editieren, wobei jede Änderung nachvollziehbar dokumentiert wird.

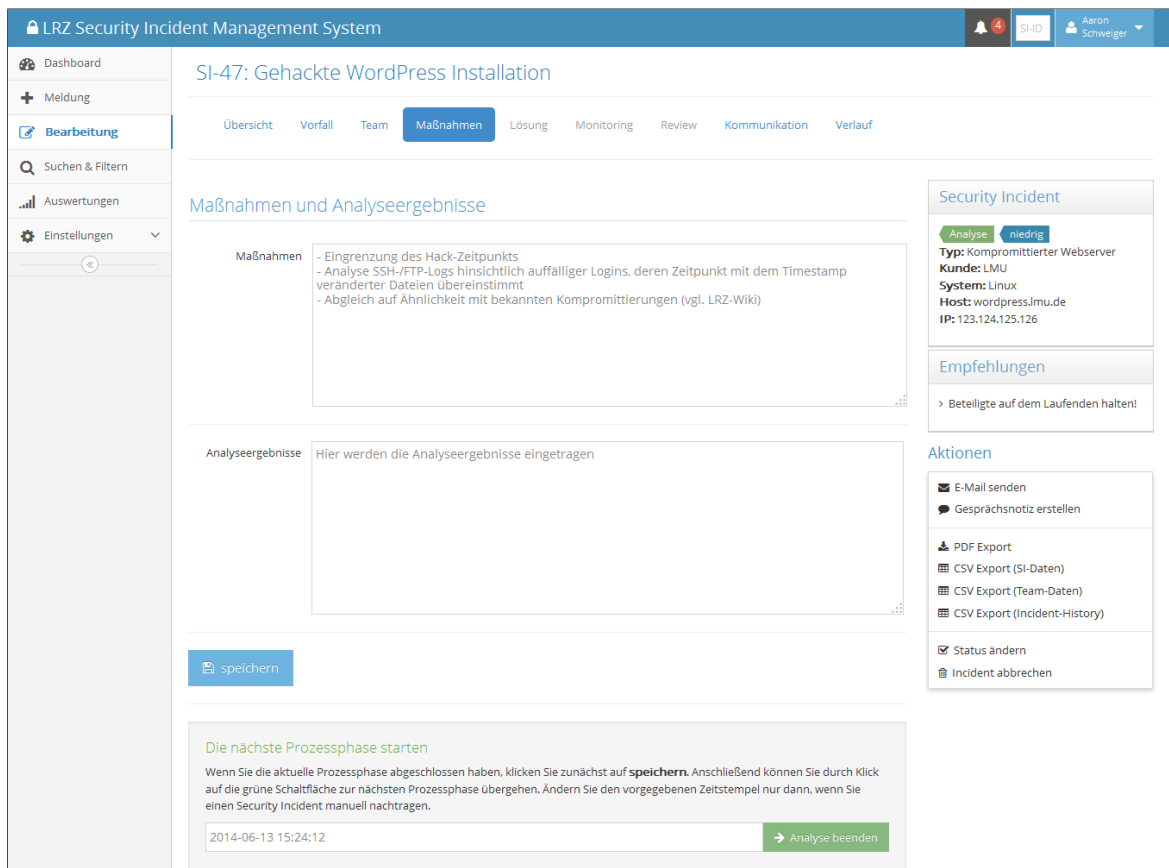


Abbildung 7.12.: Screenshot: Analysephase

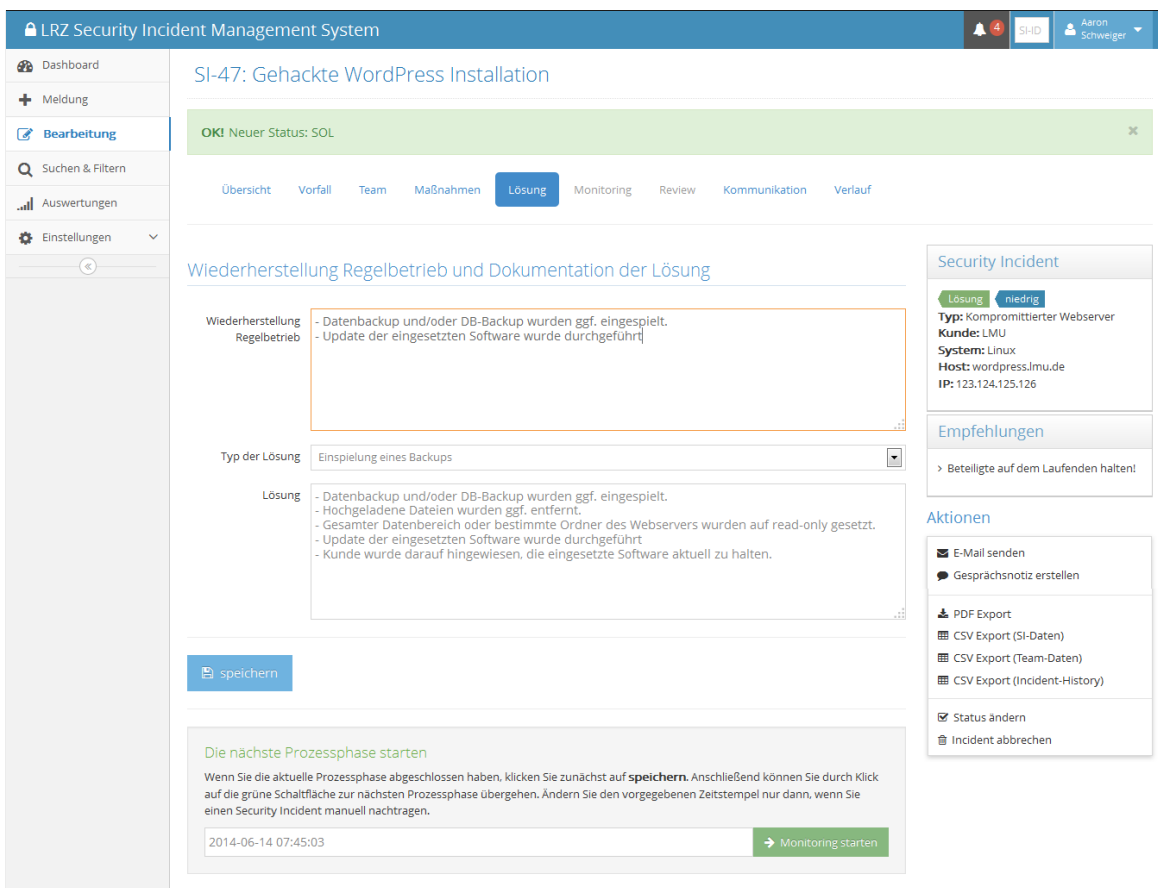


Abbildung 7.13.: Screenshot: Lösungsphase

Nach Abschluss der Analysearbeiten erfolgt der Wechsel in die Lösungsphase, wobei der Zeitstempel des Prozessphasenwechsels angepasst werden kann. Dies ist ausnahmsweise erforderlich, wenn ein Sicherheitsvorfall manuell nachgetragen wird. Mit dem Beginn der Lösungsphase endet die Messung der Analysedauer. Während dieser Phase sind die Wiederherstellung des Regelbetriebs und die Lösung des Vorfalls zu dokumentieren (Screenshot 7.13). Darüber hinaus ist der Lösungstyp aus einem Dropdown-Feld auszuwählen, um eine spätere Filterung aller Vorfälle des gleichen Lösungstyps zu ermöglichen.

In der Sidebar am rechten Rand stehen während allen Prozessphasen verschiedene Kommunikations- und Exportfunktionen bereit. Dort könnte auch der Abbruch eines Incidents bewirkt werden, falls sich ein „Fehlalarm“ herausstellen sollte. Die Sidebar beinhaltet außerdem eine Zusammenfassung der wesentlichen Vorfalldaten sowie ein Empfehlungswidget, das den Hotliner beispielsweise daran erinnert, den Vorfall an die LRZ-Leitung zu melden, falls der Incident eine hohe Priorität aufweist.

Ist die Incidentlösung abschließend dokumentiert, wird die Messung der Lösungsdauer beendet. Gleichzeitig erfolgt ein Wechsel in die Monitoringphase.

7. Prototypische Anwendung der Software

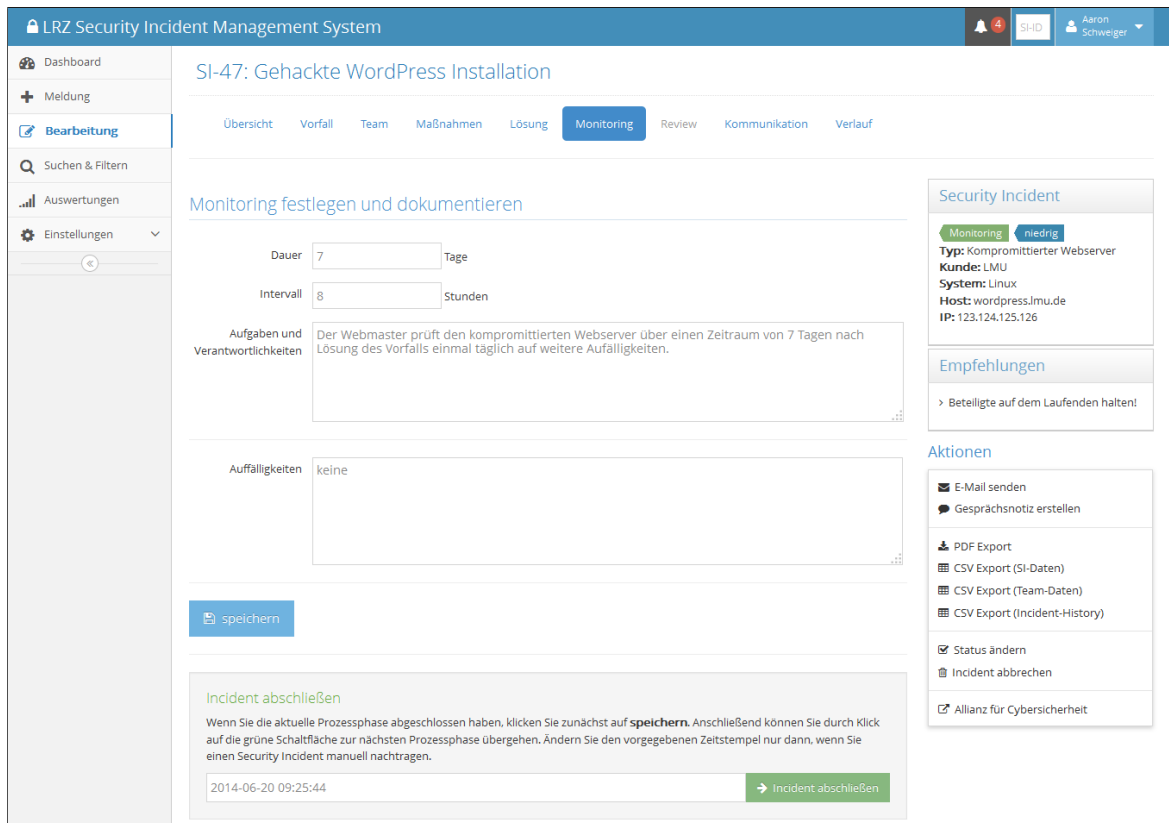


Abbildung 7.14.: Screenshot: Monitoringphase

Während dieser letzten SIR-Prozessphase werden geeignete Monitoringmaßnahmen durchgeführt und dokumentiert (Screenshot 7.14). Das in die Webanwendung integrierte Formular zur Meldung des Vorfalls an die Allianz für Cyber-Sicherheit ist in der Monitoringphase und nach Abschluss des Vorfalls über das Aktionsmenü in der Sidebar erreichbar. Im Beispiel haben sich während des Monitorings keine weiteren Auffälligkeiten ergeben, sodass der Hotliner den Vorfall nun abschließen kann. Der Incidentabschluss löst einerseits eine E-Mail an den Vorfallmelder (Abbildung 7.15) aus und beendet andererseits die Messung der Nachlaufdauer.

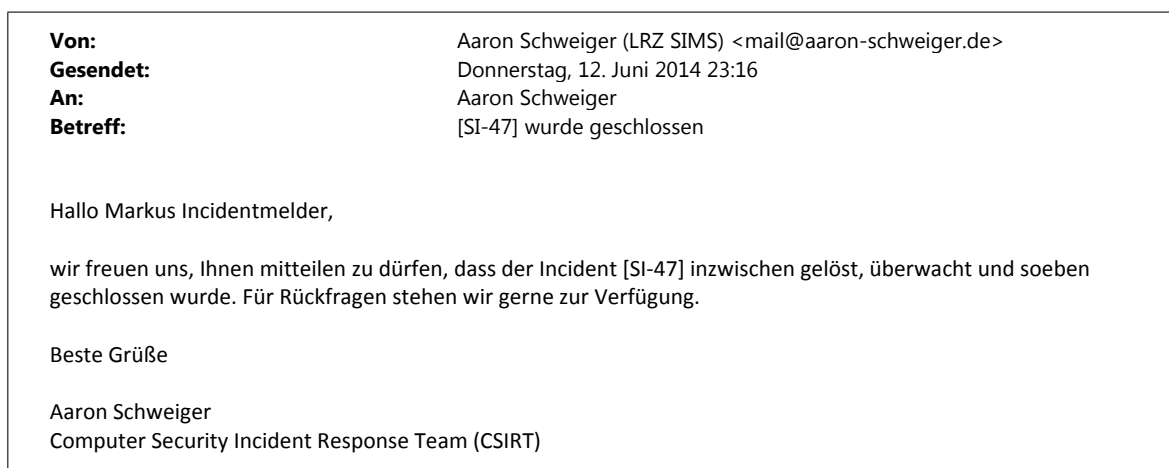


Abbildung 7.15.: Benachrichtigung des Vorfallmelders über den Incidentabschluss

Nach dem Abschluss des Vorfalls können keine Änderungen mehr an den Vorfalldaten vorgenommen werden. Falls sich nachträgliche Ergänzungen ergeben und neue Auffälligkeiten bekannt werden, muss der Incident wiedereröffnet werden. Die entsprechende Funktion ist im Aktionsmenü der Sidebar enthalten (Screenshot 7.16).

The screenshot displays the 'Incident-Details' page for 'SI-47: Gehackte WordPress Installation'. The interface includes a sidebar with navigation options like 'Dashboard', 'Meldung', 'Bearbeitung', 'Suchen & Filtern', 'Auswertungen', and 'Einstellungen'. The main content area shows the incident title, a green notification bar indicating 'OK! Neuer Status: CLOSED', and a breadcrumb trail: 'Übersicht > Vorfall > Team > Maßnahmen > Lösung > Monitoring > Review > Kommunikation > Verlauf'. Below this, the 'Incident-Daten' section provides a table of key information:

ID	47
Titel	Gehackte WordPress Installation
Beschreibung	Die WordPress basierte Website des Projekts XYZ wurde gehackt. Im Ordner "uploads" wurden Dateien gelöscht. Darüber hinaus wurden Seiteninhalte manipuliert. Außerdem wurde der WordPress-Admin-Account manipuliert, weswegen aktuell kein Zugriff auf das Backend möglich ist.
Vorfallzeitpunkt	zwischen 09.06. und 11.06.2014
Kunde	LMU
Betriebssystem	Linux
Hostname	wordpress.lmu.de
IP-Adresse	123.124.125.126

To the right, the 'Security Incident' summary shows: 'abgeschlossen' (green), 'niedrig' (blue), 'Typ: Kompromittierter Webserver', 'Kunde: LMU', 'System: Linux', 'Host: wordpress.lmu.de', and 'IP: 123.124.125.126'. Below this is an 'Aktionen' menu with options like 'E-Mail senden', 'Gesprächsnotiz erstellen', 'PDF Export', 'CSV Export (SI-Daten)', 'Incident wiedereröffnen', and 'Allianz für Cybersicherheit'. The 'Meta-Informationen' section contains two tables:

Art des Vorfalls	Meldezeitpunkt	Aktueller Status	Abgeschlossen
Security Incident Kompromittierter Webserver	2014-06-12 16:36:08	abgeschlossen 2014-06-20 09:25	SI erfolgreich 2014-06-20 09:25

Reaktionszeit	Analysezeit	Lösungszeit	Nachlaufzeit
2 Std. 31 Min. 16 Std. 32 Min.	6 Std. 17 Min. 6 Std. 17 Min.	36 Min. 16 Std. 21 Min.	1 Tag 17 Std. 26 Min. 6 Tage 1 Std. 41 Min.

The 'Klassifikation' section shows a table with columns: 'Auswirkung', 'Priorität', 'Reaktionszeit', 'MI zu erwarten', and 'Hochschulstart'. The values are: gering, niedrig, 10 Std., Nein, Nein.

The 'Security Incident Team' section lists team members:

Funktion	Name	Telefon	Mobil	E-Mail
Vorfallmelder	Incidentmelder, Markus	+49-89-35831-5678	+49-173-9977213	mail@aaron-schweiger.de
Hotliner	Metzger, Stefan	+49-89-35831-8846	+49-123-456789	Stefan.Metzger@lrz.de
Administrator	Nickl, Ewald	+49-89-35831-8748		Ewald.Nickl@lrz.de
SI Coordinator	Reiser, Helmut	+49-89-35831-7854		Helmut.Reiser@lrz.de

Abbildung 7.16.: Screenshot: Incidentstatusseite

Die Incidentstatusseite (Screenshot 7.16) ist in allen Prozessphasen aufrufbar und fasst die zu diesem Vorfall gespeicherten Informationen zusammen. Sie enthält auch die Laufzeiten der einzelnen Prozessphasen, wobei fett gedruckte Werte die Dauer während der Servicezeiten angeben. Die darunter stehenden Werte beziffern die Gesamtdauer. Standardmäßig wird die überblicksartige Version der Statusseite angezeigt. Jedoch lassen sich mit einem Klick alle Details einblenden.

7. Prototypische Anwendung der Software

Auch das Kommunikationsmodul steht in allen Prozessphasen zur Verfügung. Es ermöglicht den E-Mail-Versand von Incidentdaten an einzelne Teammitglieder, Gruppen sowie manuell einzutragende Empfänger. Darüber hinaus ist die Erstellung von Gesprächsnotizen möglich. Das Kommunikationsprotokoll (Screenshot 7.17) listet sämtliche Kommunikationsvorgänge auf. Für jeden Eintrag ist eine Detailseite aufrufbar.

The screenshot displays the 'Kommunikation' (Communication) tab for incident 'SI-47: Gehackte WordPress Installation'. A green notification bar at the top states: 'OK! Die Meldung an die Allianz für Cybersicherheit wurde erfolgreich versendet.' Below this, a table lists communication events:

Zeitpunkt	Art	Empfänger	Betreff	Teammitglied
2014-06-12 22:12	→	LRZ-CSIRT (CSIRT)	[SI-47] Neuer Security Incident eingegangen	SYSTEM
2014-06-12 22:12	→	Incidentmelder, Markus (MELDER)	[SI-47] Neuer Security Incident eingegangen	SYSTEM
2014-06-12 23:16	→	Incidentmelder, Markus (MELDER)	[SI-47] wurde geschlossen	Schweiger, Aaron
2014-06-12 23:22	→	LRZ-CSIRT (CSIRT)	[SI-47] Neue Infos zum WordPress-Incident	Schweiger, Aaron
2014-06-12 23:26	→	Allianz für Cybersicherheit (ACS)	[SI-47] Meldung an die Allianz für Cybersicherheit	Schweiger, Aaron

On the right side, a 'Security Incident' summary box shows: 'Typ: Kompromittierter Webserver', 'Kunde: LMU', 'System: Linux', 'Host: wordpress.lmu.de', 'IP: 123.124.125.126'. Below it, an 'Aktionen' (Actions) menu includes options like 'E-Mail senden', 'Gesprächsnotiz erstellen', 'PDF Export', 'CSV Export (SI-Daten)', and 'CSV Export (Team-Daten)'.

Abbildung 7.17.: Screenshot: Kommunikationsprotokoll

Anhand des incidentbezogenen Verlaufsprotokolls (Screenshot 7.18) können alle Einzelschritte der Vorfallbearbeitung nachvollzogen werden. Das Protokoll ist tabellarisch aufgebaut und kann sowohl sortiert als auch durchsucht werden. Darüber hinaus ist ein Datenexport im CSV-Format möglich.

The screenshot displays the 'Verlauf / History' (History) tab for incident 'SI-47: Gehackte WordPress Installation'. It shows a table of incident state changes:

Zeitpunkt	Feld	alter Wert	neuer Wert	Benutzer
2014-06-12 23:16:06	closedState	NONE	SI_SUCCESS	Schweiger, Aaron
2014-06-12 23:16:06	currentState	MON	CLOSED	Schweiger, Aaron
2014-06-12 23:14:17	currentState	SOL	MON	Schweiger, Aaron
2014-06-12 23:14:10	regelbetrieb		- Datenbackup und/oder DB-Backup wurden ggf. eingespielt. - Update der eingesetzten Software wurde durchgeführt	Schweiger, Aaron
2014-06-12 23:12:06	currentState	ANAL	SOL	Schweiger, Aaron
2014-06-12 23:11:24	analyseerg		Hier werden die Analyseergebnisse eingetragen	Schweiger, Aaron
2014-06-12 23:08:49	currentState	CLASS	ANAL	Schweiger, Aaron
2014-06-12 23:07:26	TEAM		Ewald Nickl wurde als ADMIN hinzugefügt.	Schweiger, Aaron

The interface includes a search bar and a 'CSV Export' button for the history data.

Abbildung 7.18.: Screenshot: Verlaufsprotokoll

Nach dem Abschluss des Sicherheitsvorfalls kann der Hotliner die Review-Seite (Screenshot 7.19) aufrufen, um dort die Ergebnisse des Post Incident Reviews incidentbezogen zu erfassen. Zugleich wird die Durchführung des Reviews im Verlaufsprotokoll automatisch dokumentiert. Die Review-Ergebnisse sind Bestandteil der Incidentstatusseite und des PDF-Reports und können daher auch exportiert werden.

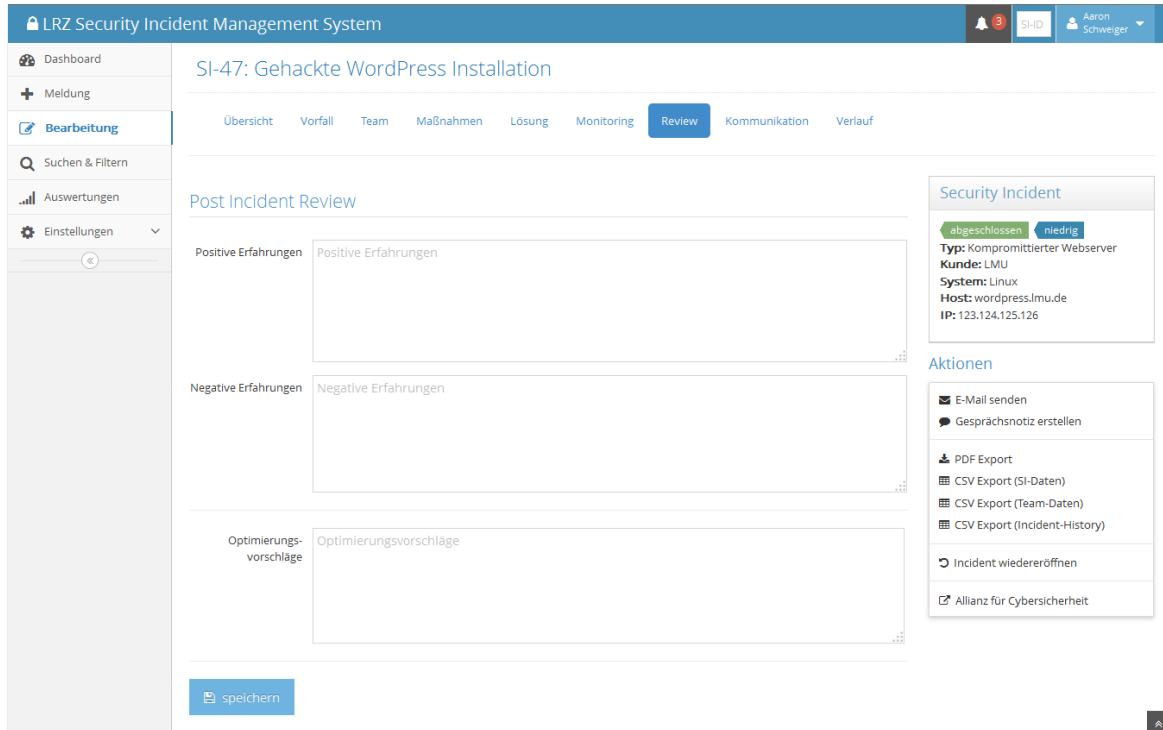


Abbildung 7.19.: Screenshot: Post Incident Review

8. Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde eine webbasierte Information Security Incident Managementsoftware für das Leibniz-Rechenzentrum (LRZ) sowohl konzipiert als auch implementiert. Ausgehend von nationalen und internationalen ITSM-Standards wurde der Security Incident Response Prozess des LRZ betrachtet. Die daraus abgeleiteten Möglichkeiten der Prozessoptimierung sowie die in der Aufgabenstellung verlangten Aspekte dienten der Anforderungsanalyse als Grundlage. Unter Berücksichtigung der insgesamt 40 Anforderungen wurde anschließend ein Konzept für eine individuelle Softwarelösung erarbeitet. Während der Implementierungsphase ist eine benutzerfreundliche Webanwendung entstanden, die den SIR-Prozess des LRZ digital abbildet und zu einer effizienteren Bearbeitung von Sicherheitsvorfällen beiträgt.

Während der erste Abschnitt dieses Kapitel die Ergebnisse der vorliegenden Arbeit im Bezug auf den implementierten Funktionsumfang zusammenfasst, folgt in Abschnitt 8.2 ein Ausblick auf mögliche Erweiterungen der Webanwendung.

8.1. Zusammenfassung

Entsprechend der Zielsetzung dieser Arbeit (vgl. Abschnitt 1.2) wurde ein webbasiertes, den bestehenden SIR-Prozess des LRZ abbildendes Softwaretool zur Optimierung des Information Security Incident Managements entwickelt. Dabei ermöglicht die implementierte Webanwendung im Wesentlichen die standardisierte Erfassung von Sicherheitsvorfällen, deren prozesskonforme Bearbeitung sowie die flexible Nutzung des Datenbestands.

Über die zehn in der Aufgabenstellung (vgl. Abschnitt 1.3) explizit verlangten Funktionen hinaus wurden 23 weitere Anforderungen implementiert. Dazu zählt beispielsweise der Umgang mit Incident Duplikaten ebenso wie die Bearbeitung von Information Requests. Darüber hinaus wurde eine automatisierte Vorfallopriorisierung sowie eine feingranulare Dokumentation sämtlicher Aktionen inklusive der Kommunikationsvorgänge implementiert. Eine LDAP-Integration zum Zweck der Benutzerauthentifizierung und Personendatenabfrage rundet den Funktionsumfang ab. Je nach Blickwinkel richtet sich der Fokus auf verschiedene Aspekte der Individualsoftware:

- Aus *ITSM-Sicht* zeichnet sich die Webanwendung durch eine vollständige SIR-Prozessunterstützung inklusive Zeitmessung auf Prozessphasenebene aus.
- Aus *technischer Sicht* handelt es sich um eine sichere Webanwendung auf PHP-MySQL-Basis, die eine unkomplizierte Wartung und Erweiterung ermöglicht.
- Aus der *Perspektive des Anwenders* steht die übersichtlich strukturierte und komfortabel zu bedienende Benutzeroberfläche im Mittelpunkt.

8. Zusammenfassung und Ausblick

Die Konzeption und Datenmodellierung erfolgte – wie in der Aufgabenstellung gefordert – auf produkt- und plattformunabhängiger Basis. Da ein Standard-Webserver des LRZ als Entwicklungsumgebung verwendet wurde, ist die Lauffähigkeit der Webanwendung auf einem solchen Server sichergestellt.

Bei einer detaillierten Betrachtung des implementierten Funktionsumfangs ist festzustellen, dass die entstandene Webanwendung alle in der Aufgabenstellung verlangten Funktionen und 91% der ermittelten Anforderungen (gewichtet) erfüllt. Der Implementierungsgrad berechnet sich unter Berücksichtigung der Anforderungsgewichtung als Division der Summe der implementierten Anforderungen durch die Summe aller Anforderungen. Aus Abbildung 8.1 ist zu entnehmen, dass alle 18 Muss-Anforderungen vollständig umgesetzt wurden. Darüber hinaus wurden 10 von 11 der Soll-Anforderungen implementiert. Auch von den weniger relevanten Kann-Anforderungen wurden 45% erfüllt. Tabelle 8.1 zeigt den Implementierungsstatus der einzelnen Anforderungen, wobei die in der Aufgabenstellung explizit verlangten Funktionen grau hinterlegt sind.

Mit der im Rahmen dieser Arbeit entwickelten Webanwendung steht dem LRZ ein Werkzeug zur Verfügung, welches das CSIRT bei der effizienten Umsetzung des Security Incident Response Prozesses unterstützt. Im Gegensatz zur momentanen Microsoft Word und Excel basierten Incidentdokumentation ermöglicht die Webanwendung beispielsweise eine effiziente Filterung und Suche innerhalb der abgeschlossenen Security Incidents nach vergleichbaren Vorfällen. Darüber hinaus ist die technische Messung der Prozessperformance anzuführen, die eine Auswertung von KPIs zur Identifikation von Security Trends bereitstellt. Zudem bewirkt die Webanwendung eine Optimierung der Prozessperformance durch die Teilautomatisierung von organisatorischen Tätigkeiten. Als weitere Vorteile sind die Verbesserung der Qualität durch eine Steigerung der Prozesskonformität und die lückenlose Dokumentation zur Erhöhung der Nachvollziehbarkeit zu erwähnen. Letztendlich kann die Webanwendung zu einem noch professionelleren und zugleich effizienteren Umgang mit Sicherheitsvorfällen und somit zur Verbesserung des Gesamtsicherheitsniveaus am LRZ beitragen.

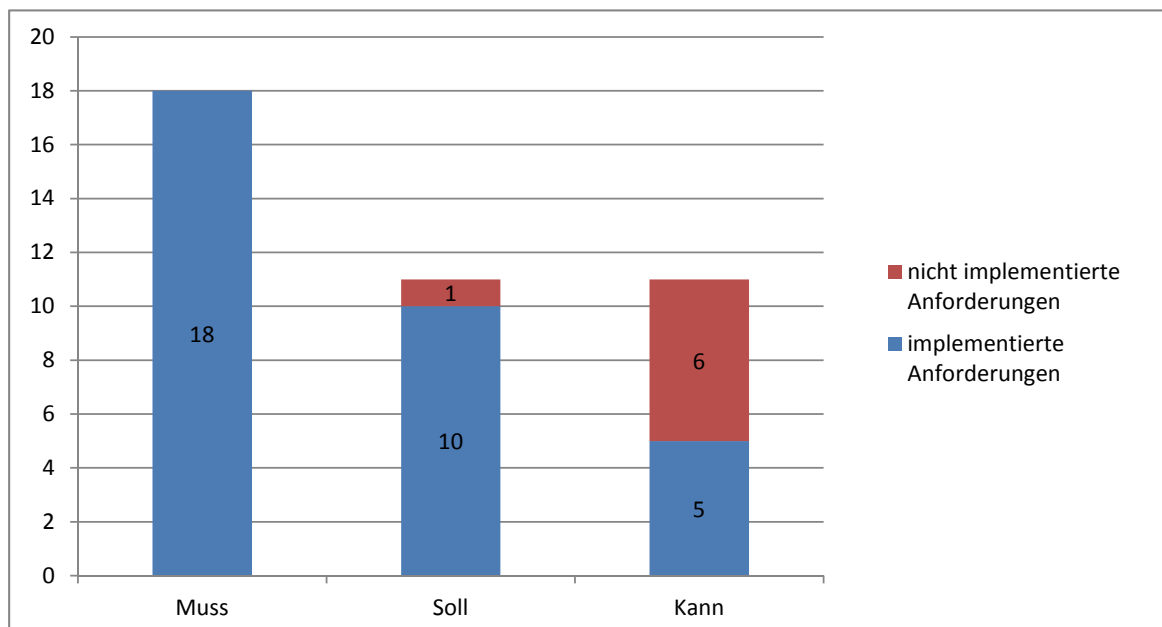


Abbildung 8.1.: Implementierter Funktionsumfang

ID	Anforderung	Impl.	Gewicht
R01	Incidenterfassung mittels Webformular	✓	3
R02	Automatisierte Incidenterfassung durch SIEM-Lösungen	∅	1
R03	Initiale Incident Klassifikation	✓	3
R04	Umgang mit Incident Duplikaten	✓	2
R05	Bearbeitung von Information Requests	✓	1
R06	Initiale Bearbeitung von Security Incidents	✓	3
R07	Unterstützung von SSIs	✓	3
R08	Priorisierung mittels Auswirkungsmatrix	✓	2
R09	Information Request an den Vorfallemitter	✓	1
R10	Weitere Incidentbearbeitung	✓	3
R11	Erweitertes Monitoring	∅	1
R12	Wiedereröffnung von abgeschlossenen Incidents	✓	2
R13	Erfassung der PIR-Ergebnisse	✓	3
R14	Archivierung abgeschlossener Security Incidents	✓	3
R15	Prozess-Unterstützung für Anwender	✓	2
R16	Dashboard	✓	1
R17	Incidentstatusseite	✓	3
R18	Dateiverwaltung	∅	2
R19	Automatisierte Handlungsempfehlungen	∅	1
R20	Dokumentation aller Aktionen	✓	2
R21	E-Mail-Benachrichtigungen	✓	3
R22	Dokumentation der Kommunikation	✓	2
R23	Meldungen an die Allianz für Cyber-Sicherheit	✓	3
R24	Such- und Filterfunktion	✓	3
R25	Tagging von Security Incidents	∅	1
R26	Einzelexport eines Security Incidents	✓	3
R27	Massenexport von Security Incidents	✓	1
R28	Einfache statistische Auswertungen	✓	3
R29	Export von Auswertungen	✓	2
R30	Automatisierter E-Mail-Versand von Auswertungen	∅	1
R31	LRZ-Branding der PDF-Dokumente	✓	1
R32	Benutzerauthentifizierung	✓	3
R33	Rechtevergabe anhand statischer Benutzergruppen	✓	2
R34	Rechtevergabe anhand dynamischer Benutzergruppen	∅	1
R35	Webbasierte Technik	✓	3
R36	Zentrale Datenhaltung in einer Datenbank	✓	3
R37	Lauffähigkeit auf einem Standard-Webserver	✓	3
R38	Gebrauchstauglichkeit und Benutzerfreundlichkeit	✓	2
R39	IT-Sicherheit	✓	3
R40	Wartbarkeit und Erweiterbarkeit	✓	2

Tabelle 8.1.: Übersicht der implementierten und nicht implementierten Anforderungen (explizit verlangte Funktionen grau hinterlegt)

8.2. Ausblick

Im Rahmen einer zukünftigen Arbeit könnte die Webanwendung um die im Folgenden beschriebenen Funktionen erweitert werden.

Als Ergänzung der klassischen Meldewege wäre eine automatisierte Incidenterfassung durch eine SIEM-Lösung wünschenswert. Hierfür wäre eine Schnittstelle zur Anbindung von Drittsystemen zu implementieren. Dadurch wäre es möglich, auch das für reguläre Störungen verwendete Ticketsystem bidirektional zu integrieren, sodass beispielsweise in der iET ITSM-Suite erfasste Störungen mit wenigen Klicks in die Security Incident Managementsoftware übertragen werden können. In der anderen Richtung könnten Information Requests in iET-Tickets konvertiert werden.

Darüber hinaus könnte die Webanwendung mit einem Modul zur Dateiverwaltung ausgestattet werden. Dadurch könnten im Rahmen der Beweissicherung gesammelte Dateien wie beispielsweise Logfiles incidentbezogen zentral gespeichert und aus der Webanwendung heraus geöffnet werden. In diesem Zusammenhang könnte das Meldeformular um eine Uploadmöglichkeit ergänzt werden, sodass der Vorfalleinmelder beispielsweise Auszüge aus Logfiles oder andere Dateien direkt an das CSIRT übermitteln kann. Verschiedene Ansätze für eine technische Umsetzung werden in Abschnitt 5.2 diskutiert.

Da während der Incidentbearbeitung das Hinzuziehen externer Personen wie zum Beispiel IT-Forensiker erforderlich sein kann, erscheint es sinnvoll, den Bereich der Teamzusammensetzung diesbezüglich zu ergänzen, sodass externe Personen, deren Kontaktdaten nicht via LDAP abrufbar sind, manuell hinzugefügt werden können.

Das Kommunikationsmodul könnte dahingehend erweitert werden, dass sich die Anwender innerhalb des Systems gegenseitig Nachrichten zuschicken und Dateianhänge beifügen können. Der E-Mail-Versand wäre dann nur noch für externe Parteien relevant, die über keinen Systemzugriff verfügen.

Das System könnte anhand von Parallelfällen dem Anwender die Lessons Learned aufzeigen und konkrete Handlungsempfehlungen für die Festlegung der weiteren Vorgehensweise bieten. Durch die Empfehlungsautomatik könnte vorhandenes Wissen optimal in den Prozessablauf eingebunden werden. Weitere Überlegungen zu einer derartigen Wissensmanagementfunktionalität werden in Abschnitt 5.3.3 diskutiert.

Um eine noch höhere Qualität bei der Incident Nachsorge zu erzielen, könnten die vorhandenen Monitoringfunktionen erweitert werden, wobei das System die verantwortlichen Personen per E-Mail pünktlich an die Durchführung der jeweiligen Monitoringaktivitäten erinnert. Die Erledigung könnte durch Klick auf einen Link in der E-Mail bestätigt werden. Außerdem wäre ein Rückkanal für die Mitteilung etwaiger Auffälligkeiten einzurichten.

Weiterhin könnte eine Tagging-Funktion ergänzt werden, die das Anreichern von Security Incidents mit flexiblen Metainformationen ermöglicht. Durch die Verwendung gleicher Tags können ähnliche Vorfälle flexibel gruppiert werden. Diese zusätzlichen Metadaten könnten zum Zweck einer verfeinerten Filterung und Auswertung genutzt werden.

Da anhand der relativen Zu- oder Abnahme von Kennzahlen im Zeitverlauf Trends leichter zu identifizieren sind, könnten bei einer Erweiterung des Auswertungsmoduls die Werte des Auswertungszeitraums in Relation zu einem Referenzzeitraum gesetzt werden, sodass prozentuale Veränderungen sichtbar werden. Zudem wäre ein automatisierter Reportversand an das CSIRT denkbar.

Ebenso wäre es sinnvoll, wenn beteiligte Systemadministratoren und Abteilungsleiter den aktuellen Stand eines Vorfalls einsehen und beispielsweise Analyse- oder Monitoringergebnisse direkt im System erfassen könnten. Hierfür wäre eine Rechtevergabe anhand dynamischer Benutzergruppen notwendig, die bereits für den Zugriff auf die Incidentstatusseite realisiert wurde. Diese Funktionalität könnte zusätzlich im Bereich der Incidentbearbeitung angewendet werden, wobei zunächst ein differenziertes Konzept für die Vergabe von Lese- und Schreibrechten auf Objektebene zu erarbeiten wäre.

Um die Webanwendung im Kontext einer organisationsübergreifenden Vorfallobearbeitung einsetzen zu können, wären vor allem in den Bereichen Benutzerverwaltung, Authentifizierung und Rechtevergabe Anpassungen notwendig.

Da bei der Implementierung auf eine unkomplizierte Erweiterbarkeit der Webanwendung geachtet wurde, sollte eine Weiterentwicklung entsprechend der vorgenannten Aspekte problemlos möglich sein.

Abkürzungsverzeichnis

ACS	Allianz für Cyber-Sicherheit
BSB	Bayerische Staatsbibliothek
BSI	Bundesamt für Sicherheit in der Informationstechnik
BYOD	Bring Your Own Device
CSIRT	Computer Security Incident Response Team
CSS	Cascading Style Sheets
CSV	Comma-separated values
DEISA	Distributed European Infrastructure for Supercomputing Applications
DFN	Deutsches Forschungsnetz
EGI	European Grid Infrastructure
ENISA	European Union Agency for Network and Information Security
FTP	File Transfer Protocol
GUI	Graphical User Interface
HM	Hochschule für angewandte Wissenschaften München
HTML	Hypertext Markup Language
ISMS	Information Security Management System
ISM	Information Security Management
ITIL	IT Infrastructure Library
ITSM	IT Service Management
IT	Informationstechnik (<i>engl. information technology</i>)
KPI	Key-Performance-Indikator
LDAP	Lightweight Directory Access Protocol
LGPL	Lesser General Public License
LMU	Ludwig-Maximilians-Universität München
LRZ	Leibniz-Rechenzentrum
MD5	Message-Digest Algorithm 5
MIC	Major Incident Coordinator
MIT	Massachusetts Institute of Technology

8. Zusammenfassung und Ausblick

MWN	Münchener Wissenschaftsnetz
NIST	National Institute of Standards and Technology
PDF	Portable Document Format
PDO	PHP Data Objects
PHP	PHP: Hypertext Preprocessor
PIR	Post Incident Review
RSS	Really Simple Syndication
SIC	Security Incident Coordinator
SIEM	Security Information and Event Management
SIM	Security Incident Management
SIR	Security Incident Response
SLA	Service Level Agreement
SMB	Server Message Block
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
SSH	Secure Shell
SSI	Standard Security Incident
SSL	Secure Sockets Layer (<i>alte Bezeichnung für TLS</i>)
TLS	Transport Layer Security
TUM	Technische Universität München
VPN	Virtual Private Network
XML	Extensible Markup Language
XSS	Cross-Site-Scripting

Abbildungsverzeichnis

2.1. Einordnung des Security Incident Managements in den ITSM Kontext	5
3.1. Ablauf des Security Incident Response Prozesses am LRZ, vgl. [Met13]	14
5.1. Prozessphasen eines Sicherheitsvorfalls	30
5.2. Darstellung des SIR-Prozesses als Flussdiagramm	31
5.3. Meldewege und Incidenterfassung in der Webanwendung	41
5.4. Datenmodell der Webanwendung als relationales Datenbankschema	55
6.1. Ordnerstruktur	64
6.2. Strukturdiagramm und Ablauflogik der Softwarekomponenten	67
6.3. Module und Aktionen	68
6.4. Screenshot: Sperrung von mobilen Endgeräten	81
6.5. Screenshot: Dashboard (/dashboard.php)	82
6.6. Screenshot: Statuswechsel	84
6.7. Visualisierung diffBusinessHours	86
6.8. Screenshot: Incidentstatusseite (/si-overview.php)	91
6.9. Screenshot: Initiale Klassifikation (/si-init.php)	96
6.10. Screenshot: Suchen & Filtern (/filter.php)	107
6.11. Screenshot: Auswertungen (/stats.php)	111
7.1. Screenshot: Meldung eines Security Incidents	115
7.2. Bestätigungsmail an den Vorfallemitter	116
7.3. Benachrichtigungsmail an das CSIRT	116
7.4. Screenshot: Dashboard aus CSIRT-Sicht	117
7.5. Screenshot: Initiale Klassifikation	118
7.6. Screenshot: Weitere Klassifikation und Priorisierung	119
7.7. Screenshot: Auswahl des Hotliners	120
7.8. Screenshot: Auswahl des Security Incident Coordinators	120
7.9. Screenshot: Hinzufügen weiterer Teammitglieder	121
7.10. Screenshot: Verwaltung des Security Incident Teams	121
7.11. Screenshot: Mailfunktion	122
7.12. Screenshot: Analysephase	122
7.13. Screenshot: Lösungsphase	123
7.14. Screenshot: Monitoringphase	124
7.15. Benachrichtigung des Vorfallemitters über den Incidentabschluss	124
7.16. Screenshot: Incidentstatusseite	125
7.17. Screenshot: Kommunikationsprotokoll	126
7.18. Screenshot: Verlaufsprotokoll	126
7.19. Screenshot: Post Incident Review	127

Abbildungsverzeichnis

8.1. Implementierter Funktionsumfang	130
D.1. Beispiel für einen PDF-Report (Seite 1)	156
D.2. Beispiel für einen PDF-Report (Seite 2)	157
D.3. Beispiel für einen PDF-Report (Seite 3)	158

Tabellenverzeichnis

4.1. Übersicht der Anforderungen	28
5.1. Datenbanktabelle <code>incidents</code>	57
5.2. Datenbanktabelle <code>contacts</code>	58
5.3. Datenbanktabelle <code>communication</code>	58
5.4. Datenbanktabelle <code>history</code>	59
5.5. Datenbanktabelle <code>ssiTpl</code>	59
5.6. Datenbanktabelle <code>users</code>	60
5.7. Datenbanktabelle <code>types</code>	60
5.8. Datenbanktabelle <code>settings</code>	61
5.9. Datenbanktabelle <code>syslog</code>	61
8.1. Übersicht der implementierten und nicht implementierten Anforderungen (explizit verlangte Funktionen grau hinterlegt)	131
A.1. SSI-Template: „Umgang mit kompromittierten Webservern“	148

Quellcodeverzeichnis

6.1. Codestruktur am Beispiel <code>si-monitoring.php</code>	65
6.2. Cron-Skript auf der Login-Seite (<code>/index.php</code>)	73
6.3. Nutzerauthentifizierung via LDAP (<code>/inc/login.php</code>)	75
6.4. Dynamische Rechtevergabe (<code>/inc/si-overview.php</code>)	76
6.5. LDAP-Synchronisation (<code>/inc/cron.php</code>)	78
6.6. Erzwingung einer HTTPS-Verbindung (<code>/inc/basics.php</code>)	79
6.7. <code>.htaccess</code> -Verzeichnisschutz	79
6.8. Serverseitige Eingabefilterung	80
6.9. Implementierung des Session-Tokens	80
6.10. Unterbindung der Nutzung auf mobilen Endgeräten (<code>/index.php</code>)	81
6.11. Funktion <code>setAlert</code> (<code>/inc/functions.php</code>)	83
6.12. Ausgabe von Systemmeldungen (<code>/inc/tpl.alert.php</code>)	83
6.13. Aktion <code>updateState</code> (<code>/actions/incidents.php</code>)	84
6.14. Funktion <code>updateState</code> (<code>/inc/functions.si.php</code>)	85
6.15. Funktion <code>diffBusinessHours</code> (<code>/inc/functions.php</code>)	88
6.16. Funktion <code>checkTeamComplete</code> (<code>/inc/functions.si.php</code>)	89
6.17. Funktion <code>insertHistory</code> (<code>/inc/functions.si.php</code>)	90
6.18. Funktion <code>mailtoSomebody</code> (<code>/inc/functions.si.php</code>)	93
6.19. Aktion <code>newIncident</code> (<code>/actions/new.php</code>)	95
6.20. Aktion <code>classifyIncident</code> (<code>/actions/incidents.php</code>)	97
6.21. Funktion <code>applySsiTemplate</code> (<code>/inc/functions.si.php</code>)	97
6.22. Aktion <code>sendAwMail</code> (<code>/actions/request.php</code>)	99
6.23. Automatische Ermittlung von Auswirkung, Priorität und Reaktionszeit in der Aktion <code>updateIncidentValues</code>	100
6.24. Aktion <code>addContact</code> (<code>/actions/contacts.php</code>)	101
6.25. Aktion <code>updateIncidentValues</code> (<code>/actions/incidents.php</code>)	103
6.26. Aktion <code>closeIncident</code> (<code>/actions/incidents.php</code>)	105
6.27. Implementierung der Incidentfilterung	108
6.28. Überwachung der Reaktionszeiten (<code>/stats.php</code>)	109
6.29. Auswertung nach Prioritäten (<code>/stats.php</code>)	110
6.30. Modul <code>export-csv</code> (<code>/actions/export-csv.php</code>)	113
6.31. Aktion <code>si-collection</code> im Modul <code>export-pdf</code> (<code>/actions/export-pdf.php</code>)	114
B.1. Datenbank-Setup: Erstellung der Tabellen	149
C.1. Datensätze der <code>types</code> Tabelle	153

Literaturverzeichnis

- [Ark13a] ARKIN, BRAD: *Illegal Access to Adobe Source Code*. In: *Adobe Secure Software Engineering Team Blog*. Oktober 2013. <http://blogs.adobe.com/asset/2013/10/illegal-access-to-adobe-source-code.html>.
- [Ark13b] ARKIN, BRAD: *Important Customer Security Announcement*. In: *Adobe Featured Blogs*. Oktober 2013. <http://blogs.adobe.com/conversations/2013/10/important-customer-security-announcement.html>.
- [BIT13] BITKOM BUNDESVERBAND INFORMATIONSWIRTSCHAFT, TELEKOMMUNIKATION UND NEUE MEDIEN E.V.: *Leitfaden - Bring Your Own Device*, April 2013. http://www.bitkom.org/files/documents/20130404_LF_BYOD_2013_v2.pdf.
- [Bun09] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *IT-Grundschatz-Baustein B 1.8 Behandlung von Sicherheitsvorfällen*, 2009. https://www.bsi.bund.de/DE/Themen/ITGrundschatz/ITGrundschatzKataloge/Inhalt/_content/baust/b01/b01008.html.
- [Bun11] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *Leitfaden IT-Forensik*, März 2011. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Internetsicherheit/Leitfaden_IT-Forensik_pdf.pdf.
- [Bun12a] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *Goldene Regeln für den IT-Grundschatz-Baustein B 5.21 Webanwendungen*, November 2012. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschatz/Download/Vorabversionen/Baustein_Webanwendungen_Goldene_Regeln.pdf.
- [Bun12b] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *IT-Grundschatz-Baustein B 5.21 Webanwendungen*, April 2012. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschatz/Download/Vorabversionen/Baustein_Webanwendungen.pdf.
- [Bun13] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *Fokus IT-Sicherheit 2013*, Juli 2013. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Fokus_IT-Sicherheit_2013_nbf.pdf.
- [Bun14] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *Allianz für Cyber-Sicherheit*, Juni 2014. https://www.bsi.bund.de/DE/Themen/Cyber-Sicherheit/Strategie/Allianz_fuer_Cybersicherheit/Allianz_node.html.

- [CMGS12] CICHONSKI, PAUL, TOM MILLAR, TIM GRANCE und KAREN SCARFONE: *Computer Security Incident Handling Guide. Recommendations of the National Institute of Standards and Technology. NIST Special Publication 800-61 Revision 2*, August 2012. <http://csrc.nist.gov/publications/nistpubs/800-61rev2/SP800-61rev2.pdf>.
- [Fra13] FRANKFURTER ALLGEMEINE ZEITUNG: *Hacker stehlen Daten von Millionen Adobe-Kunden*, Oktober 2013. <http://www.faz.net/-gqm-7i4k9>.
- [Fre13] FREUND, ROSA: *Standard-Security-Incident 0002. Prozessbeschreibung. Umgang mit kompromittierten Webservern*. Internes Dokument des Leibniz-Rechenzentrums, Mai 2013.
- [Ges11] GESCHONNECK, ALEXANDER: *Computer-Forensik: Computerstraftaten erkennen, ermitteln, aufklären*. iX-Edition. Dpunkt.Verlag GmbH, September 2011.
- [Hei13] HEISE SECURITY: *Einbruch bei Adobe: Millionen Kundendaten sowie Sourcecode von ColdFusion und Acrobat geklaut*, Oktober 2013. <http://heise.de/-1972175>.
- [ISO13a] ISO/IEC 27000:2014: *Information technology – Security techniques – Information security management systems – Overview and vocabulary*, 2013. ISO/IEC, Geneva, Switzerland.
- [ISO13b] ISO/IEC 27001:2013: *Information technology – Security techniques – Information security management systems – Requirements*, 2013. ISO/IEC, Geneva, Switzerland.
- [ISO13c] ISO/IEC 27002:2013: *Information technology – Security techniques – Code of practice for information security controls*, 2013. ISO/IEC, Geneva, Switzerland.
- [ISO13d] ISO/IEC 27035 (WORKING DRAFT): *Information technology – Security techniques – Information security incident management – Part 1-3*, 2013. ISO/IEC, Geneva, Switzerland.
- [Kle13] KLEINER, FRITZ: *IT Service Management: Aus der Praxis für die Praxis*. Springer Vieweg, Wiesbaden, 2013.
- [Klu13] KLUGE, HANS-GEORG: *Adobe-Datenklau: Passwörter von Adobe-Kunden in Gefahr*, November 2013. <http://www.teltarif.de/adobe-datenklau-passwoerter-gefahr-hack/news/53207.html>.
- [Met13] METZGER, STEFAN: *Information Security Incident Management. Prozessbeschreibung*. Internes Dokument des Leibniz-Rechenzentrums, Dezember 2013.
- [MHR11] METZGER, STEFAN, WOLFGANG HOMMEL und HELMUT REISER: *Integriertes Management von Sicherheitsvorfällen*. In: *Sicherheit in vernetzten Systemen, 18. DFN Workshop*. DFN-CERT Services, Hamburg, Februar 2011.
- [Pip12] PIPEK, VOLKMAR: *Ubiquitous Computing*. In: *Enzyklopädie der Wirtschaftsinformatik. Online-Lexikon*. Oldenbourg Wissenschaftsverlag, Oktober 2012. <http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/technologien-methoden/Rechnernetz/Ubiquitous-Computing>.

- [Stü13] STÜCKLER, MORITZ: *Warum der Adobe-Hack noch viel schlimmer ist als bisher angenommen*, November 2013. <http://t3n.de/news/adobe-hack-noch-viel-schlimmer-506598/>.
- [Wei91] WEISER, MARK: *The Computer for the 21st Century*. Scientific American, 265(3):66–75, January 1991.

Anhang

A. SSI-Template: „Umgang mit kompromittierten Webservern“

MI zu erwarten	Nein
Hochschulstart betroffen	Nein
Standort Zielsystem	MWN, Grid-Systeme
Standort Quellsystem	extern
Betroffene Dienste	keine kritischen Dienste
Betroffene Informationen	keine vertraulichen Infos
Anzahl der betroffenen Systeme	1
Auswirkung	gering
Priorität	niedrig
Reaktionszeit	10 Std.
Erstmaßnahmen	<ul style="list-style-type: none"> - Seitenbetreiber benachrichtigen, falls der Hinweis nicht von diesem kam - Webserver deaktivieren, falls dies notwendig ist, um weiteren Schaden abzuwenden - htdocs-Ordner und DB-Dump archivieren - Kennungs- und DB-Passwort zurücksetzen
Maßnahmen	<ul style="list-style-type: none"> - Eingrenzung des Hack-Zeitpunkts - Analyse SSH-/FTP-Logs hinsichtlich auffälliger Logins, deren Zeitpunkt mit dem Timestamp veränderter Dateien übereinstimmt - Abgleich auf Ähnlichkeit mit bekannten Kompromittierungen (vgl. LRZ-Wiki)
Analyseergebnisse	<i>individuell</i>
Regelbetrieb	<i>individuell</i>
Typ der Lösung	Backup eingespielt
Lösung	<ul style="list-style-type: none"> - Datenbackup und/oder DB-Backup wurden ggf. eingespielt. - Hochgeladene Dateien wurden ggf. entfernt. - Gesamter Datenbereich oder bestimmte Ordner des Webservers wurden auf read-only gesetzt. - Update der eingesetzten Software wurde durchgeführt. - Kunde wurde darauf hingewiesen, die eingesetzte Software aktuell zu halten.
Monitoring-Dauer	7 Tage
Monitoring-Intervall	10 Stunden
Monitoring Aufgaben und Verantwortlichkeiten	Der Webmaster prüft den kompromittierten Webserver über einen Zeitraum von 7 Tagen nach Lösung des Vorfalls einmal täglich auf weitere Auffälligkeiten.
Auffälligkeiten	keine

Tabelle A.1.: SSI-Template: „Umgang mit kompromittierten Webservern“

B. Datenbank-Setup

Listing B.1: Datenbank-Setup: Erstellung der Tabellen

```

1  -- Tabellenstruktur für Tabelle 'sims_communication'
2  CREATE TABLE IF NOT EXISTS 'sims_communication' (
3    'id' int(11) NOT NULL AUTO.INCREMENT,
4    'si' mediumint(9) DEFAULT NULL,
5    'user' int(11) DEFAULT NULL,
6    'userName' varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
7    'stamp' timestamp NULL DEFAULT CURRENT_TIMESTAMP,
8    'comStamp' timestamp NULL DEFAULT NULL,
9    'medium' enum('TEL','MAIL','PERS') COLLATE utf8_unicode_ci DEFAULT NULL,
10   'richtung' enum('IN','OUT') COLLATE utf8_unicode_ci DEFAULT NULL,
11   'subject' varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
12   'content' text COLLATE utf8_unicode_ci,
13   'name' varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
14   'tel' varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
15   'email' varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
16   'contactRole' varchar(12) COLLATE utf8_unicode_ci DEFAULT NULL,
17   PRIMARY KEY ('id'),
18   KEY 'si' ('si')
19 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
20
21 -- Tabellenstruktur für Tabelle 'sims_contacts'
22 CREATE TABLE IF NOT EXISTS 'sims_contacts' (
23   'id' int(11) NOT NULL AUTO.INCREMENT,
24   'si' mediumint(9) NOT NULL,
25   'contactRole' varchar(12) COLLATE utf8_unicode_ci DEFAULT NULL,
26   'kennung' varchar(40) COLLATE utf8_unicode_ci DEFAULT NULL,
27   'name' varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
28   'vorname' varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
29   'anrede' enum('Frau','Herr') COLLATE utf8_unicode_ci DEFAULT NULL,
30   'email' varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
31   'tel' varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
32   'mobil' varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
33   'comment' varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
34   PRIMARY KEY ('id'),
35   KEY 'si' ('si')
36 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
37
38 -- Tabellenstruktur für Tabelle 'sims_history'
39 CREATE TABLE IF NOT EXISTS 'sims_history' (
40   'id' int(11) NOT NULL AUTO.INCREMENT,
41   'si' mediumint(9) DEFAULT NULL,
42   'user' int(11) DEFAULT NULL,
43   'stamp' timestamp NULL DEFAULT CURRENT_TIMESTAMP,
44   'field' varchar(40) COLLATE utf8_unicode_ci DEFAULT NULL,
45   'old' text COLLATE utf8_unicode_ci,
46   'new' text COLLATE utf8_unicode_ci,
47   'userName' varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
48   PRIMARY KEY ('id'),
49   KEY 'id' ('id'),
50   KEY 'si' ('si')
51 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
52
53

```

```

54 — Tabellenstruktur für Tabelle 'sims_incidents'
55 CREATE TABLE IF NOT EXISTS 'sims_incidents' (
56   'id' mediumint(9) NOT NULL AUTO.INCREMENT,
57   'duplicateId' mediumint(9) DEFAULT NULL,
58   'user' int(11) DEFAULT NULL,
59   'stamp' timestamp NULL DEFAULT CURRENT.TIMESTAMP,
60   'createdStamp' timestamp NULL DEFAULT NULL,
61   'closedStamp' timestamp NULL DEFAULT NULL,
62   'closedState' enum('NONE', 'IR_LAW', 'IR_FWD', 'SLD_DUPLICATE', 'SLC_CANCEL', '
        SLSUCCESS') COLLATE utf8_unicode_ci NOT NULL DEFAULT 'NONE',
63   'currentStamp' timestamp NULL DEFAULT NULL,
64   'currentState' enum('NEW', 'CLASS', 'ANAL', 'SOL', 'MON', 'CLOSED', 'IR_LAW', '
        IR_FWD') COLLATE utf8_unicode_ci DEFAULT NULL,
65   'reopened' tinyint(1) DEFAULT NULL,
66   'saldo1' int(11) NOT NULL DEFAULT '0',
67   'saldo2' int(11) NOT NULL DEFAULT '0',
68   'saldo3' int(11) NOT NULL DEFAULT '0',
69   'saldo4' int(11) NOT NULL DEFAULT '0',
70   'saldo1bh' int(11) NOT NULL DEFAULT '0',
71   'saldo2bh' int(11) NOT NULL DEFAULT '0',
72   'saldo3bh' int(11) NOT NULL DEFAULT '0',
73   'saldo4bh' int(11) NOT NULL DEFAULT '0',
74   'title' varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
75   'description' text COLLATE utf8_unicode_ci,
76   'vorfallzeitpunkt' varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
77   'kunde' varchar(12) COLLATE utf8_unicode_ci DEFAULT NULL,
78   'host' varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
79   'ip' varchar(39) COLLATE utf8_unicode_ci DEFAULT NULL,
80   'os' varchar(12) COLLATE utf8_unicode_ci DEFAULT NULL,
81   'klassifikation' enum('INCIDENT', 'DUPLICATE', 'REQUEST') COLLATE
        utf8_unicode_ci DEFAULT NULL,
82   'incidentType' varchar(12) COLLATE utf8_unicode_ci DEFAULT NULL,
83   'majorExpected' tinyint(1) DEFAULT NULL,
84   'hochschulstart' tinyint(1) DEFAULT NULL,
85   'matrixZielsys' tinyint(1) DEFAULT NULL,
86   'matrixDienste' tinyint(1) DEFAULT NULL,
87   'matrixInfos' tinyint(1) DEFAULT NULL,
88   'matrixAnzahl' tinyint(1) DEFAULT NULL,
89   'matrixQuellsys' tinyint(1) DEFAULT NULL,
90   'auswirkung' tinyint(1) DEFAULT NULL,
91   'prio' tinyint(1) DEFAULT NULL,
92   'reaktionszeit' mediumint(9) DEFAULT NULL,
93   'firstaid' text COLLATE utf8_unicode_ci,
94   'vorgehensweise' text COLLATE utf8_unicode_ci,
95   'analyseerg' text COLLATE utf8_unicode_ci,
96   'lsgDetails' text COLLATE utf8_unicode_ci,
97   'lsgType' varchar(12) COLLATE utf8_unicode_ci DEFAULT NULL,
98   'regelbetrieb' text COLLATE utf8_unicode_ci,
99   'monDauer' tinyint(4) DEFAULT NULL,
100  'monIntervall' tinyint(4) DEFAULT NULL,
101  'monAufgabe' text COLLATE utf8_unicode_ci,
102  'monAuffaellig' text COLLATE utf8_unicode_ci,
103  'revPositiv' text COLLATE utf8_unicode_ci,
104  'revNegativ' text COLLATE utf8_unicode_ci,
105  'revOptimierung' text COLLATE utf8_unicode_ci,
106  PRIMARY KEY ('id')
107 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

```

```

108
109 — Tabellenstruktur für Tabelle 'sims_settings'
110 CREATE TABLE IF NOT EXISTS 'sims_settings' (
111   'id' varchar(30) COLLATE utf8_unicode_ci NOT NULL,
112   'value' varchar(255) COLLATE utf8_unicode_ci NOT NULL,
113   'comment' varchar(255) COLLATE utf8_unicode_ci NOT NULL,
114   PRIMARY KEY ('id')
115 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
116
117 — Tabellenstruktur für Tabelle 'sims_ssiTpl'
118 CREATE TABLE IF NOT EXISTS 'sims_ssiTpl' (
119   'id' varchar(12) COLLATE utf8_unicode_ci NOT NULL,
120   'majorExpected' tinyint(1) DEFAULT NULL,
121   'hochschulstart' tinyint(1) DEFAULT NULL,
122   'matrixZielsys' tinyint(1) DEFAULT NULL,
123   'matrixDienste' tinyint(1) DEFAULT NULL,
124   'matrixInfos' tinyint(1) DEFAULT NULL,
125   'matrixAnzahl' tinyint(1) DEFAULT NULL,
126   'matrixQuellsys' tinyint(1) DEFAULT NULL,
127   'auswirkung' tinyint(1) DEFAULT NULL,
128   'prio' tinyint(1) DEFAULT NULL,
129   'reaktionszeit' mediumint(9) DEFAULT NULL,
130   'firstaid' text COLLATE utf8_unicode_ci,
131   'vorgehensweise' text COLLATE utf8_unicode_ci,
132   'analyseerg' text COLLATE utf8_unicode_ci,
133   'lsgDetails' text COLLATE utf8_unicode_ci,
134   'lsgType' varchar(12) COLLATE utf8_unicode_ci DEFAULT NULL,
135   'regelbetrieb' text COLLATE utf8_unicode_ci,
136   'monDauer' tinyint(4) DEFAULT NULL,
137   'monIntervall' tinyint(4) DEFAULT NULL,
138   'monAufgabe' text COLLATE utf8_unicode_ci,
139   'monAuffaellig' text COLLATE utf8_unicode_ci,
140   'revPositiv' text COLLATE utf8_unicode_ci,
141   'revNegativ' text COLLATE utf8_unicode_ci,
142   'revOptimierung' text COLLATE utf8_unicode_ci,
143   PRIMARY KEY ('id'),
144   UNIQUE KEY 'id' ('id')
145 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
146
147 — Tabellenstruktur für Tabelle 'sims_syslog'
148 CREATE TABLE IF NOT EXISTS 'sims_syslog' (
149   'id' int(11) NOT NULL AUTOINCREMENT,
150   'stamp' timestamp NULL DEFAULT CURRENT_TIMESTAMP,
151   'user' int(11) DEFAULT '0',
152   'si' mediumint(9) DEFAULT '0',
153   'action' varchar(40) COLLATE utf8_unicode_ci DEFAULT NULL,
154   'status' enum('OK','ERROR','INFO') COLLATE utf8_unicode_ci DEFAULT NULL,
155   'details' varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
156   PRIMARY KEY ('id')
157 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
158
159 — Tabellenstruktur für Tabelle 'sims_types'
160 CREATE TABLE IF NOT EXISTS 'sims_types' (
161   'id' mediumint(9) NOT NULL AUTOINCREMENT,
162   'category' varchar(40) COLLATE utf8_unicode_ci DEFAULT NULL,
163   'name' varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
164   'value' varchar(12) COLLATE utf8_unicode_ci NOT NULL,

```

```
165 'sort' tinyint(4) DEFAULT NULL,
166 PRIMARY KEY ('id')
167 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci AUTOINCREMENT
    =93 ;
168
169 — Tabellenstruktur für Tabelle 'sims_users'
170 CREATE TABLE IF NOT EXISTS 'sims_users' (
171 'id' int(11) NOT NULL AUTOINCREMENT,
172 'loginStamp' timestamp NULL DEFAULT NULL,
173 'ip' varchar(40) COLLATE utf8_unicode_ci DEFAULT NULL,
174 'userRole' enum('ROOT', 'CSIRT', 'USER') COLLATE utf8_unicode_ci DEFAULT NULL,
175 'kennung' varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
176 'name' varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
177 'vorname' varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
178 'anrede' enum('Frau', 'Herr') COLLATE utf8_unicode_ci DEFAULT NULL,
179 'email' varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
180 'tel' varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
181 'mobil' varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
182 'comment' varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
183 'lastLdapUpdate' date DEFAULT NULL,
184 PRIMARY KEY ('id'),
185 UNIQUE KEY 'kennung' ('kennung')
```

C. Datensätze der types-Tabelle

Listing C.1: Datensätze der types Tabelle

```

1 INSERT INTO 'sims_types' ('id', 'category', 'name', 'value', 'sort') VALUES
2 (1, 'kunde', 'LRZ', 'LRZ', 1),
3 (2, 'kunde', 'LMU', 'LMU', 2),
4 (3, 'kunde', 'TUM', 'TUM', 3),
5 (4, 'kunde', 'BSB', 'BSB', 5),
6 (5, 'kunde', 'Sonstiger', 'SONST', 127),
7 (6, 'os', 'Windows', 'WIN', 1),
8 (7, 'os', 'Linux', 'LINUX', 2),
9 (8, 'os', 'Mac OS', 'MAC', 3),
10 (9, 'os', 'Mobiles System', 'MOBILE', 4),
11 (10, 'incidentType', 'Externer SSH Scan', 'SSI-1', 1),
12 (11, 'incidentType', 'Kompromittierter Webserver', 'SSI-2', 2),
13 (12, 'incidentType', 'Kompromittierung virt. Server', 'SSI-3', 3),
14 (13, 'incidentType', 'Viren-infiziertes LRZ-System', 'SSI-4', 4),
15 (14, 'incidentType', 'Kein Standard SI', 'SONST', 127),
16 (15, 'lsgType', 'Neuinstallation des Systems', 'NEWINST', 1),
17 (16, 'lsgType', 'Bereinigung des Systems', 'CLEANING', 2),
18 (17, 'lsgType', 'Anpassung der Systemkonfiguration', 'CONFIG', 3),
19 (18, 'lsgType', 'Einspielung eines Backups', 'BACKUP', 4),
20 (20, 'prio', 'niedrig', '1', 1),
21 (21, 'prio', 'mittel', '2', 2),
22 (22, 'prio', 'hoch', '3', 3),
23 (23, 'prio', 'sehr hoch', '4', 4),
24 (24, 'matrixZielsys', 'extern', '1', 1),
25 (25, 'matrixZielsys', 'MWN Grid-Systeme', '2', 2),
26 (26, 'matrixZielsys', 'LRZ-intern', '3', 3),
27 (27, 'matrixQuellsys', 'extern', '1', 1),
28 (28, 'matrixQuellsys', 'MWN Grid-Systeme', '2', 2),
29 (29, 'matrixQuellsys', 'LRZ-intern', '3', 3),
30 (30, 'matrixDienste', 'keine kritischen Dienste', '1', 1),
31 (31, 'matrixDienste', 'Dienste für MWN/Grid-Umgebung', '2', 2),
32 (32, 'matrixDienste', 'Wichtige LRZ-Dienste', '3', 3),
33 (33, 'matrixInfos', 'keine vertraulichen Infos', '1', 1),
34 (34, 'matrixInfos', 'vertrauliche Infos bzgl. MWN/Grid', '2', 2),
35 (35, 'matrixInfos', '(Streng) Vertrauliche Infos', '3', 3),
36 (36, 'matrixAnzahl', '1', '1', 1),
37 (37, 'matrixAnzahl', '2-3', '2', 2),
38 (38, 'matrixAnzahl', '> 3', '3', 3),
39 (39, 'auswirkung', 'gering', '1', 1),
40 (40, 'auswirkung', 'mittel', '2', 2),
41 (41, 'auswirkung', 'groß', '3', 3),
42 (42, 'auswirkung', 'sehr groß', '4', 4),
43 (43, 'reaktionszeit', '10 Std.', '600', 1),
44 (44, 'reaktionszeit', '4 Std.', '240', 2),
45 (45, 'reaktionszeit', '2 Std.', '120', 3),
46 (46, 'reaktionszeit', '15 Min.', '15', 4),
47 (61, 'currentState', 'neu', 'NEW', 1),
48 (60, 'anrede', 'Herr', 'Herr', 2),
49 (59, 'anrede', 'Frau', 'Frau', 1),
50 (47, 'contactrole', 'Vorfallmelder', 'MELDER', 0),
51 (48, 'contactrole', 'Hotliner', 'HOTLINER', 0),
52 (49, 'contactrole', 'SI Coordinator', 'SIC', 0),
53 (50, 'contactrole', 'CSIRT Mitglied', 'CSIRT', 1),

```

```

54 (51, 'contactrole', 'Administrator', 'ADMIN', 2),
55 (52, 'contactrole', 'Abteilungsleiter', 'ABTL', 3),
56 (53, 'contactrole', 'Gruppenleiter', 'GRUL', 4),
57 (57, 'contactrole', 'SFH-Vertreter', 'SFH', 0),
58 (55, 'contactrole', 'Major Incident Coordinator', 'MIC', 0),
59 (56, 'contactrole', 'LRZ-Leitung', 'CEO', 0),
60 (54, 'contactrole', 'Sonstige', 'SONST', 5),
61 (62, 'currentState', 'Klassifikation', 'CLASS', 2),
62 (63, 'currentState', 'Analyse', 'ANAL', 3),
63 (64, 'currentState', 'Lösung', 'SOL', 4),
64 (65, 'currentState', 'Monitoring', 'MON', 5),
65 (66, 'currentState', 'abgeschlossen', 'CLOSED', 6),
66 (67, 'hochschulstart', 'Nein', '0', 2),
67 (68, 'hochschulstart', 'Ja', '1', 1),
68 (69, 'majorExpected', 'Nein', '0', 2),
69 (70, 'majorExpected', 'Ja', '1', 1),
70 (71, 'klassifikation', 'Security Incident', 'INCIDENT', 1),
71 (72, 'klassifikation', 'Duplikat', 'DUPLICATE', 2),
72 (73, 'klassifikation', 'Information Request', 'REQUEST', 3),
73 (74, 'closedState', 'Request beantwortet', 'IRAW', 4),
74 (75, 'closedState', 'Request weitergeleitet', 'IRFWD', 5),
75 (76, 'closedState', 'SI Duplikat', 'SIDUPLICATE', 3),
76 (77, 'closedState', 'SI abgebrochen', 'SICANCEL', 2),
77 (78, 'closedState', 'SI erfolgreich', 'SISUCCESS', 1),
78 (79, 'currentState', 'IR Antwort', 'IRAW', 0),
79 (80, 'currentState', 'IR Weiterleitung', 'IRFWD', 0),
80 (81, 'contactrole', 'Allianz für Cybersicherheit', 'ACS', 0),
81 (82, 'lsgType', 'Installation eines Patches / Softwareupdates', 'PATCH', 5),
82 (83, 'lsgType', 'Sperrung einer Kennung', 'LOGINLOCK', 6),
83 (84, 'lsgType', 'Zurücksetzen einer Kennung', 'LOGINRESET', 7),
84 (85, 'lsgType', 'Einrichtung/Anpassung einer Firewallregel', 'FIREWALL', 8),
85 (86, 'lsgType', 'Sonstige', 'SONST', 127),
86 (87, 'saldo', 'Reaktionszeit', 'saldo1bh', 1),
87 (88, 'saldo', 'Analysezeit', 'saldo2bh', 2),
88 (89, 'saldo', 'Lösungszeit', 'saldo3bh', 3),
89 (90, 'saldo', 'Nachlaufzeit', 'saldo4bh', 4),
90 (91, 'kunde', 'HM', 'HM', 4),
91 (92, 'kunde', 'Hochschulstart', 'SFH', 6);

```

D. PDF-Export Musterdokument

Auf den drei folgenden Seiten ist ein beispielhafter Incident Report abgebildet, der durch die Aktion `si-details` des Moduls `export-pdf` dynamisch generiert wurde. Aus pragmatischen Gründen wurden teilweise Platzhaltertexte eingesetzt.

SI-46: Musterincident für die Erstellung eines PDF-Reports

Incident-Daten

Beschreibung	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.
Vorfallzeitpunkt	zwischen 09.06. und 11.06.2014
Kunde	Hochschulstart
Betriebssystem	Linux
Hostname	hochschulstart.lrz.de
IP-Adresse	123.124.125.126

Meta-Informationen

Art des Vorfalls	Meldezeitpunkt	Aktueller Status	Abgeschlossen
Security Incident Kompromittierter Webserver	2014-06-05 14:51:04	Monitoring 2014-06-28 20:47	

Reaktionszeit	Analysezeit	Lösungszeit	Nachlaufzeit
11 Std. 9 Min. (3 Tage 1 Std. 4 Min.)	5 Tage 18 Std. 20 Min. (18 Tage 18 Std. 26 Min.)	5 Std. 40 Min. (1 Tag 10 Std. 28 Min.)	

Klassifikation

Auswirkung	Priorität	Reaktionszeit	MI zu erwarten	Hochschulstart
groß	hoch	2 Std.	Nein	Ja

Standort Zielsystem	LRZ-intern
Standort Quellsystem	extern
Betroffene Dienste	Wichtige LRZ-Dienste
Betroffene Informationen	(Streng) Vertrauliche Infos
Anzahl der betroffenen Systeme	1

Abbildung D.1.: Beispiel für einen PDF-Report (Seite 1)



Security Incident Report
 LRZ Security Incident Management System
 Stand: 2014-06-29 13:58 / Schweiger, Aaron

Security Incident Team

Funktion	Name	Telefon	E-Mail
Hotliner	Metzger, Stefan	+49-89-35831-8846	Stefan.Metzger@lrz.de
Administrator	Pöhn, Daniela	+49-89-35831-8763	Daniela.Poehn@lrz.de
SI Coordinator	Reiser, Helmut	+49-89-35831-7854	Helmut.Reiser@lrz.de
Vorfallelder	Schweiger, Aaron	+49-8131-7791500	mail@aaron-schweiger.de

Maßnahmen und Analyseergebnisse

Erstmaßnahmen	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.
Maßnahmen	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.
Analyseergebnisse	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

Wiederherstellung Regelbetrieb und Dokumentation der Lösung

Regelbetrieb	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.
Typ der Lösung	Anpassung der Systemkonfiguration
Lösung	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

Abbildung D.2.: Beispiel für einen PDF-Report (Seite 2)


	Security Incident Report LRZ Security Incident Management System Stand: 2014-06-29 13:58 / Schweiger, Aaron
Monitoring	
Dauer	7 Tage
Intervall	8 Stunden
Aufgaben und Verantwortlichkeiten	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.
Auffälligkeiten	keine
Post Incident Review	
Positive Erfahrungen	
Negative Erfahrungen	
Optimierungsvorschläge	
Seite 3 / 3	

Abbildung D.3.: Beispiel für einen PDF-Report (Seite 3)