MIN-TEC Sicherheitsprobleme

Robin Meis

Aachen, 3. April 2018

Disclaimer

Alle in dieser Dokumentation enthaltenen Informationen wurden nach besten Wissen und Gewissen erstellt. Während sämtlicher Tests, wurde soweit möglich, auf im Internet offen dokumentierte Testinstallationen zurückgegriffen. Es wurden zu keinem Zeitpunkt private Daten ausgelesen, verändert oder gelöscht.

Die Dokumentation wurde unentgeltlich als Hobbyprojekt in meiner Freizeit erstellt. Sie steht in keinem Zusammenhang mit meiner Arbeit als Kleingewerbetreibender oder meinem Studium. Gegenüber der Firma EDV-Schaupp GmbH und anderen beteiligten Firmen oder Organisationen bestehen ausdrücklich keine finanziellen Forderungen.

Die an die Firma EDV-Schaupp GmbH übermittelte Version dieser Dokumentation wurde mittels GPG signiert, sodass ein Nachweis über den tatsächlich übermittelten Inhalt jederzeit möglich ist.

Inhaltsverzeichnis

1 Ausgangslage

1.1 IT in Schulen

Im Zeitalter digitaler Innovationen müssen auch Schulen ihre Infrastruktur laufend anpassen und erweitern. Mit der zunehmenden Digitalisierung ergeben sich jedoch auch neue Problemstellungen in Bezug auf Datenschutz und IT Sicherheit. Insbesondere die Schule als Schutzraum muss an dieser Stelle besondere Pflichten erfüllen, um die Daten der Schüler sicher vor Zugriffen Dritter zu speichern. Leider stellte sich in persönlichen Erfahrungen immer wieder heraus, dass Schulen häufig mit erheblichen Problemen im Bereich der IT Sicherheit konfrontiert werden. Derartige Probleme gilt es frühzeitig zu erkennen und geeignete Gegenmaßnahmen zu ergreifen.

1.2 Bargeldlose Zahlungssysteme

Bargeldlose Zahlungssysteme erfreuen sich zunehmender Beliebtheit. Diese Entwicklung zieht auch in die Mensen der Schulen ein, wo neben der bargeldlosen Bezahlung der Fokus auch auf einer einfachen Bestellung sowie der transparenten und diskreten Abrechnung von Sozialleistungen liegt.

1.3 Über mich

Mein Name ist Robin Meis. Ich habe mich bereits früh mit Software- und Hardwareentwicklung sowie deren Sicherheit gegenüber Angreifern beschäftigt. So habe ich in der Vergangenheit bereits Schwachstellen von Übertragungsprotokollen im Bahnbereich sowie mangelnde Manipulationssicherheit von abiturrelevanten Taschenrechnern nachgewiesen. Teile meiner bisherigen Arbeit habe ich dabei auf meiner Internetseite[1] dokumentiert.
Auch im schulischen Bereich zeigten sich kontinuierlich Probleme im Bereich der Netzwerksicherheit. Dabei habe ich sowohl die Systeme eines allgemeinbildenen Gymnasiums sowie einer berufsbildenen Schule kennengelernt. Dort legte im Jahr 2017 mein Abitur mit dem Schwerpunkt Elektrotechnik und Mathematik ab. Seit dem Wintersemester 2017/18 studiere ich Informatik an der RWTH Aachen.
In Aachen wurde ich zudem durch ein Projekt des Chaos Computer Clubs Teil der dortigen TheThingsNetwork Community, die es sich zum Ziel gesetzt hat ein offenes Netzwerk für IoT Anwendungen aufzubauen.
Ich verfolge weiterhin gelegentlich die Aktivitäten beider Schulen. Dabei fiel mir auch der Einsatz der Software MIN-TEC auf. Nach einer kurzen Betrachtung ergaben sich drei offensichtliche Angriffsmöglichkeiten für SQL Injections, die mich zu weiteren Tests veranlassten. Dabei konnte ich ein weiteres Problem im Bereich des Bezahlprozesses feststellen. Beim Verfassen der Dokumentation entdeckte ich zudem ein drittes Problem, dass zu ungewollt gespeicherten Passwörtern führt. Ziel dieser Dokumentation ist es, die Sicherheitslücken für den Hersteller nachvollziehbar und vollständig zu dokumentieren, um eine möglichst schnelle Behebung zu ermöglichen.

[1] `https://robin.meis.space/`

2 MIN-TEC

MIN-TEC ist eine von der EDV-Service Schaupp GmbH entwickelte Software zur Verwaltung von Essensbestellungen in Schulen. Die Bezahlung erfolgt dabei bargeldlos. Zur Authentifizierung dient dabei eine GiroCard. Die Bestellung erfolgt über eine Internetseite, bei der sich die Schüler unter Angabe ihrer Kartennummer und eines Passworts anmelden.

2.1 Zertifizierungen

2.1.1 Grundsätze ordnungsmäßiger Buchführung (GoB)

Die MIN-TEC Version 4.7 2016.4.6173 wurde am 30.11.2016 durch die Wirtschaftsprüfungsgesellschaft Baker Tilly Roelfs AG

> im Hinblick auf die Einhaltung der Anforderungen der Grundsätze ordnungsmäßiger Buchführung (GoB), wie sie sich aus den handels- und steuerrechtlichen Vorschriften ableiten

zertifiziert[2].

2.1.2 Externe Sicherheitsprüfung

Am 08.03.2018 wurden die Ergebnisse einer Sicherheitsprüfung[3] nach dem Blackbox Prinzip durch ein externes Unternehmen veröffentlicht. Darin wurde der Software mit einem Risikoergebnis von 0 aus 125 die höchstmögliche Sicherheitsstufe bescheinigt.

2.2 Betreiber der Software

Die meisten Schulen greifen auf ein externes Hosting von MIN-TEC zurück, anstatt die Software auf eigenen Servern zu betreiben. Es gibt jedoch auch andere Hoster.

2.2.1 Hosting

Mithilfe einer Google Suche wurde ein Teil der Anbieter ermittelt.

- bestellen.schulon.org
  Betreiber (whois): Kommunales Rechenzentrum Niederrhein
- schulmensa.net
  Betreiber (whois): EDV-Service Schaupp GmbH
- meinessen.net
  Betreiber (whois): EDV-Service Schaupp GmbH
- schulessen.net
  Betreiber (whois): EDV-Service Schaupp GmbH
- essenbestellen.net
  Betreiber (whois): EDV-Service Schaupp GmbH
- schulcatering.net
  Betreiber (whois): EDV-Service Schaupp GmbH
- min-tec.de
  Der Betreiber wurde aufgrund der Einschränkungen bei whois Abfragen für die ccTLD .de nicht geprüft.
- 100pro-schulverpflegungplus.de
  Betreiber: Kreissparkasse Köln (aus Weiterleitung abgeleitet, kein whois)

[2] Softwarebescheinigung (GoB): `http://www.edv-schaupp.de/data/files/schaupp-(e-urkunde-min-tec-v4.7-2016.4.6173-inkl.-aab-2016-12-05-final).pdf`

[3] Sicherheitsprüfung: `http://www.edv-schaupp.de/news/aktuelles/min-tec-online-bestell-und-abrechnungssystem-mit-hoechste`

Es zeigt sich, dass das Hosting in den meisten Fällen durch die EDV-Service Schaupp GmbH erfolgt. Dies ist der Anwendungssicherheit zuträglich, da mögliche Sicherheitsaktualisierungen an der Software zeitnah durch den Hersteller der Software erfolgen können. Im Umkehrschluss bedeutet dies, dass die dort eingesetzten Versionen keine dem Hersteller bekannten Sicherheitslücken aufweisen sollten.

2.2.2 Demo Installationen

Es werden verschiedene Demo Installationen betrieben. Dazu werden für den Bestellbereich Benutzerdaten öffentlich bekanntgegeben

- `https://bestellen.schulon.org/Sparkasse-Krefeld/Demo2/Vorbesteller/`
  Benutzername: Gast
  Passwort:123456 (Version: 2017.1.6249_3)
- `https://bestellen.schulon.org/sparkasse-krefeld/demogymnasium/Vorbesteller/`
  Benutzername: Gast
  Passwort:123456 (Version: 2017.1.6249_3)
- `http://www.100pro-schulverpflegungplus.de/test`
  Benutzername: Gast
  Passwort:123456

2.3 HTML Formulare

Anhand der Google Suche konnten drei HTML Formulare gefunden werden, die im Ordner der jeweiligen Installation abgelegt werden und ohne einen Login zugänglich sind. Je nach verwendetem Design können die auf den Screenshots abgebildeten Ansichten von anderen Installationen abweichen.

2.3.1 default.aspx

Enthält die Login Maske für den normalen Besteller. Dabei werden Benutzername bzw. Kartennummer und das Passwort abgefragt.

2.3.2 Passwort vergessen

Manche Installationen bieten auf der Seite default.aspx einen Button zum Zurücksetzen des Passworts. Es öffnet sich ein Eingabefeld, dass die Eingabe der Buchungsnummer bzw. der E-Mail Adresse ermöglicht.

2.3.3 office.aspx

Verwaltungsoberfläche für den Administrator. Es wird Name und Passwort abgefragt. Ab dem dritten Fehlversuch wird ein Captcha eingeblendet.

3 SQL Injections

3.1 Problembeschreibung

Alle drei öffentlich zugänglichen Eingabefelder für den Login als Besteller bzw. Administator und der Dialog zum Zurücksetzen des Passworts erlauben offenbar SQL Injections und könnten somit tiefgehende Zugriffe auf die Datenbank ermöglichen. Für diese Tests wurden lediglich die Bedingungen der Abfragen manipuliert, jedoch keine Daten verändert oder gelöscht.

3.2 Reproduktion

Im Folgenden werden die Beweise für die theoretische Möglichkeit einer SQL Injection geführt. Das Verhalten der Software impliziert nicht, dass durch Veränderung der ursprünglichen SQL Statements tatsächlich Daten unbefugt ausgelesen, verändert oder gelöscht werden können.

3.2.1 Login Besteller

Dieser Test wurde gegen die folgenden Installationen durchgeführt:

- `https://bestellen.schulon.org/sparkasse-krefeld/Demo2/Vorbesteller/`
- `https://www.schulcatering.net/mensa-geporta/vorbesteller/`
- `https://100pro-schulverpflegungplus.de/szneuenhof/vorbesteller/`
- `https://www.schulmensa.net/hcs/sz-evks/Vorbesteller/`

Zunächst wird das Verhalten der Anwendung für den Fall betrachtet, dass Benutzername und Passwort falsch sind. Hierzu sei der Benutzername *sql* und das Passwort *injection*. In diesem Fall melden alle vier getesteten Installationen

> Anmeldung fehlgeschlagen

Nun wird der Benutzername anstatt *sql* auf *'* geändert. Im Ergebnis erfolgt bei allen getesteten Installationen kein Login. Es wird jedoch auch keine Rückmeldung gegeben. Die einzige Ausnahme stellt hierbei die Installation *szneuenhof* dar. Hier erfolgt die Rückmeldung

> Anmeldung fehlgeschlagen

Das Verhalten der übrigen drei Installationen ist ein erster Indikator für eine mögliche SQL Injection. Daher wird nun versucht die SQL Abfrage nachzubilden:

```
SELECT * FROM users WHERE username = '$user' and password = '$pwd'
```

Indem der Benutzername auf einen einzelnen Apostroph gesetzt wurde, wurde offensichtlich der String in der Abfrage verlassen, sodass die Syntax der SQL Query ungültig wurde und es zu einem unbehandelten Server Fehler kam.
Um im letzten Schritt zu prüfen, ob es sich wirklich um einen Syntax Fehler handelt, wird ein zweites Apostroph hinzufügt, sodass der Benutzername als *''* gesetzt wird. Somit bleibt die SQL Syntax in Takt. Aufgrund des nun weiterhin falschen Benutzernamens, aber intakter SQL Syntax erfolgt bei allen Installationen nun wieder die Meldung

> Anmeldung fehlgeschlagen

Somit ist davon auszugehen, dass die SQL Syntax durch das Hinzufügen eines Apostrophs ungültig wird. Dieser Umstand impliziert die Möglichkeit einer SQL Injection in der Login Maske.

3.2.2 Passwort zurücksetzen

Dieser Test wurde gegen die folgenden Installationen durchgeführt:

- `https://bestellen.schulon.org/sparkasse-krefeld/Demo2/Vorbesteller/`
- `https://www.schulcatering.net/mensa-geporta/vorbesteller/`
- `https://www.schulmensa.net/hcs/sz-evks/Vorbesteller/`

100pro-schulverpflegungplus.de wurde nicht getestet, da dort das Zurücksetzen des Passworts abgeschaltet ist.
Die Identifikation eines Benutzers erfolgt in diesem Fall entweder über die Buchungsnummer oder die E-Mail Adresse. Damit MIN-TEC eine E-Mail Adresse erkennt und überprüft, muss die angegebene Zeichenkette ein @ enthalten. Der Test wird zunächst analog zum bereits durchgeführten Test für die Login Maske durchgeführt, jedoch ein @ am Anfang vor die Apostrophe gestellt. Die Eingabe lautet nun also *@'*. Dabei stellt sich heraus, dass die Installationen *schulon.org* und *schulmensa.net* den HTTP Status 500 (Internal Server Error) in der Netzwerkkonsole des Browsers melden. Für die Installation *schulcatering.de* wird 404 zurückgegeben. Dieses Verhalten ist ein erster Indikator für eine mögliche SQL Injection.
Nun wird ein zweiter Apostroph hinzugefügt, sodass die Eingabe *@"* lautet. Nun geben alle drei Installationen die Meldung

> Unbekannte Mailadresse

zurück.

Im nächsten Schritt wird nun versucht eine Bedingung zu erzeugen, die für alle Einträge in der Datenbank wahr ist. Dazu wird zunächst die Ursprüngliche SQL Query rekonstruiert:

```
SELECT * FROM users WHERE email = '$email'
```

eine mögliche Eingabe lautet wie folgt:

```
' OR '@' = '@
```

Da @ immer identisch zu @ ist, wird dieser Teil der Aussage immer zu wahr auswerten. Eingesetzt in das gesamte SQL Statement ergibt sich:

```
SELECT * FROM users WHERE email = '' OR '@' = '@'
```

Es zeigt sich, dass die Aussage nur für die Einträge in der Datenbank wahr ist, bei denen eine leere E-Mail Adresse hinterlegt ist. Da diese Teilaussage jedoch mit der Tautologie '@' = '@' verodert ist, wird nun jede Zeile zurückgegeben. Unter Verwendung dieser Eingabe meldet MIN-TEC zurück:

> Diese Mailadresse ist bei mehreren Kunden hinterlegt. Somit kann keine eindeutige Zuordnung stattfinden. Wenn vorhanden versuchen Sie es bitte mit Ihrer Buchungs-Nummer oder wenden Sie sich an den System-Administrator

Dieses Verhalten ist korrekt, da nun alle Kunden aus der Datenbank zurückgegeben werden, und keine eindeutige Zuordnung zu einem einzelnen Kunden mehr möglich ist. Es zeigt jedoch auch, dass die SQL Injection an dieser Stelle erfolgreich war.

3.2.3 Login Administrator

Dieser Test wurde gegen die folgenden Installationen durchgeführt:

- `https://bestellen.schulon.org/sparkasse-krefeld/Demo2/Vorbesteller/Office.aspx`
- `https://www.schulcatering.net/mensa-geporta/vorbesteller/Office.aspx`
- `https://100pro-schulverpflegungplus.de/szneuenhof/vorbesteller/Office.aspx`
- `https://www.schulmensa.net/hcs/sz-evks/Vorbesteller/Office.aspx`

Hierbei wird analog zum Login für Besteller verfahren. Im Fall eines einzelnen Apostrophs im Benutzernamen, also bei invalider SQL Syntax wird die folgende Meldung ausgegeben:

> Datenbank zur Zeit nicht verfügbar

Sobald der Benutzername mit zwei Apostrophen oder als normaler Text übergeben wird, wird die Meldung

> Zugriff verweigert

zurückgegeben. Auch hier impliziert das Verhalten eine Anfälligkeit für SQL Injections.

3.3 Lösungsvorschlag

Mittels prepared statements oder geeigneten escape Funktionen kann eine SQL Injection verhindert werden.

4 Benutzerdaten in der Browserchronik

4.1 Problembeschreibung

Auf der Administrationsseite Office.aspx werden die Formulardaten Benutzername und Passwort als GET Request übermittelt:

```
&username=test&txtPwd=test
```

Dadurch werden die Zugangsdaten in der Browserchronik gespeichert. Dies ist vor allem bei Logins an fremden Computern problematisch.

4.2 Problemlösung

Die Formulardaten sollten als POST Request übermittelt werden.

5 Preismanipulation

5.1 Problembeschreibung

Aufgrund einer Sicherheitslücke im Bestellprozess kann ein Benutzer den Preis, den er für ein Essen bezahlen möchte selbst festlegen. Dabei kann er den Preis ändern und dabei auch auf 0,00€ setzen.
Alle Tests wurden gegen `https://bestellen.schulon.org/Sparkasse-Krefeld/Demo2/Vorbesteller/` durchgeführt.

5.2 Üblicher Bestellablauf

Im Folgenden wird der Ablauf einer Bestellung mit den damit verbundenen HTTP Übertragungen beschrieben.
Der Benutzer wählt zunächst ein Menü aus, indem er entweder auf das entsprechende Feld in der Tabelle klickt, oder die gewünschte Anzahl über das Dropdown Feld auswählt.
Unmittelbar nach der Auswahl erfolgt ein HTTP Post Request an */OrderForm.aspx/setBestellung*.
Dabei werden die folgenden Parameter als JSON übermittelt:

- **dbdate** 2018-04-11
  Datum, an dem das Gericht serviert wird
- **esshash** 9A7BCE8A810396D416B2156BCFC683B8
  Ein Hashwert
- **menuanzahl** 1
  Anzahl der bestellten Gerichte
- **menulinie_oid** 1
  Zeile in der Tabelle
- **menupreis** 4,20
  Einzelpreis des Gerichts

Der Server meldet auf diese Anfrage

> {“success“:true,“usermessage“:“Menu wurde Warenkorb hinzugefügt“,“adminmessage“:“Menu wurde Warenkorb hinzugefügt 2018-04-11 1 x Menü 1 zu je 4,20“,“parameter“:“503“}

Anschließend klickt der Benutzer auf *Kasse* und auf *Bezahlen*. Das Guthaben wurde nun von seinem Konto abgezogen.

5.3 Erfolgloser Angriff

5.3.1 Ablauf

In einem ersten Versuch soll nun lediglich der Parameter menupreis geändert werden. Die neuen Daten lauten wie folgt:

- **dbdate** 2018-04-11
  Datum, an dem das Gericht serviert wird
- **esshash** 9A7BCE8A810396D416B2156BCFC683B8
  Hashwert
- **menuanzahl** 1
  Anzahl der bestellten Gerichte
- **menulinie_oid** 1
  Zeile in der Tabelle
- **menupreis** 0,00
  Einzelpreis des Gerichts

Der Server meldet nun

> {“success“:false,“usermessage“:“Fehler bei der Bestellung“,“adminmessage“:“Bestell Hash ist Falsch“,“parameter“:““}

Offenbar stimmt nun der *esshash* nicht mehr mit den gesendeten Daten überein, sodass der Angriff erfolgreich verhindert wird. Um den Angriff erfolgreich durchführen zu können, muss die Funktion zur Erzeugung des Hashwertes rekonstruiert werden.

5.4 Hashfunktion

Im Folgenden Abschnitt wird die Rekonstruktion der Hashfunktion beschrieben, um einen erfolgreichen Angriff auf den übermittelten Menüpreis durchführen zu können.

5.4.1 Grundgedanke

Da der Browser bei einer Bestellung den Preis des Gerichts an den Server schickt, könnte ein Angreifer den Preis in den gesendeten Daten wie oben beschrieben verändern und somit kostenlos essen. Um zu verifizieren, dass der Benutzer den Preis nicht verändert hat, wird ein Hash mit einer dem Angreifer unbekannten Hashfunktion erzeugt. Dieser Hash dient somit als Signatur.

5.4.2 Funktionsweise

Sobald der Benutzer den Speiseplan abruft, wird die Übersichtstabelle serverseitig erzeugt. Dabei wird auch für jedes Gericht der jeweilige esshash berechnet. Dieser besteht aus den jeweiligen Attributen des Gerichts wie Datum oder Preis. Der esshash wird neben dem Preis, dem Datum sowie der Zeile in der Tabelle als Attribut zu dem jeweiligen Gericht im HTML abgelegt.
Klickt ein Benutzer nun auf das gewünschte Gericht, liest eine JavaScript Funktion die jeweiligen Attribute aus und schickt sie an den Server.
Der Server bildet nun den Hash aus den übermittelten Daten und vergleicht sie mit dem übermittelten Hash. Stimmen beide Hashwerte überein, nimmt der Server an, dass die Daten durch den Client nicht verändert wurden.

5.4.3 Verwendete Daten

In den Hashwert können nur die Daten einfließen, die der Benutzer nicht verändern kann. Die Anzahl der bestellten Menüs entfällt also als Einfluss auf den Hashwert. Es bleiben somit als verwendete Parameter das Datum, der Preis sowie die Zeile in der Tabelle.

5.4.4 Hash Algorithmus

Die verwendeten Daten werden als String konkateniert und anschließend mit einer kryptographischen Hashfunktion gehasht. Bevor die Reihenfolge der Konkatenation der Daten ermittelt werden kann, muss daher ermittelt werden, mit welcher Funktion die Daten gehasht werden.
Der Hash weist eine Länge von 128 Bit auf. Dies entspricht der Länge des weit verbreiteten MD5 Algorithmus, weshalb an dieser Stelle zunächst angenommen wird, dass MD5 zum hashen der Daten zum Einsatz kommt.

5.4.5 Konkatenation

Bevor die Daten gehasht werden, müssen sie als String konkateniert werden. Hierbei ist zunächst unklar, in welche Reihenfolge die Daten konkateniert werden, und welche Trennzeichen dazwischen verwendet werden.
Die Reihenfolge der Daten lässt sich durch Probieren ermitteln. Um den Aufwand zu minimieren, wurde ein Python Programm geschrieben, das definierbare Trennzeichen zwischen alle möglichen Permutationen der angegebenen Werte schreibt. Anschließend wird der entstandene String per MD5 gehasht. Da Datum, Preis, Zeile in der Tabelle sowie der korrekte esshash aus dem Speiseplan bekannt sind, kann nun der berechnete Hash mit dem bekannten Hashwert verglichen werden. Stimmen beide Hashes überein, wurde die korrekte Reihenfolge gefunden.

```
import itertools
import hashlib

delimiters = ["","␣", ",", "-", "_"]          #Trennzeichen
values = ["2018-04-11", "4,20", "1"]          #Bekannte Daten
esshash = "9A7BCE8A810396D416B2156BCFC683B8" #Bekannter Hash

permutations = list(itertools.permutations(values))

for permutation in permutations: #Alle Permutationen durchlaufen
    for delimiter in delimiters:
        string = ""
        for element in permutation: #String mit Trennzeichen erzeugen
            string += element + delimiter
        print(string[0:-1])
        m = hashlib.md5()
        m.update(bytes(string, "UTF-8"))
        print(m.hexdigest())

        if (m.hexdigest().upper() == esshash.upper()): #Hashes vergleichen
            print("FOUND!")
            print(permutation)
            exit()
```

Die letzte getestete Permutation lautet wie folgt:

```
12018-04-114,2
9a7bce8a810396d416b2156bcfc683b8
FOUND!
('1', '2018-04-11', '4,20')
```

Nun ist bekannt, in welcher Reihenfolge die Daten konkateniert werden und, dass keine Trennzeichen verwendet werden.

5.4.6 Rekonstruierte Funktion

Aus den nun bekannten Daten kann ein Hashfunktion implementiert werden, die für beliebige Werte einen gültigen Hash berechnet.

```
import hashlib

#dbdate im Format yyyy-mm-dd (string)
#menulinie_oid (string)
#menupreis: Nullen muessen erhalten bleiben, also als String uebergeben

def esshash(dbdate, menulinie_oid, menupreis):
    m = hashlib.md5()
    m.update(bytes(menulinie_oid + dbdate + menupreis, "UTF-8"))
    return m.hexdigest().upper()

print (esshash("2018-04-11", "1", "0,00"))
```

Nach Ausführung dieses Programms ergibt sich der folgende Hash:

1B25031211CD97983B8C9973CFCDEDD6

5.4.7 Test: Kostenlose Bestellung

Mit dem neu berechneteen Hashwert muss nun in einem weiteren Test die kostenlose Bestellung möglich sein. Nach dem Senden der Anfrage

- **dbdate** 2018-04-11
  Datum, an dem das Gericht serviert wird
- **esshash** 1B25031211CD97983B8C9973CFCDEDD6
  Hashwert
- **menuanzahl** 1
  Anzahl der bestellten Gerichte
- **menulinie**_oid 1
  Zeile in der Tabelle
- **menupreis** 0,00
  Einzelpreis des Gerichts

antwortet der Server

{“success“:true,“usermessage“:“Menu wurde Warenkorb hinzugefügt“,“adminmessage“:“Menu wurde Warenkorb hinzugefügt 2018-04-11 1 x Menü 1 zu je 0,00“,“parameter“:“503“}

An der Kasse wird nun eine Gesamtsumme von 0,00€ gemeldet und der Kontostand auch nach dem Klick auf *bezahlen* nicht verringert.

Somit ist bewiesen, dass der Hashwert nicht ausreicht, um einen wirksamen Schutz vor der Manipulation des Preises durch einen Angreifer zu erreichen. Selbstverständlich wurden alle Bestellungen im Demosystem nach dem Test storniert.

5.5 Negative Preise

Wird ein negativer Preis, wie beispielweise *-4,20€* übermittelt und korrekt gehasht, wird sowohl eine Bestellung durchgeführt als auch das Guthaben des Nutzers erhöht.

5.5.1 Durchführung

Dazu werden die folgenden Daten an den Server geschickt:

- **dbdate** 2018-04-11
  Datum, an dem das Gericht serviert wird
- **esshash** A872060B1B3E1FE0A50B3A6DB4F5360F
  Hashwert
- **menuanzahl** 1
  Anzahl der bestellten Gerichte
- **menulinie_oid** 1
  Zeile in der Tabelle
- **menupreis** -4,20
  Einzelpreis des Gerichts

Der Server antwortet:

> {“success“:true,“usermessage“:“Menu wurde Warenkorb hinzugefügt“,“adminmessage“:“Menu wurde Warenkorb hinzugefügt 2018-04-11 1 x Menü 1 zu je -4,20“,“parameter“:“503“}

Abbildung 1: Die Kasse berechnet ein höheres Guthaben als neuen Kontostand

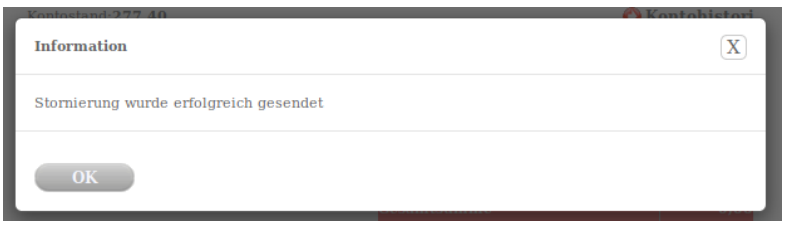

Abbildung 2: Laut Meldung wurde eine Stornierung durchgeführt

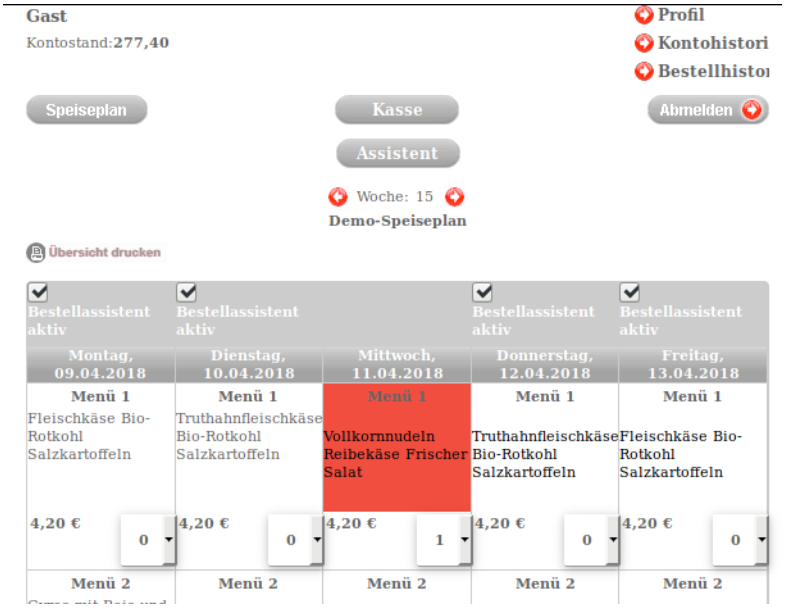

Abbildung 3: Das Gericht wurde trotz Stornierungsmeldung bestellt und das Kontoguthaben erhöht

5.5.2 Stornierung (Kasse)

Erfolgt die Stornierung an der Kasse, wird der ursprüngliche Kontostand wiederhergestellt.

5.5.3 Stornierung (Speiseplan)

In diesem Versuch wird das Gericht im Speiseplan abbestellt, indem die Anzahl auf *0* gesetzt wird. Anschließend wird an der Kasse die Stornierung abgeschlossen. Hierbei erfolgt erneut eine Gutschrift in Höhe des tatsächlichen Preises bzw. des Preises, der bei der Abbestellung übermittelt wurde. Somit erfolgt eine Doppelte Gutschrift.

5.6 Lösungsvorschlag

Anstatt den Preis bei der Bestellung erneut an den Server zu übertragen, sollte der Server den Preis für den Bezahlprozess selbst aus der Datenbank auslesen.
Dazu benötigt jedes Gericht eine eindeutige ID. Diese ID kann durch Auto Increment Werte durch die Datenbank automatisch vergeben werden. Sobald der Benutzer auf ein Feld zur Bestellung klickt, sendet er nicht mehr den Preis oder die Zeile der Tabelle an den Server, sondern die ID des jeweiligen Gerichts. Der Server liest dann im Anschluss selbstständig den Preis für die ID aus der Datenbank aus und fügt den Betrag dem Warenkorb hinzu. Somit ist eine Manipulation des Preises durch den Benutzer ausgeschlossen.
Von der Verwendung zusätzlicher Parameter im Hash oder eines Salts sollte abgesehen werden, da es sich dabei lediglich um eine Verlängerung des Strings handelt, jedoch nicht um einen langfristig wirksamen Schutz, der auch Angriffen in einem Grey- oder Whitebox Szenario widerstehen würde.

6 Aktuelle Versionen

Mithilfe mehrerer Google Suchen konnte eine Liste über MIN-TEC Installationen erstellt werden. Im Anschluss wurde jede Installation auf dieser Liste durch ein Python Programm aufgerufen und nach Merkmalen für die MIN-TEC Version durchsucht. Hierbei wird auf die Versionsnummer zurückgegriffen, die an den eingebundenen JavaScript Dateien als Parameter zum Schutz vor Caching verwendet wird:

```
<script type="text/javascript" src="/Vorbesteller/Scripts/ESS.js?4.7-2017.3.6444">
```

Dabei wurde als derzeit (Stand: 03.April 2018) aktuellste Version die Version *2018.1.6606.1* ermittelt. Die gefundenen Sicherheitslücken wurden erneut gegen drei Installationen dieser Version getestet.

- `https://bestellen.schulon.org/Sparkasse-Aachen/BPG/Vorbesteller/`
- `https://www.min-tec.de/Gemmrigheim/vorbesteller/`
- `https://100pro-schulverpflegungplus.de/szdd/vorbesteller/`

6.1 SQL Injection

6.1.1 Benutzer Login

In aktuellen Versionen wurde dieser Fehler behoben.

6.1.2 Passwort Wiederherstellung

In aktuellen Versionen wurde dieser Fehler behoben.

6.1.3 Administrator Login

Weiterhin anfällig. Bei ungültige SQL Syntax wird weiterhin

> Datenbank zur Zeit nicht verfügbar

zurückgegeben.

6.2 Benutzerdaten in der Browserchronik

In aktuellen Versionen wurde dieser Fehler behoben.

6.3 Preismanipulation

Da für diese Version derzeit keine Testinstallation mit vorhandenem Speiseplan existiert, konnte die Möglichkeit der Preismanipulation nicht abschließend getestet werden.
In der entsprechenden JavaScript Datei *MINTEC_FO.js* befindet sich jedoch weiterhin der Code, um das Attribut *SEC* auszulesen und den darin gespeicherten Hashwert an den Server zu übermitteln. Daher ist davon auszugehen, dass dieses Problem weiterhin existiert.

6.4 Aktualisierungen durch EDV-Schaupp GmbH

Auf Domains, die durch die EDV-Schaupp GmbH betrieben werden, werden weiterhin veraltete Versionen eingesetzt. Hier findet offenbar keine Aktualisierung bereits eingerichteter Installationen statt.

7 Abschließende Bewertung

Sämtliche Tests erfolgten aus Sicht eines potentiellen Angreifers, also als Blackbox-Test. Daher können Auswirkungen der gefundenen Schwachstellen nur teilweise abgeschätzt werden.

7.1 SQL Injections

Die SQL Injection Problematik lässt sich leicht beheben, die Auswirkungen jedoch nicht abschließend bewerten. Bei allen Tests wurde auf das Auslesen, Verändern oder Löschen von Daten verzichtet.
Offenbar wurde die Sicherheitslücke inzwischen in Teilen geschlossen, sodass sie in neueren Versionen ausschließlich im Administrator Login auftritt.
Durch weitere Tests ließe sich jedoch die Datenbank- und Tabellenstruktur rekonstruieren und möglicherweise Angriffe direkt auf die Datenbank durchführen. Somit wären sämtlichen Sicherheitsfunktionen von MIN-TEC als Anwendungsschicht ausgehebelt, sodass auch die Zertifizierung im Hinblick auf die Grundsätze ordnungsgemäßer Buchführung hinfällig wird.

7.2 Preismanipulation

Um die Problematik der Preismanipulation nachhaltig zu beheben, sind im Vergleich zur Behebung der SQL Injections größere Änderungen im Source Code erforderlich. Dabei gilt, dass der tatsächliche Aufwand, insbesondere in Bezug auf notwendige Änderungen an der Datenbank nicht abgeschätzt werden kann.

7.3 Zertifizierung der Software

Mit diesem Ergebnis muss ich zu dem Schluss kommen, dass die Sicherheitszertifizierung der Software MIN-TEC zu keinem Zeitpunkt hätte erteilt werden dürfen. Der Pentester hätte die Verwaltungsoberfläche mittels einer Google Suche finden können.
Sofern eine andere Version zertifiziert worden ist, als die Versionen, die ich getestet habe, so wird die EDV-Schaupp GmbH dringend aufgefordert Sicherheitsupdates für die derzeit eingesetzten Versionen anzubieten und durchzuführen.

7.4 Aktualisierungen

Selbst auf Servern der EDV-Schaupp GmbH sind weiterhin veraltete Versionen installiert, die für alle drei benannten SQL Injections anfällig sind. Offenbar werden nur einmalige Installationen durchgeführt und die Wartung anschließend vernachlässigt. An dieser Stelle wird die Firma EDV-Schaupp GmbH dringend aufgefordert veraltete Versionen auf einen aktuellen und somit sichereren Stand zu aktualisieren.
Da die Anzahl der vorhandenen SQL Injections vermindert wurde, ist davon auszugehen, dass dem Hersteller die Lücken inzwischen bekannt sind. Daher ist die mangelnde Aktualisierung als grob fahrlässig zu werten.

7.5 Google Ergebnisse

Ein Teil dieser Sicherheitsanalyse wurde durch Google Suchergebnisse ermöglicht. Die Seite office.aspx tauchte bei der Recherche ebenfalls in den Suchergebnissen auf. Es ist zu prüfen, ob Suchmaschinen mittels *robots.txt* angewiesen werden sollten, Teile der MIN-TEC Installationen nicht zu indizieren.

7.6 Risikoeinschätzung

7.6.1 Caterer

Da Caterer den durch MIN-TEC bereitgestellten Prozessen vertrauen, fallen Manipulationen durch Dritte möglicherweise erst spät oder nicht auf. In diesem Fall hat der Caterer mit finanziellen Schäden zu kämpfen, die je nach Aufmerksamkeit unentdeckt bleiben. Die Bestellung mit negativen Preisen führt dabei zudem zu doppelten Schäden.

7.6.2 Nutzer

Die Nutzer der Software MIN-TEC sind in den meisten Fällen Schüler. Deren Daten lassen sich mittels SQL Injections durch Dritte auslesen oder verändern. Insbesondere Schüler stellen jedoch eine besonders schützenswerte Gruppe dar, da sie mögliche Gefahren und Folgen eines Angriffs nicht erkennen und abschätzen können. Zudem gelten bei der Verarbeitung personenbezogener Daten Minderjähriger besondere Anforderungen.
Aufgrund der Einfachheit der aufgezeigten Sicherheitsprobleme, könnten Schüler auch Daten an ihrer eigenen Schule manipulieren und sich in Folge dessen mit juristischen Konsequenzen konfrontiert sehen.

7.6.3 Softwareentwickler

Die Firma EDV-Schaupp GmbH hat spätestens mit Erhalt dieser Dokumentation Kenntnis von den gefundenen Sicherheitslücken erlangt. Ein weiterer Vertrieb der Software MIN-TEC ohne die zeitnahe Behebung der genannten Probleme stellt grobe Fahrlässigkeit dar. Im Falle eines Angriffs auf eine oder mehrere Installationen könnte sich die EDV-Schaupp GmbH mit Regressforderungen konfrontiert sehen. Daher sind auch bisher installierte Versionen zeitnah zu aktualisieren.

Aufgrund der Risiken ist es für alle Beteiligten unverzichtbar, dass die gefundenen Sicherheitslücken in allen derzeit aktiven Installationen zeitnah behoben werden. Zudem besteht die Notwendigkeit, auch bereits eingerichtete Installationen zu aktualisieren.

8 Ausblick

Während der Tests zu den oben dokumentierten Problemen, wurden Hinweise auf weitere mögliche Probleme gefunden, die jedoch nicht näher analysiert wurden. Aus Gründen der Transparenz und Vollständigkeit werden sie dennoch aufgeführt.

8.1 Schwimmbadverwaltung

In Ahaus wird neben MIN-TEC[4] auch eine weitere Lösung zur Abrechnung in Schwimmbädern[5] mit der AHAUS CARD betrieben. Diese Installation ist gemäß der oben durchgeführten Tests ebenfalls anfällig für SQL Injections. Zusätzlich gibt der Server im Falle einer ungültigen SQL Syntax eine konkrete Fehlermeldung aus, die für ein weiteres Vorgehen genutzt werden könnte.

8.2 Captcha

Die Administration von MIN-TEC unter Office.aspx zeigt nach dem dritten Fehlerversuch beim Login ein Captcha an. Dieses Captcha verschwindet jedoch nach dem Entfernen der URL Parameter.
In neueren Versionen werden POST Requests verwendet. Auch hier reicht es jedoch aus, die Informationen in den versteckten Eingabefeldern zu verwerfen. Dies kann bereits durch den erneuten Aufruf der Seite erfolgen.
An dieser Stelle gilt es zu prüfen, ob das Captcha tatsächlich einen wirksamen Schutz gegen automatisierte Brute-force Angriffe darstellt. Schließlich würde ein entsprechendes Programm mögliche Rückgaben verwerfen, sofern der Login erfolglos bleibt.

8.3 SQL Injections

Die Nutzung der gefundenen SQL Injections lässt sich noch ausweiten, um Manipulationen an der Datenbank durchzuführen. Zusätzlich sei erwähnt, dass nur die Eingabefelder für Logins und die Passwort Wiederherstellung auf Anfälligkeit geprüft wurden. Die Software könnte beispielsweise in den persönlichen Profileinstellungen weitere Sicherheitslücken aufweisen.

[4] `https://bestellen.schulon.org/Ahaus/Vorbesteller/`
[5] `https://bestellen.schulon.org/Ahaus/portal/`