

Practical 6: Gruyère

Vincent Simonet, 2015-2016

This document: <https://goo.gl/9mOkaT>

The objective of this practical exercise is to learn about common vulnerabilities from web applications, using an online tutorial named Gruyère. This tutorial comes with a web application (hosted on AppEngine). This application is full of security vulnerabilities, and your task will be to exploit them in order to learn how to protect your own applications.

Important: Accessing or attacking a computer system without authorization is illegal in many jurisdictions. You should use what you learn from the practical to make your own applications more secure. You should not use it to attack any applications other than your own.

Set up

To access Gruyère, go to <http://google-gruyere.appspot.com/start>. AppEngine will start a new instance of Gruyère for you, assign it a unique id and redirect you to <http://google-gruyere.appspot.com/123/> (where 123 is your unique id). Each instance of Gruyère is "sandboxed" from the other instances so your instance won't be affected by anyone else using Gruyère. **You'll need to use your unique id instead of 123 in all the examples.**

The Gruyère source code is available online so that you can use it for white-box hacking. You can browse the source code at <http://google-gruyere.appspot.com/code/> or download all the files from <http://google-gruyere.appspot.com/gruyere-code.zip>. You do not need to run Gruyère locally in order to do the practical.

Get familiar with Gruyère

Before trying to hack Gruyère, get familiar with the application by completing the following tasks:

- View another user's snippets by following the "All snippets" link on the main page. Also check out what they have their Homepage set to.
- Sign up for an account for yourself to use when hacking. **Do not use the same password for your Gruyère account as you use for any real service.**
- Fill in your account's profile, including a private snippet and an icon that will be displayed by your name.
- Create a snippet (via "New Snippet") containing your favorite joke.
- Upload a file (via "Upload") to your account.

This covers the basic features provided by Gruyère. Now let's break them!

Breaking Gruyère

The practical consists in a series of challenges. Each challenge aims at illustrating some particular kind of vulnerabilities of web applications. Each challenge is tagged by a symbol indicating which techniques are required to solve them:



Challenges that can be solved just by using black box techniques



Challenges that require that you look at the Gruyère source code



Challenges that require some specific knowledge of Gruyère that will be given in the first hint

You should do as much as you can from the codelab. If you are making fast progress, that's great, let's do everything. If you're making slower progress, here is a copy of the table of contents where the most important sections are bolded:

- [Beat the hackers](#)
- [Gruyere](#)
- [Set-up](#)
 - [Reset Button](#)
 - [About the Code](#)
 - [Features and Technologies](#)
- [Using Gruyere](#)
- [Cross-Site Scripting \(XSS\)](#)
 - [XSS Challenges](#)
 - [File Upload XSS](#)
 - [Reflected XSS](#)
 - [Stored XSS](#)
 - [Stored XSS via HTML Attribute](#)
 - [Stored XSS via AJAX](#)
 - [Reflected XSS via AJAX](#)
 - [More about XSS](#)
- [Client-State Manipulation](#)
 - [Elevation of Privilege](#)
 - [Cookie Manipulation](#)
- [Cross-Site Request Forgery \(XSRF\)](#)
 - [XSRF Challenge](#)
 - [More about preventing XSRF](#)
- [Cross Site Script Inclusion \(XSSI\)](#)
 - [XSSI Challenge](#)
- [Path Traversal](#)
 - [Information disclosure via path traversal](#)
 - [Data tampering via path traversal](#)
- [Denial of Service](#)
 - [DoS - Quit the Server](#)
 - [DoS - Overloading the Server](#)
 - [More on Denial of Service](#)
- [Code Execution](#)
 - [Code Execution Challenge](#)
 - [More on Remote Code Execution](#)
- [Configuration Vulnerabilities](#)
 - [Information disclosure #1](#)
 - [Information disclosure #2](#)
 - [Information disclosure #3](#)
- [AJAX vulnerabilities](#)
 - [DoS via AJAX](#)
 - [Phishing via AJAX](#)
- [Other Vulnerabilities](#)
 - [Buffer Overflow and Integer Overflow](#)
 - [SQL Injection](#)
- [After the Codelab](#)