Check (mate) vulnerabilities!

Sebastián Revuelta

Some reflections about OWASP TOP 10 and chess game

Who am I?

- Sebastian Revuelta
  - Security Engineer at Thales Alenia Space
  - More than 15 years working in software security (and quality)
  - Application Security: DevSecOps | SAST | SCA
  - Blog: securingsoftware.blog
  - Chess player in my childhood

sebastianrevuelta@gmail.com
https://www.linkedin.com/in/sebasrevuelta/

Agenda

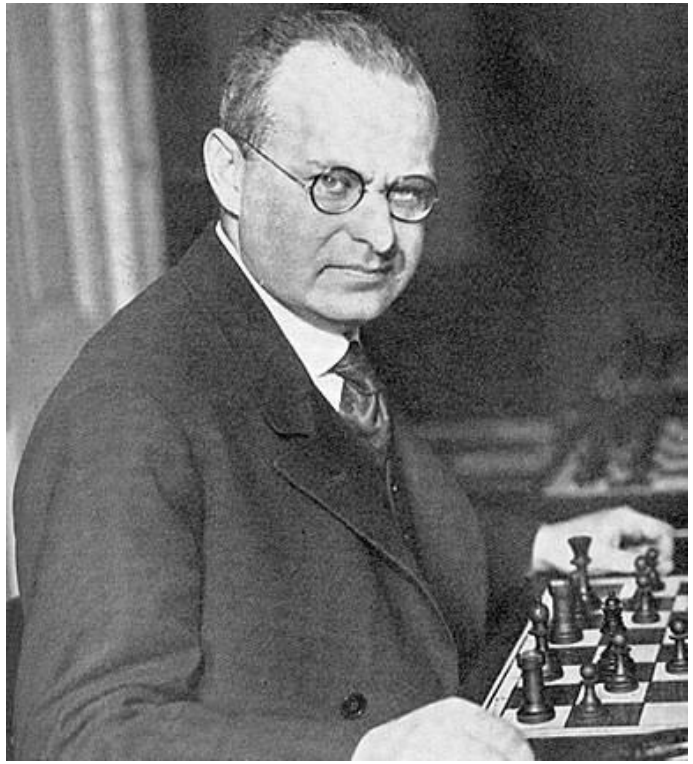
- Threats in application security
- Owasp top 10
- Playing vulnerable chess game
  - SQL Injection
  - Path Traversal
  - Cross site scripting
  - HTTP Response Splitting
- Deep code to win the match

Threats

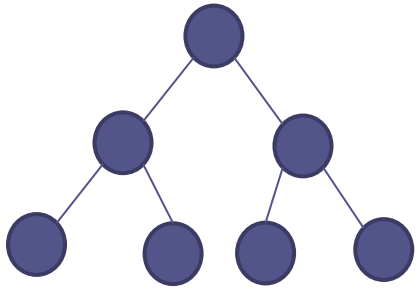
"The threat is stronger than the execution“ Aaron Nimzowitch (Latvia 1886, chess grandmaster).

We need to keep the tension and do our best all the time to avoid vulnerable points in our system.

Every move should be done to avoid risky points.

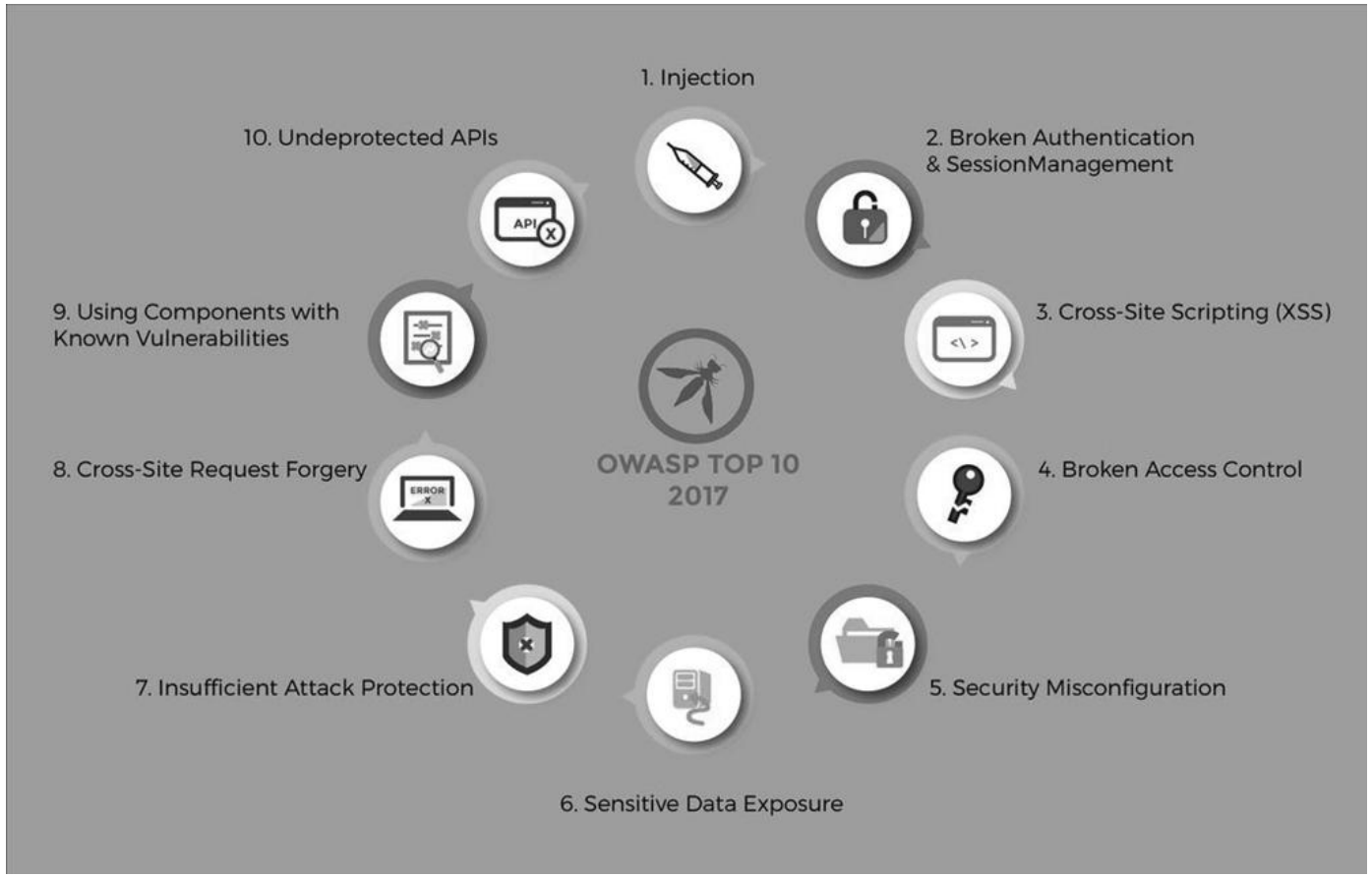


Threats



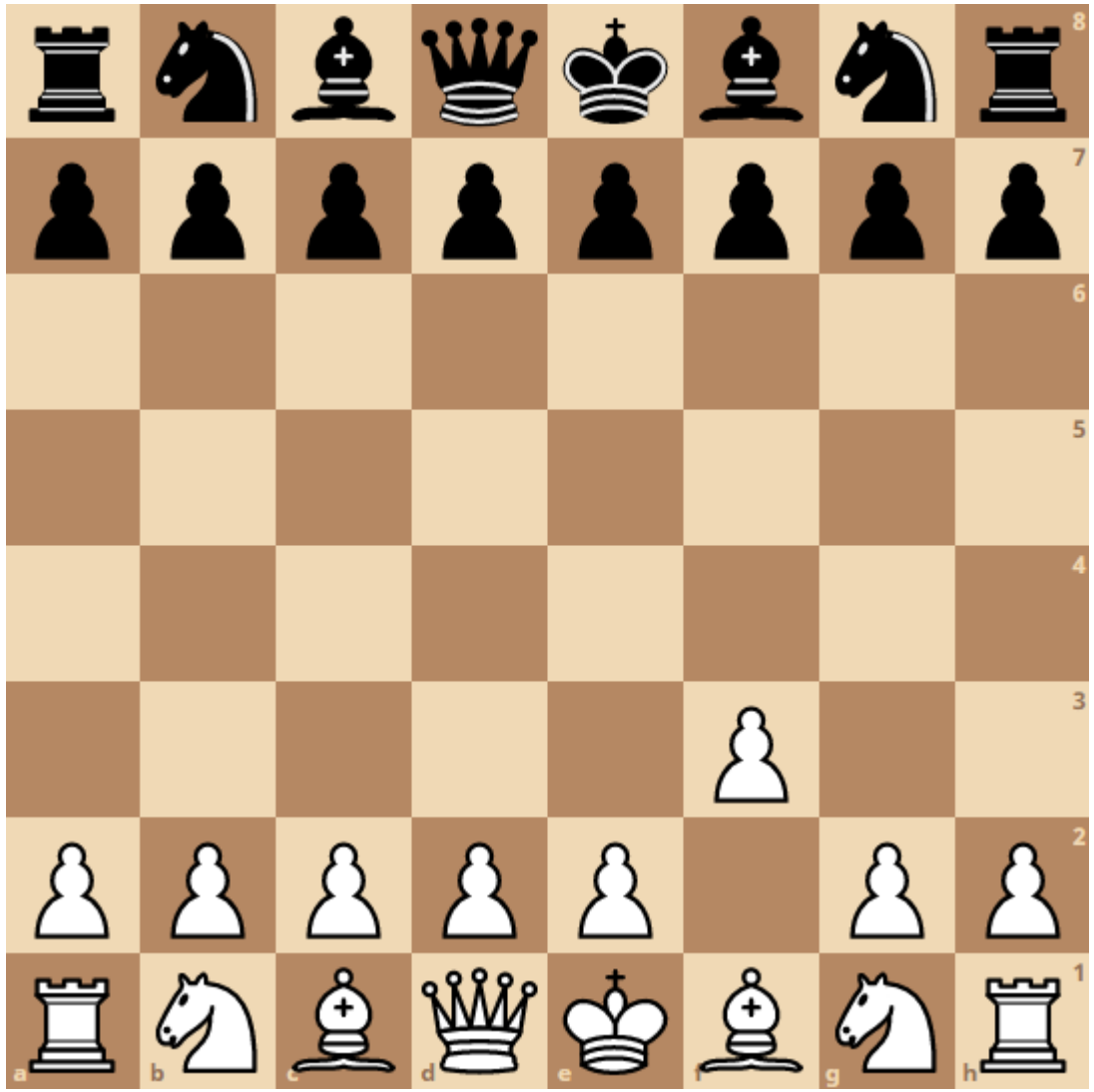
- Can we guess/predict threats for our app?
- As in a chess game we need to predict the next move of the other player (hacker?).
- In software, it happens the same, we need to predict what are the possible threats that can affect our systems.
- Threat modeling is like brainstorming of possible moves a hacker can do against our system.
- There are different methodologies to follow: STRIDE, CAPEC, Attack Tree, OWASP TOP 10 …

Owasp top 10

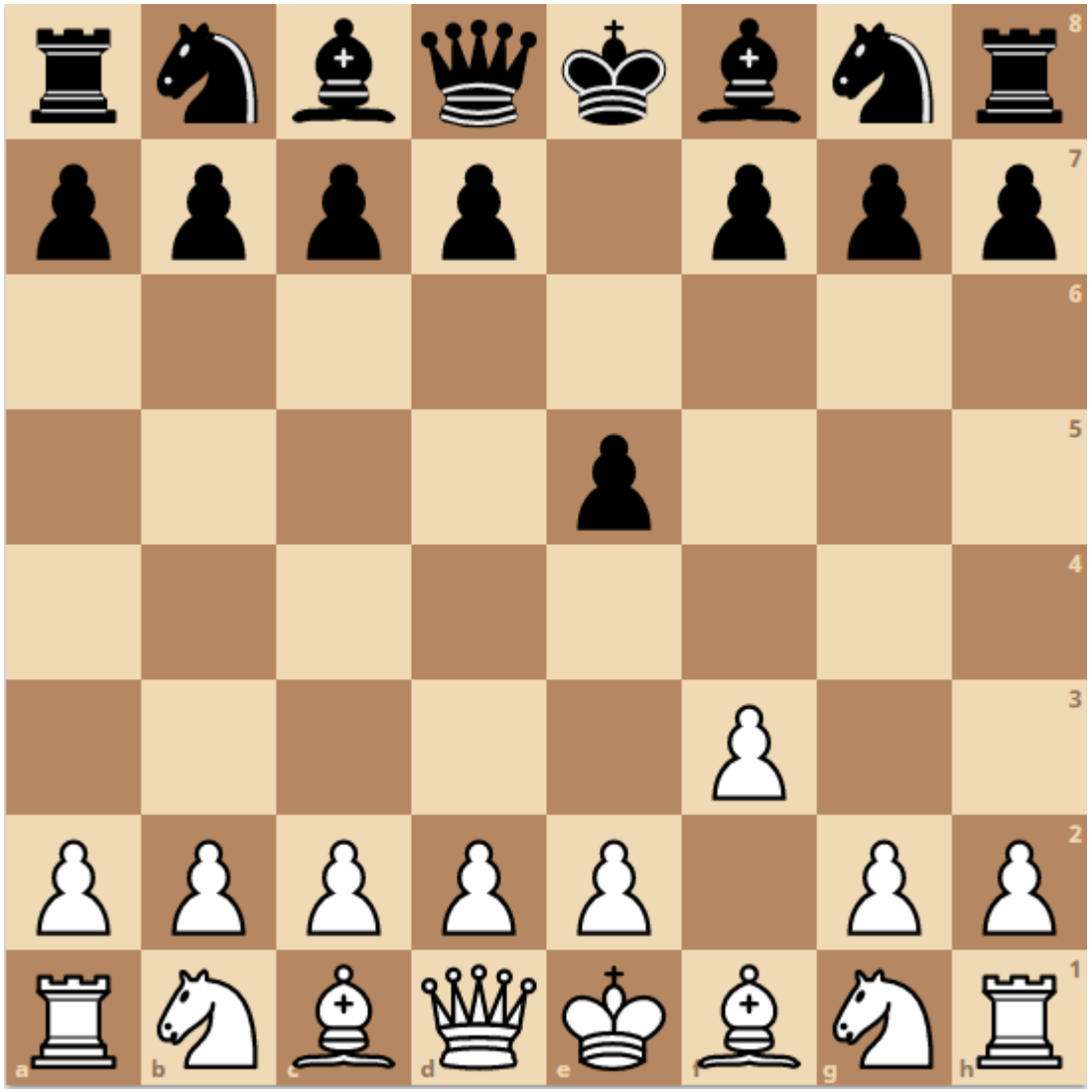

1. Injection
2. Broken Authentication & SessionManagement
3. Cross-Site Scripting (XSS)
4. Broken Access Control
5. Security Misconfiguration
6. Sensitive Data Exposure
7. Insufficient Attack Protection
8. Cross-Site Request Forgery
9. Using Components with Known Vulnerabilities
10. Undeprotected APIs
OWASP TOP 10 2017

SQL Injection

SQL Injection



SQL Injection



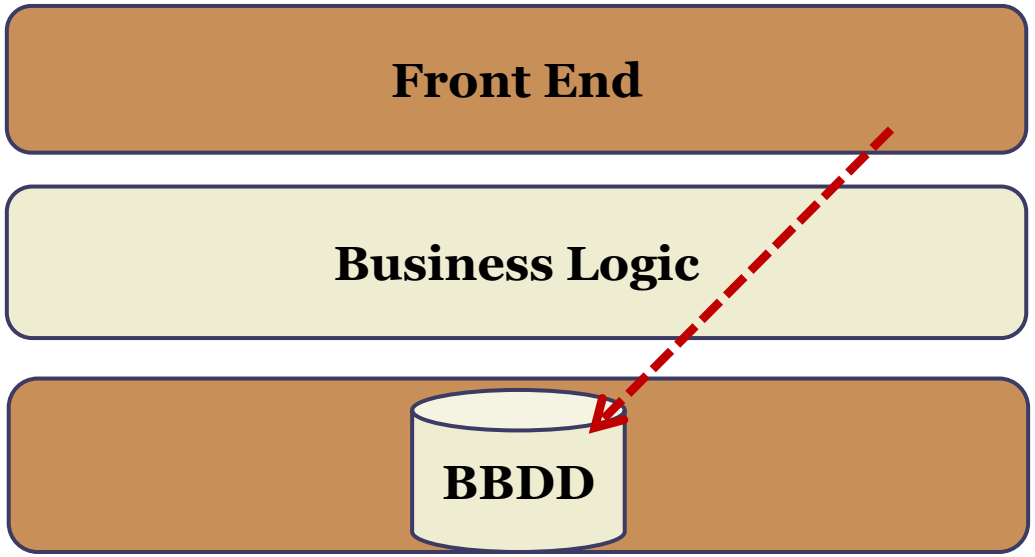
SQL Injection

SQL Injection

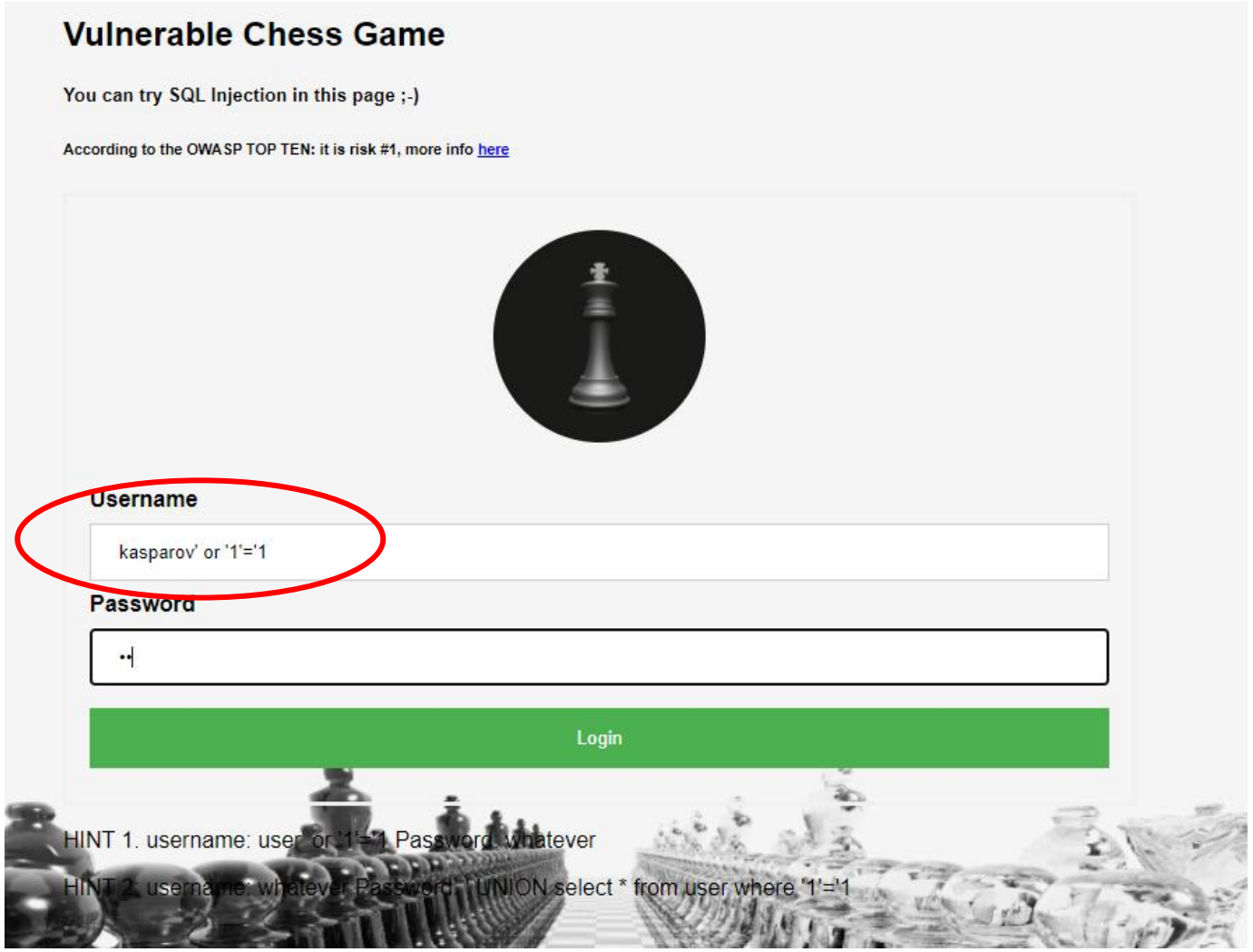
SQL Injection

SQL Injection

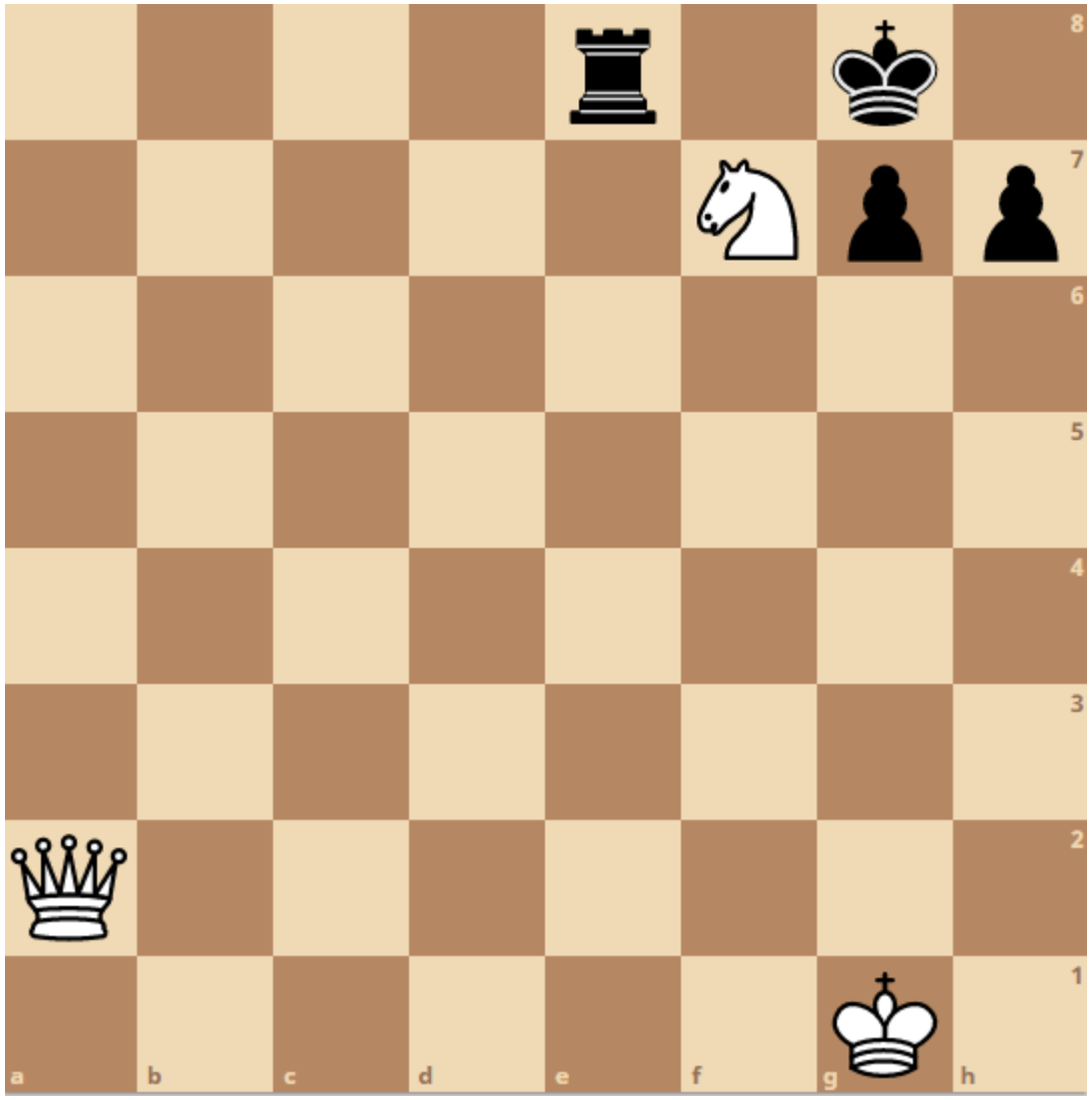
If we don't control all the access paths from our enemy lines to our systems. Then it can be the way to allow threats and then damage in our kingdom.



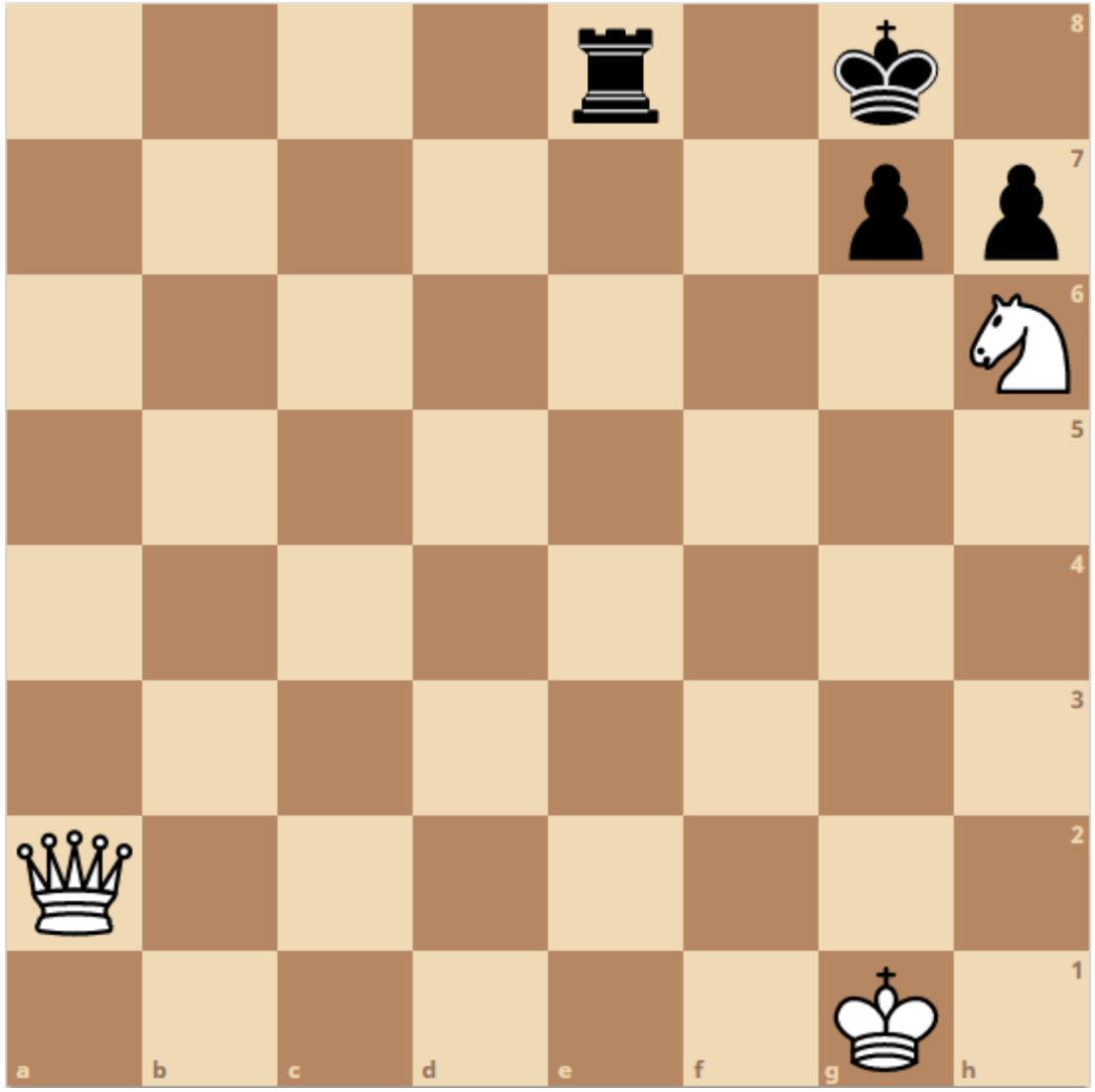
SQL Injection



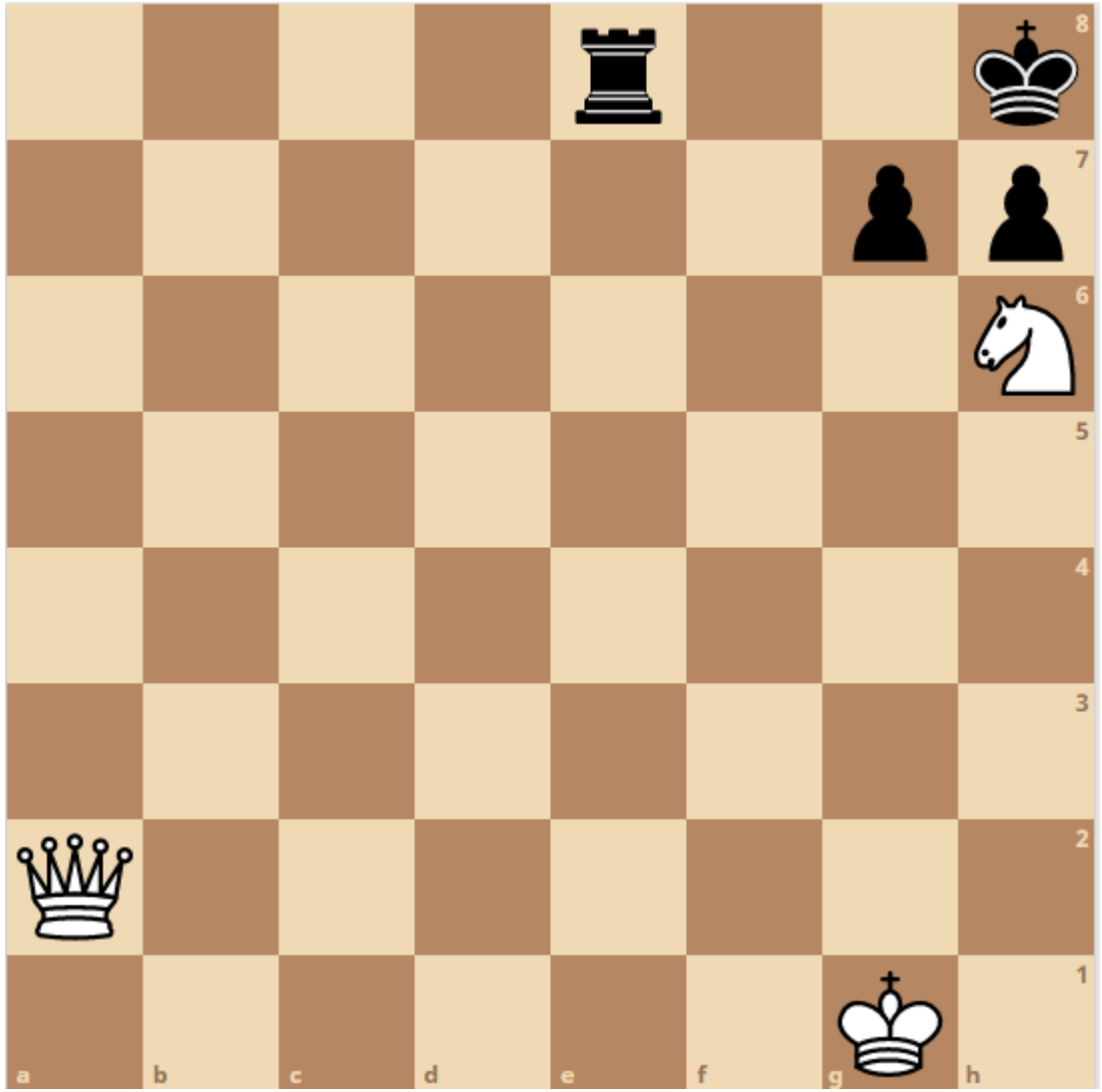
Cross site scripting



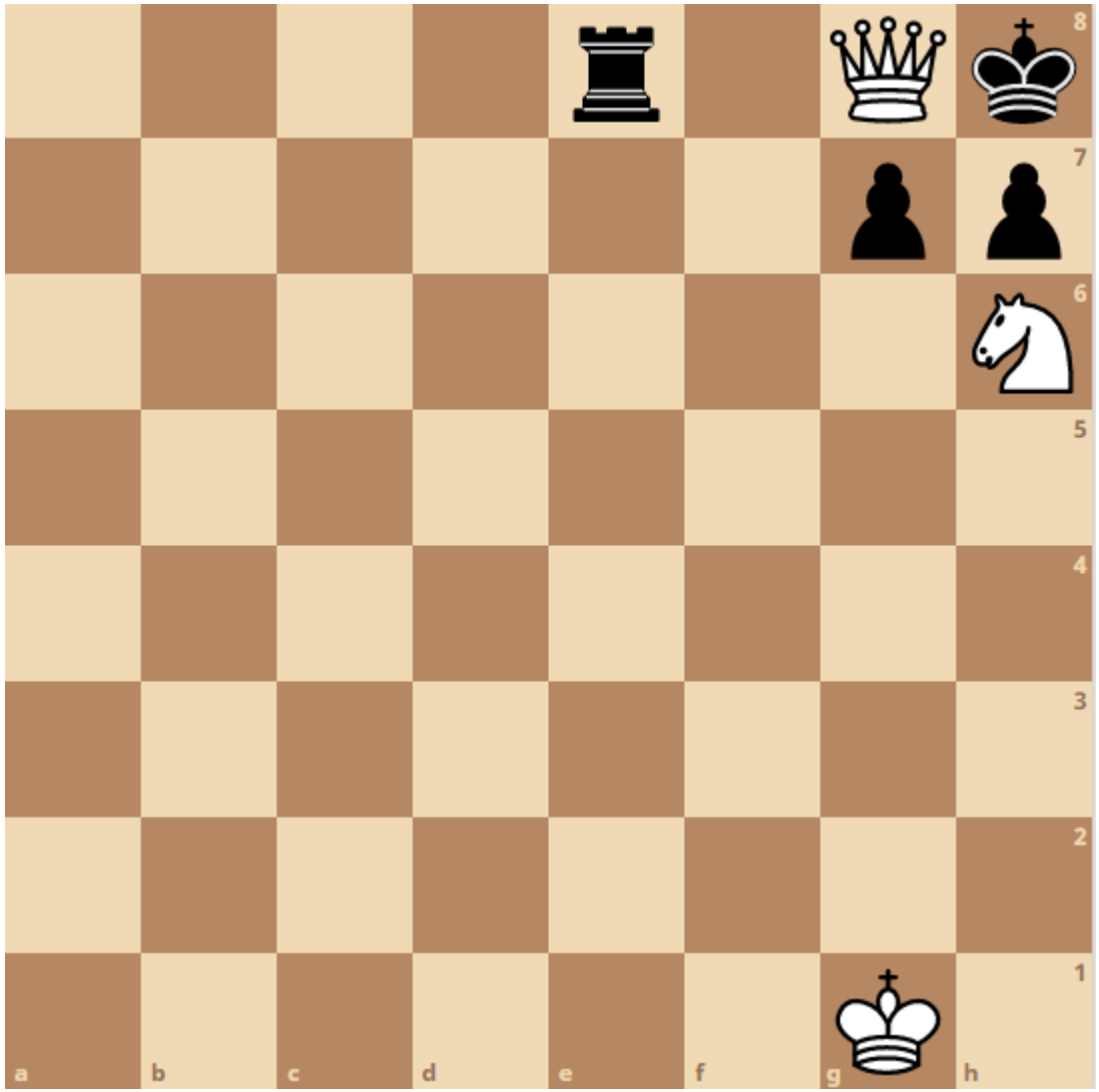
Cross site scripting



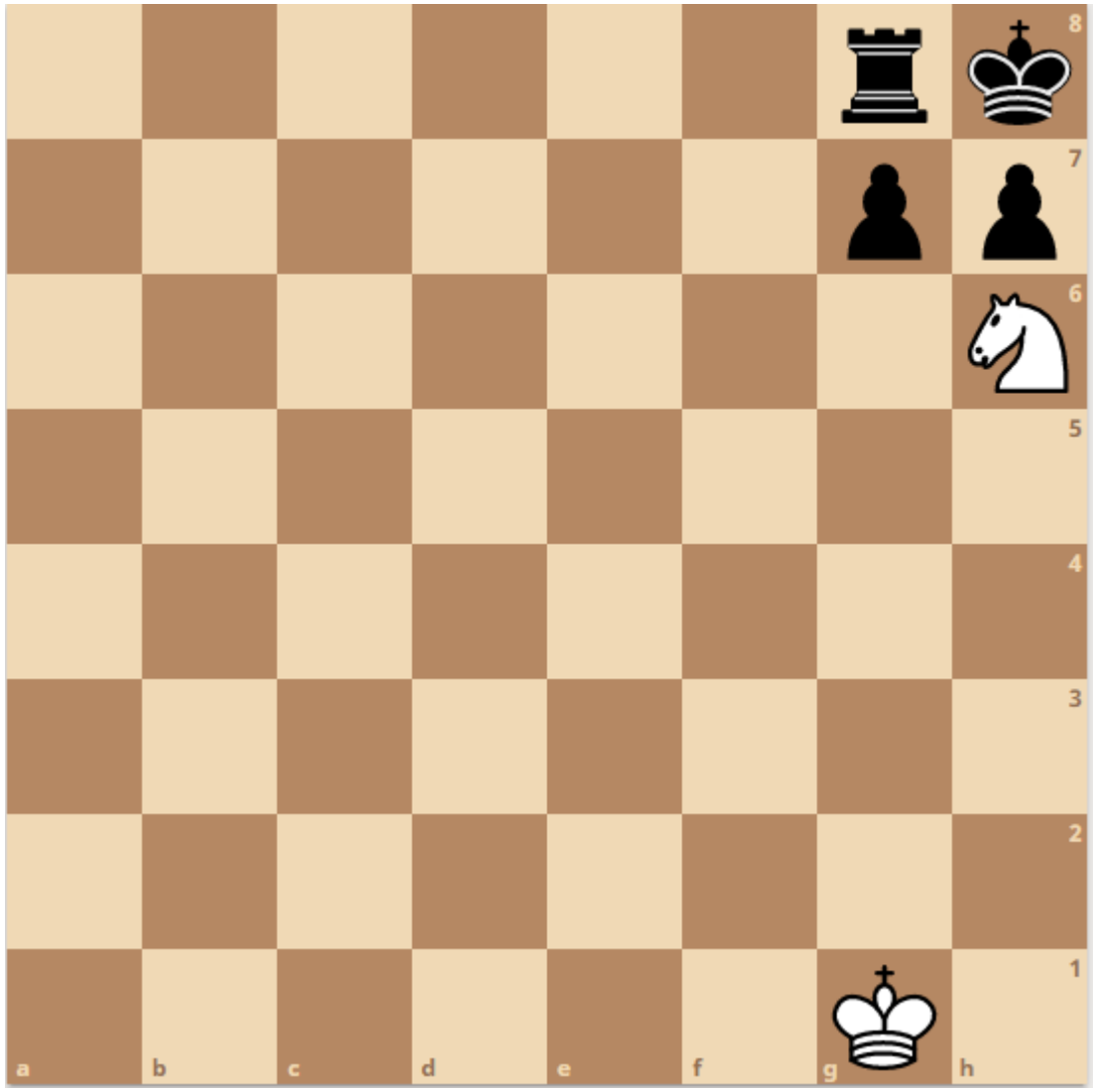
Cross site scripting



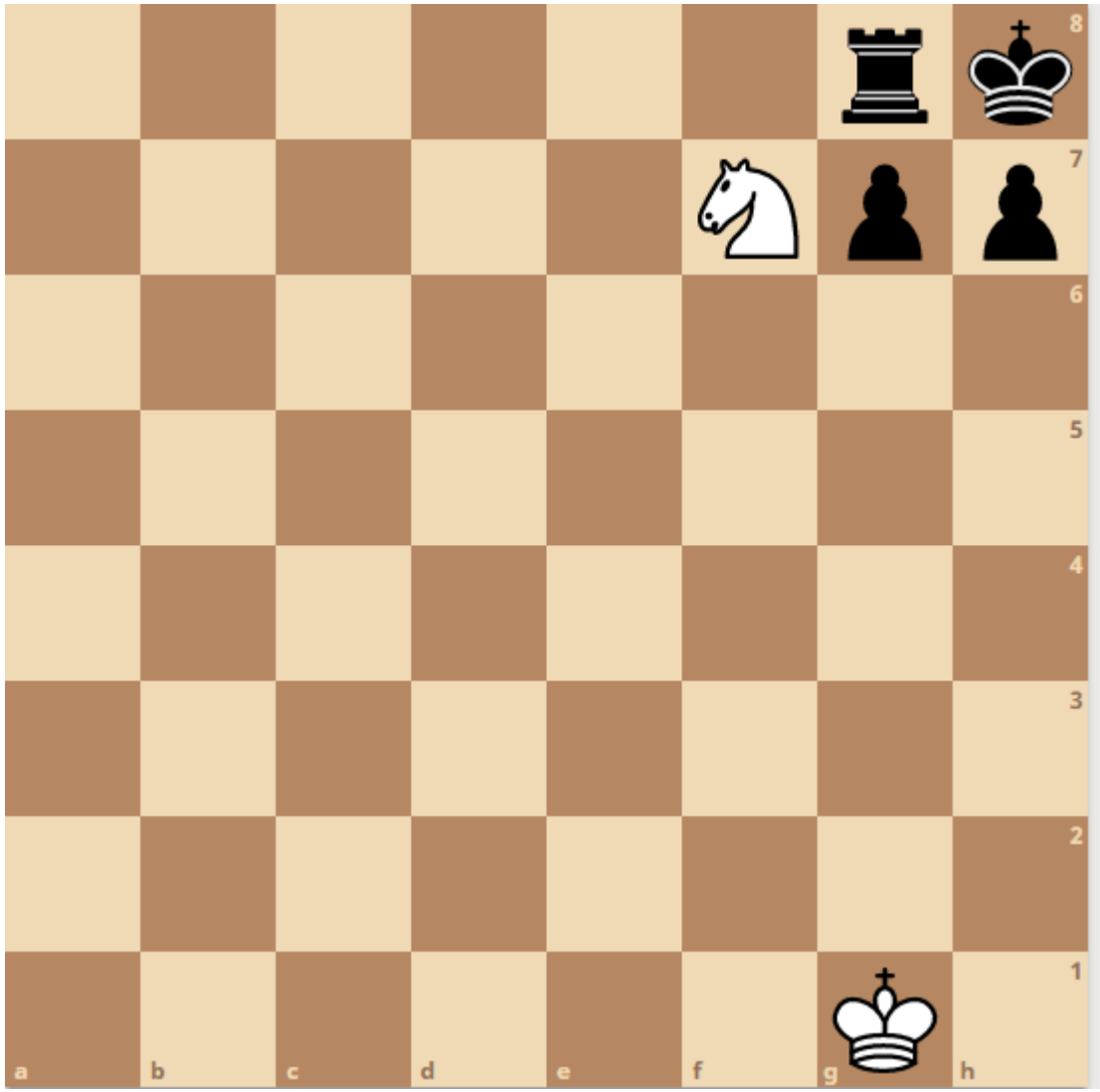
Cross site scripting



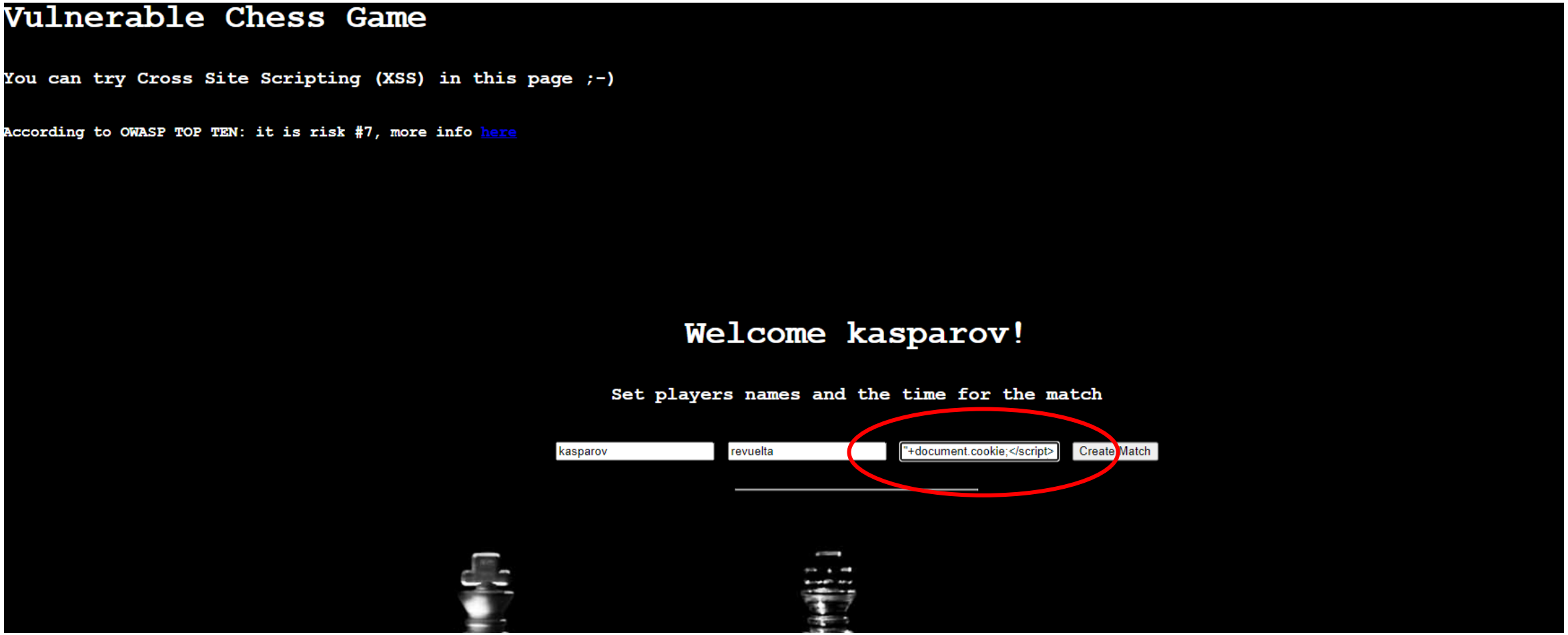
Cross site scripting



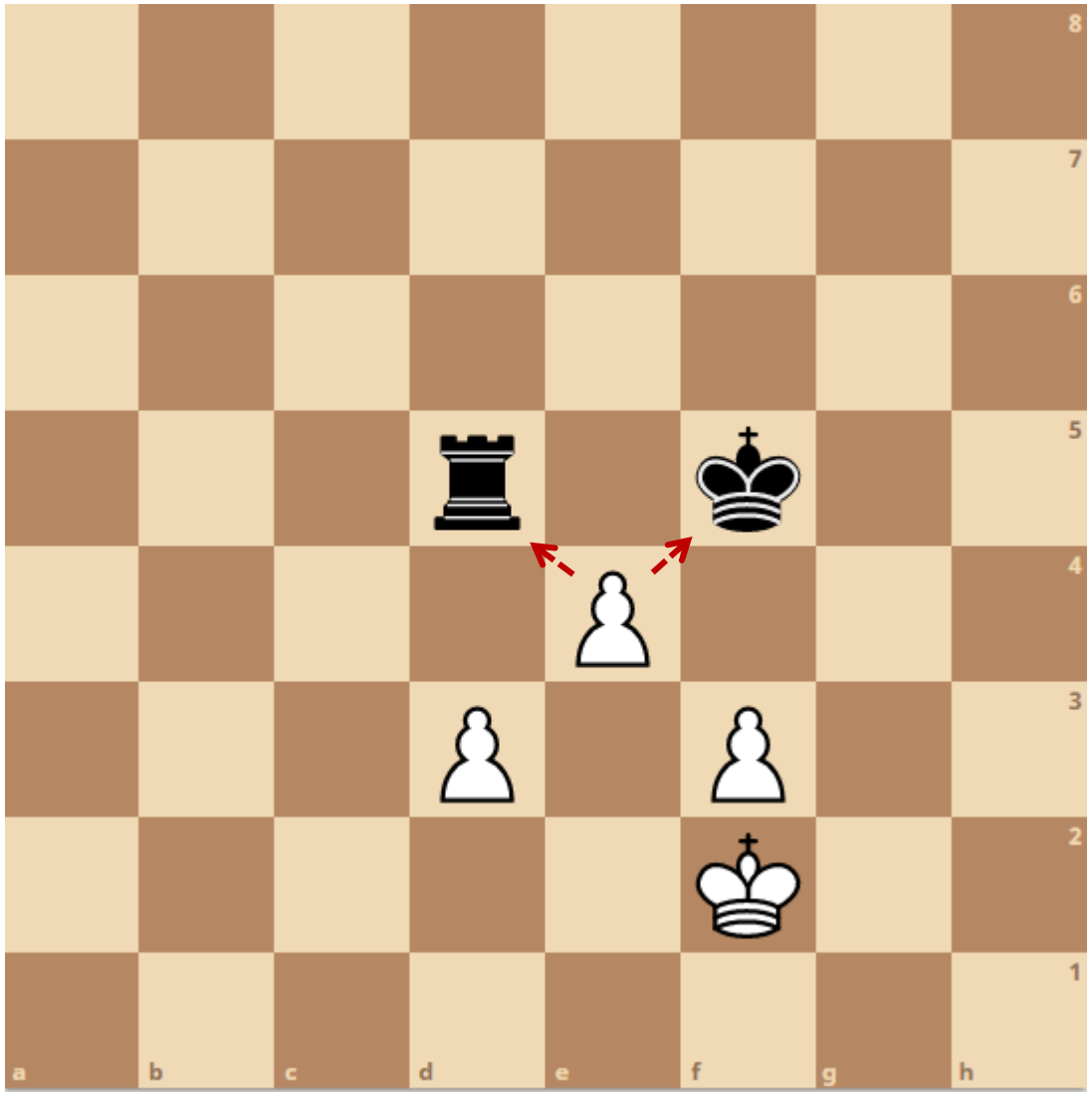
Cross site scripting



Cross site scripting



HTTP Response Splitting

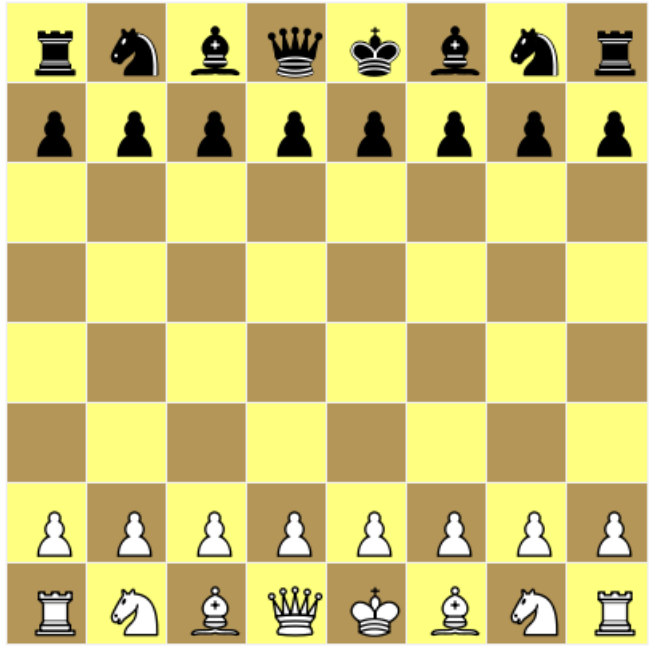


HTTP Response Splitting

Vulnerable Chess Game

Player: kasparov

Time: 10



Match:

Username
kasparov

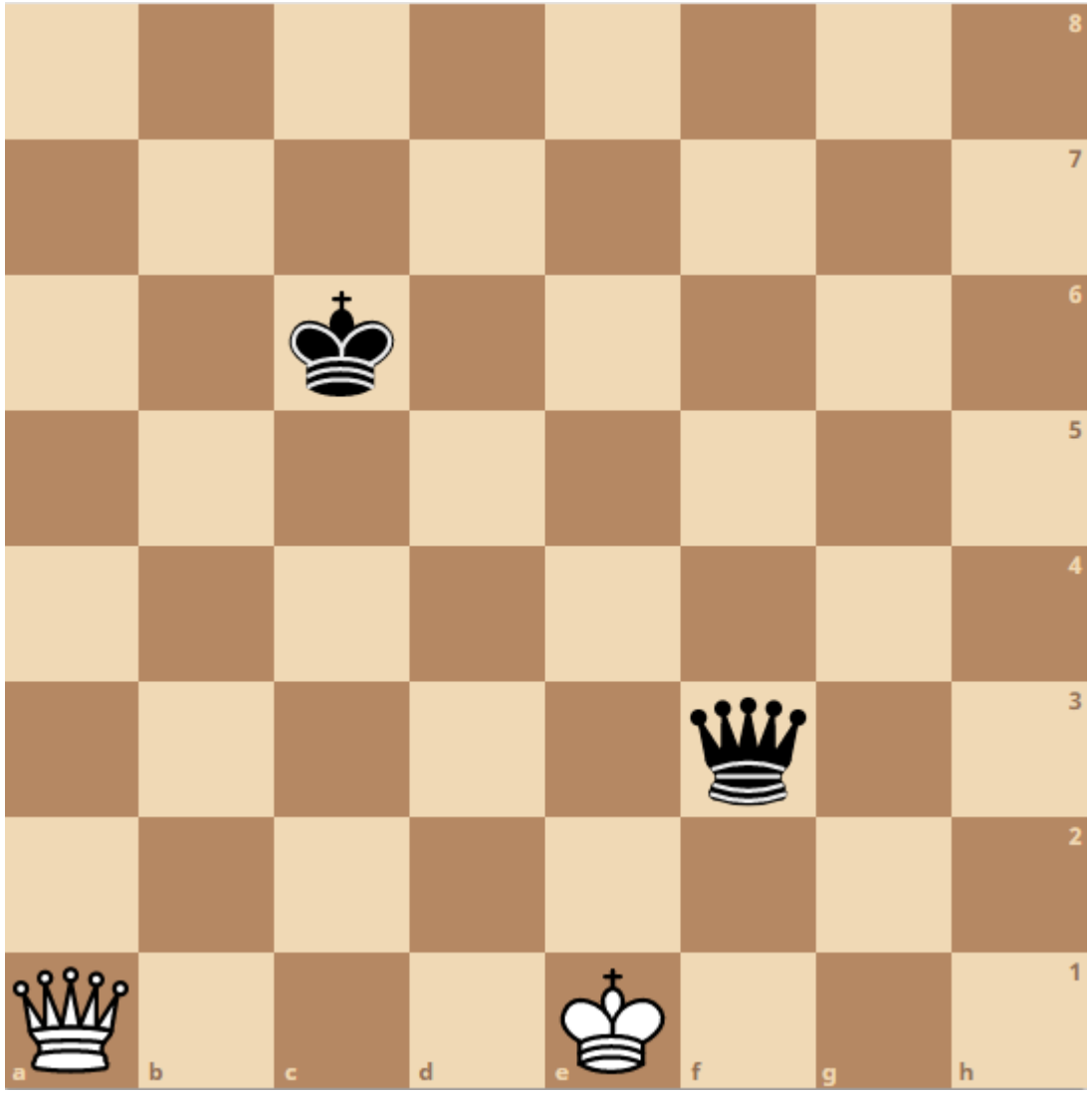
Play New Load Save

Player2: revuelta

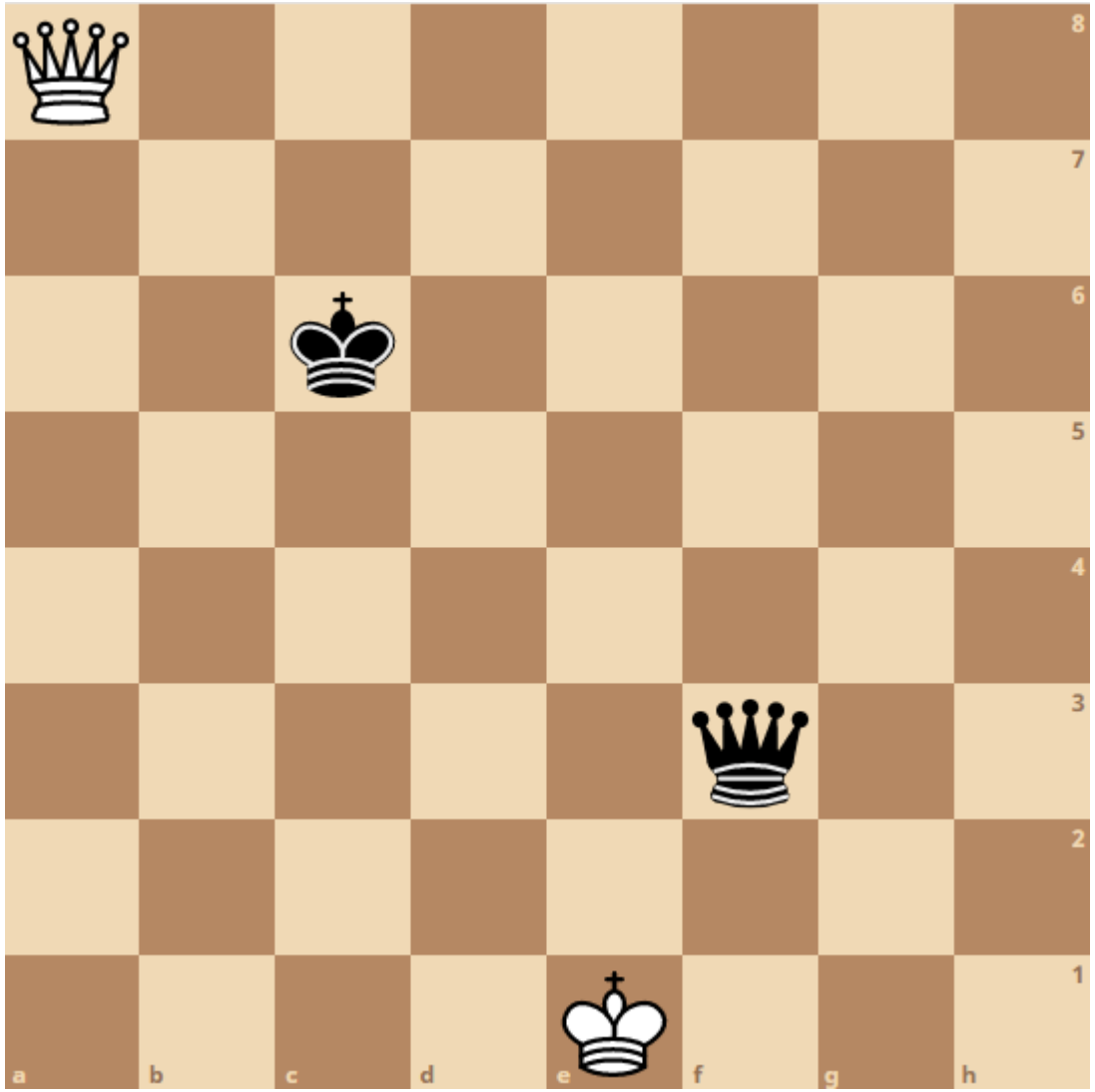
Time: 10

Click play button

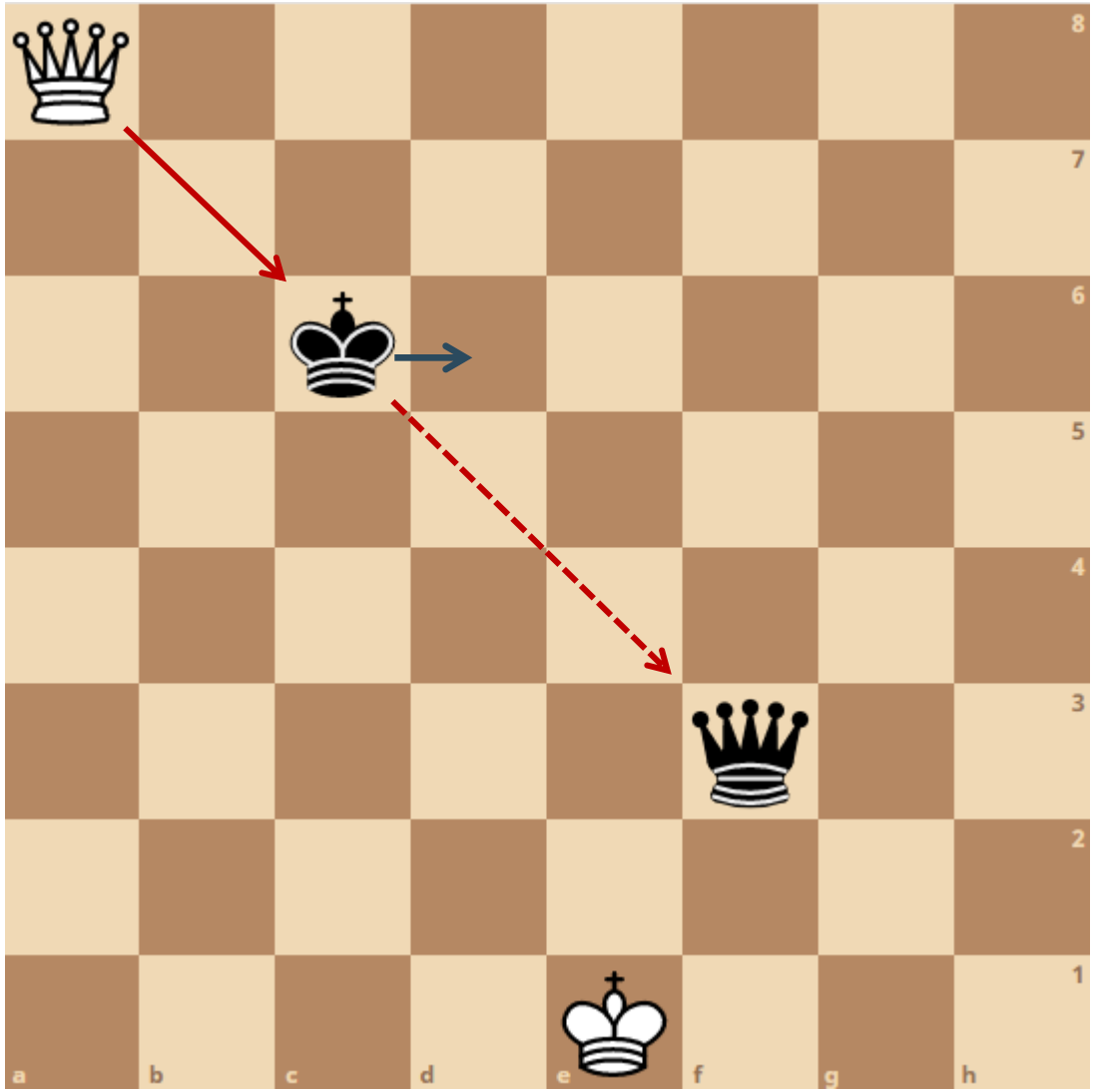
Path Traversal



Path Traversal



Path Traversal



Path Traversal (Demo)

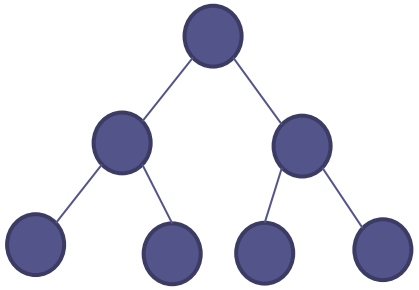
Vulnerable Chess Game

You can try Path Traversal in this page ;-)

According to OWASP TOP TEN: it is a serious risk, more info here

Deep code to win the match

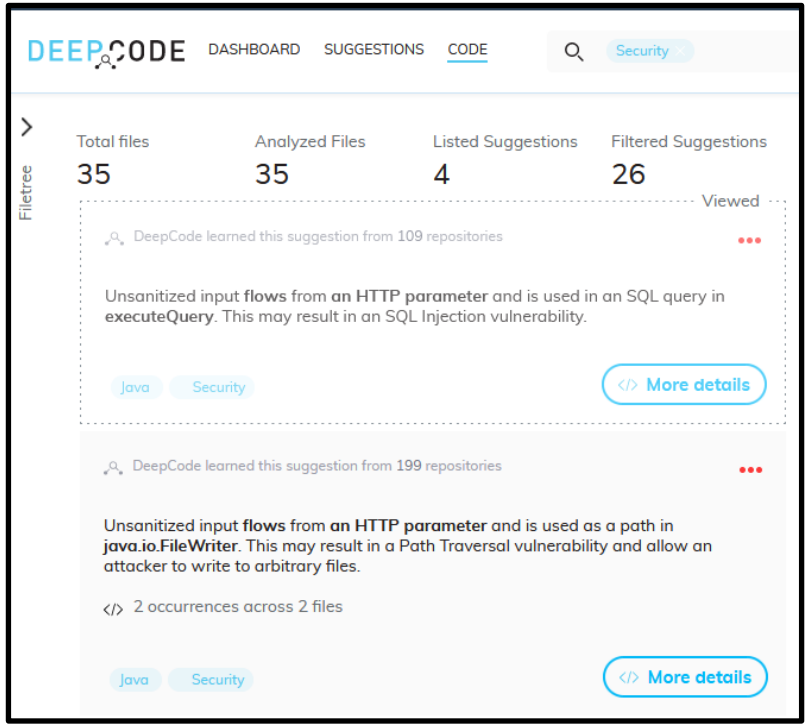
- **Deep blue** was the first machine to win a world chess champion as Kasparov (Nueva York 1997)
- We need **<u>intelligence machine</u>** to win the match against vulnerabilities.



Deep code to win the match

```
C:\projects\source\Chess>deepcode analyze --path . --result-text --with-linters --log-file=~/.deepcode.log
```

```
[Online analysis results]:
https://www.deepcode.ai/app/gh/sebastianrevuelta/DEEPCODE_PRIVATE_BUNDLE/1a71de5f48897a3b57aa8cfa8a67321a74910ebbe0517bc00c3d1fca27982666/_/%2F/code/?
```



Deep code to win the match

DEEPCODE DASHBOARD SUGGESTIONS CODE Security FILTERS

src/main/java/core/sebas/servlets/ Authentication.java

```
45          String query = "select * from user where username='" + username + "' and password = '" + password + "'";
46          Connection conn = null;
47          Statement stmt = null;
48          Properties prop = new Properties();
49
50          InputStream inputStream = getClass().getClassLoader().getResourceAsStream("config.properties");
51          if (inputStream != null) {
52              prop.load(inputStream);
53          }
54          else {
55              throw new FileNotFoundException("property file config.properties not found in the classpath");
56          }
57
58          String userDB = prop.getProperty("userDB");
59          String pwdDB = prop.getProperty("pwdDB");
60          String hostDB = prop.getProperty("hostDB");
61          String portDB = prop.getProperty("portDB");
62
63          String connectionChain = "jdbc:mysql://" + hostDB + ":" + portDB + "/chess?serverTimezone=UTC";
64          try {
65              conn = DriverManager.getConnection(connectionChain, userDB, pwdDB);
66              stmt = conn.createStatement();
67
68              //SQL injection. No validation of malicious input
69              ResultSet rs = stmt.executeQuery(query);
70
```

Next Suggestion 1 / 4

Unsanitized input **flows** [:41, :45] from **an HTTP parameter** [:41] and is used in an SQL query in **executeQuery** [:69] . This may result in an SQL Injection vulnerability.

More info

Line 69

Security Maintenance Bug Query Database Table

This issue was fixed by 109 projects. Here are 3 example fixes.

linzeqipku/SnowGraph ◂ Example 1/3 ▸

```
public class OutGoingRelationServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
```

Deep code to win the match

DEEPCODE DASHBOARD SUGGESTIONS CODE Security FILTERS

src/main/java/core/sebas/servlets/ SaveGame.java

```
26
27      private final static Logger log = Logger.getLogger(SaveGame.class);
28      private Match match;
29      public Match getMatch() { return match; }
30      public void setMatch(Match match) { this.match = match; }
31
32      /**
33       *
34       */
35      private static final long serialVersionUID = 1L;
36
37
38      @Override
39      protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
40
41          HttpSession session = req.getSession();
42          resp.setContentType("text/html;charset=UTF-8");
43          PrintWriter out = resp.getWriter();
44
45          Match match = (Match) session.getAttribute("match");
46
47          String file = req.getParameter("file");
48
49          File dir = new File("/matches/");
50          if (!dir.exists()) {
51              dir.mkdir();
52          }
53          File f = new File(dir + "/" + file);
54
55          BufferedWriter bw = new BufferedWriter(new FileWriter(f, true));
56          bw.write("Match");
57          bw.newLine();
58          bw.write("player1: " + match.getPlayer1());
59          bw.write("player2: " + match.getPlayer2());
60          bw.newLine();
61          bw.write("time: " + match.getTimer() + " minutes");
```

Previous | Next Suggestion 2 / 4

Unsanitized input **flows** [:47, :53] from **an HTTP parameter** [:47] and is used as a path in **java.io.FileWriter** [:55] . This may result in a Path Traversal vulnerability and allow an attacker to write to arbitrary files.

More info

Line 55

We found this issue 2 times in **2** files.

SaveGame.java

Security Maintenance Bug File Zip Download

This issue was fixed by 199 projects. Here are 3 example fixes.

sakaiproject/sakai ◂ Example 1/3 ▸

```
 * get file
 */
public void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
  if (!securityService.isSuperUser()){
}
```

Thanks ++

sebastianrevuelta@gmail.com
https://www.linkedin.com/in/sebasrevuelta/