Aula TP - 21/Mai/2018

Cada grupo deve colocar a resposta às perguntas dos seguintes exercícios na área do seu grupo no Github até ao final do dia 28/Mai/2018. Por cada dia de atraso será descontado 0,15 valores à nota desse trabalho.

Note que estes exercícios devem ser feitos na máquina virtual disponibilizada. Caso já tenha a versão da máquina virtual utilizada nas últimas aulas, não precisa de fazer download da nova versão.

Instruções de *update* da máquina virtual, para quem já tem a máquina virtual utilizada nas últimas aulas:

Nesta aula vamos utilizar o **WebGoat** a partir de uma imagem Docker. O **WebGoat** é uma aplicação Web deliberadamente insegura mantida pelo OWASP, cujo objetivo é ser um complemento às aulas de segurança de aplicações Web, pelo que contém falhas aplicacionais comuns.

Pode instalar o Docker na máquina virtual ou diretamente no sistema operativo do seu computador.

Se quiser **instalar o docker no seu computador** siga as instruções em https://store.docker.com/search?type=edition&offering=community relativas ao seu sistema operativo.

Se quiser **instalar o docker na máquina virtual**, siga as seguintes instruções:

1. `sudo apt-get update`
2. `sudo apt-get install apt-transport-https ca-certificates curl gnupg2 software-properties-common`
3. `curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -`
4. `sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs) stable"`
5. `sudo apt-get update`
6. `sudo apt-get install docker-ce`

Nota: Sempre que nos exercícios abaixo lhe for pedido para utilizar o **WebGoat**, deve executar os seguintes passos:

1. Garantir que já instalou o docker no seu computador ou na máquina virtual.
2. Obter a versão 7.1 do WebGoat, através do comando `sudo docker pull webgoat/webgoat-7.1`
3. Arrancar o WebGoat com o comando `sudo docker run -p 8080:8080 -t webgoat/webgoat-7.1`.

   Note que o WebGoat demora 30 - 50 segundos a arrancar, tendo arrancado quando lhe aparecer no ecran uma linha que contenha a seguinte informação `INFO: Starting ProtocolHandler ["http-bio-8080"]`
4. No Firefox da máquina virtual aceder a http://localhost:8080/WebGoat.

No final, para parar o docker e WebGoat, deverá efectuar os seguintes comandos:

- `sudo docker ps` para obter o Container ID
- `sudo docker kill <Container ID>` em que `<Container ID>` é o valor indicado no comando anterior para

esse campo.

Exercícios

1. *Injection*

Experiência 1.1

Aceda a https://free.codebashing.com/courses/java e siga o exemplo de *SQL Injection*.

Experiência 1.2

Aceda a https://free.codebashing.com/courses/java e siga o exemplo de *XXE Injection*.

Pergunta 1.1 - *String SQL Injection*

No WebGoat tente resolver o exercício *Injection Flaws -> String SQL Injection*:

1. O objetivo é utilizar *SQL Injection* que resulte em todos os cartões de crédito serem apresentados. Experimente com alguns nomes e veja como se comporta.
2. Execute o ataque de *SQL Injection* - experimente utilizar uma tautologia.

Em todos os exercícios tente primeiro resolver sem ajuda. Se chegar a um ponto em que não consegue avançar, utilize "Show Hints" para ajuda. Apenas deve utilizar "Show Solution" em situações extremas.

Pergunta 1.2 - *Numeric SQL Injection*

No WebGoat tente resolver o exercício *Injection Flaws -> Numeric SQL Injection*:

1. O objetivo é utilizar *SQL Injection* que resulte em todos os dados meteorológicos serem apresentados. Experimente com alguns locais e veja como se comporta.
2. Execute o ataque de *SQL Injection* - experimente utilizar uma tautologia. O problema é que aparentemente não temos maneira de alterar a query …
3. Talvez se possa utilizar as ferramentas de desenvolvimento do Firefox (canto superior direito do browser e selecione o botão "Developer", seguido da ferramenta "Inspector"). Mas como é que modificamos a informação que é enviada quando clicamos no botão "Go!"?
4. Talvez possa começar por procurar "Columbia" no código. A seguir altere o HTML para executar a tautologia.

Em todos os exercícios tente primeiro resolver sem ajuda. Se chegar a um ponto em que não consegue avançar, utilize "Show Hints" para ajuda. Apenas deve utilizar "Show Solution" em situações extremas.

Pergunta 1.3 - *Database Backdoors*

No WebGoat tente resolver o exercício *Injection Flaws -> Database Backdoors*, um passo de cada vez.

- Passo 1

1. O objetivo é utilizar *SQL Injection* qpara executar mais do que um comando SQL, para uma conta com o ID 101. Experimente como é que a aplicação se comporta com este ID.
2. Tente explorar a vulnerabilidade para alterar o salário para um valor mais elevado.

Relembre-se que o SQL UPDATE tem o seguinte formato:

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

- Passo 2

3. O objetivo é relativamente simples. Efetue o ataque com o comando SQL indicado para alterar o salário para um valor mais elevado.

Em todos os exercícios tente primeiro resolver sem ajuda. Se chegar a um ponto em que não consegue avançar, utilize "Show Hints" para ajuda. Apenas deve utilizar "Show Solution" em situações extremas.

2. XSS

Experiência 2.1

Aceda a https://free.codebashing.com/courses/java e siga o exemplo de *Persistent (Stored) XSS*.

Pergunta 2.1 - *Reflected XSS*

No WebGoat tente resolver o exercício *Cross-Site Scripting (XSS) -> Reflected XSS Attacks*:

1. O objetivo é utilizar o formulário de compras para refletir o input do utilizador. Como é que se pode fazer?
2. Este formulário é muito similar ao que pode encontrar em diversos sites. Tem uma lista de produtos, preços e quantidades. É possível mudar a quantidade e ver o preço alterado. No final, o utilizador tem que fornecer o cartão de crédito e o CCV para pagar os bens selecionados.
3. Por tentativa e erro, experimente combinação de letras e dígitos nos vários campos de input para ver que tipo de validações são feitas pelo servidor. Se encontrar algum campo com potencial para Reflected XSS, tente a seguinte script: `<script> alert("SSA!!!")</script>`

Em todos os exercícios tente primeiro resolver sem ajuda. Se chegar a um ponto em que não consegue avançar, utilize "Show Hints" para ajuda. Apenas deve utilizar "Show Solution" em situações extremas.

Experiência 2.2 - *Stored XSS*

No WebGoat tente resolver o exercício *Cross-Site Scripting (XSS) -> Stored XSS Attacks*:

1. O objetivo é deixar uma mensagem no seu site favorito que desencadeie uma ação quando outro utilizador for ler a sua mensagem. Como é que se pode fazer?
2. Faça alguns testes no formulário de mensagens para verificar o tipo de validação e filtragem do input que é efetuado.
3. Se achar que tem potencial para Stored XSS, tente a seguinte script: `<script language="javascript" type="text/javascript">alert(document.cookie);</script>` que vai

apresentar os cookies num Popup.
4. Verifique se funcionou.
5. Como é que poderia um atacante beneficiar desta vulnerabilidade?

Em todos os exercícios tente primeiro resolver sem ajuda. Se chegar a um ponto em que não consegue avançar, utilize "Show Hints" para ajuda. Apenas deve utilizar "Show Solution" em situações extremas.

3. Quebra na Autenticação

Pergunta 3.1 - *Forgot Password*

No WebGoat tente resolver o exercício *Authentication flaws -> Forgot Password*.

Os mecanismos de autenticação têm usualmente a funcionalidade de permitirem recuperar a password através da resposta a uma questão pessoal. Contudo, muitas vezes é simples de adivinhar a resposta à questão!

1. A aplicação WebGoat permite que os utilizadores recuperem a password se conseguirem responder a uma questão sobre a sua cor favorita. Tente o utilizador "webgoat" e cor "red" e veja o que acontece.
2. Agora tente atacar outro utilizador. Que utilizadores existem normalmente num sistema? De todos esses utilizadores, qual era a conta que gostaria de comprometer? Tente essa.
3. Como é que pode obter a password dessa conta? Talvez possa tentar acertar na cor por força bruta …

Em todos os exercícios tente primeiro resolver sem ajuda. Se chegar a um ponto em que não consegue avançar, utilize "Show Hints" para ajuda. Apenas deve utilizar "Show Solution" em situações extremas.

Projeto de desenvolvimento de software

Os alunos deverão utilizar o resto desta aula TP para continuarem o projeto de desenvolvimento de software.

O projeto 1 (Leilões online) será efetuado, em conjunto, pelos grupos 1, 6, 10, 11, 12.

O projeto 2 (Gestor de passwords com base em QrCodes) será efetuado, em conjunto, pelos grupos 2, 3, 4, 5, 7, 8, 9.

- Projeto 1 – Leilões online
  - Leilões online, com entrega de propostas em "carta fechada";
  - Pode ser uma extensão para software open source de leilões online.
- Projeto 2 – Gestor de passwords com base em QrCodes
  - Gestor de passwords, em que com base em QrCode apresentado pelo site, o telemóvel lê o QrCode e envia o user + password para desbloquear o acesso;
  - Pode ser uma extensão para software open source de gestão de passwords.

Nesta primeira fase, os dois grupos de projeto devem definir em traços gerais o projeto e as suas funcionalidades, e pensarem de que modo serão utilizado as técnicas criptográficas no projeto.

Como output desta fase, deverão ter um primeiro draft de:

- definição do projeto e suas funcionalidades,
- etapas e fluxos de comunicação / mensagens, podendo utilizar como exemplo o formato visto no segundo

exemplo do voto eletrónico, na aula teórica. Esta componente deve conter um diagrama e uma parte textual de explicação do diagrama,
- identificar os passos efetuados para a concepção e desenvolvimento do projeto, de forma a seguir os princípios de "*privacy by design*" e "*data minimization*" do RGPD (Regulamento Geral de Proteção de Dados);
- identificar de que modo o software garante os direitos do titular dos dados, de acordo com o RGPD.

Estes pontos deverão fazer parte do relatório final do projeto.

SAMM (*Software Assurance Maturity Model*)

Nesta fase do projeto é-lhe pedido para, utilizando o ciclo de melhoria contínua do SAMM,

1. Avaliar a maturidade das práticas de segurança utilizadas no desenvolvimento de software deste projeto (Fase *Assess*);
2. Estabelecer o objetivo para cada uma das 12 práticas de segurança (Fase *Set the Target*), i.e., o nível de maturidade pretendido;
3. Desenvolver o plano para atingir o nível de maturidade pretendido, em quatro fases (Fase *Define the Plan*).

Para isso deverá utilizar a Toolbox (ficheiro excel) fornecida na diretoria Aula9, onde também encontrará mais informação relativa ao SAMM.

Note que:

- Para a Fase *Assess* deverá preencher a *sheet* "*Interview*";
- Para a Fase *Set the Target*, o grupo deverá discutir qual o *score* objetivo para cada uma das 12 práticas de segurança, que sirva de guia para atuar sobre as atividades mais importantes. Pode partir do princípio que a sua organização é uma startup que vai efectuar desenvolvimento de software na área de i) sistemas web seguros online (caso do projeto 1) e ii) sistemas de identificação eletrónica (caso do projeto 2). Se necessitar de outros pressupostos, indique-os na justificação à decisão tomada;
- Para a Fase *Define the Plan* deverá preencher a *sheet* "*Roadmap*", supondo que cada uma das fases tem 3 meses de duração. Tenha em conta o esforço necessário e a eventual dependência entre atividades em cada uma das fases.

A Toolbox (ficheiro excel) deverá fazer parte do relatório final do projeto, fornecendo o ficheiro excel como anexo ao relatório. Adicionalmente, no documento do relatório deverá incorporar um anexo em que indique a decisão da Fase *Set the Target* e a justifique.

Note que não há respostas certas nem erradas.