UNIVERSIDADE FEDERAL DE SANTA CATARINA
INFORMÁTICA E ESTATÍSTICA

Lucas Vinícius da Rosa

COMPARAÇÃO E ESPECIALIZAÇÃO DE METODOLOGIAS DE SEGURANÇA EM APLICAÇÕES WEB PARA O CONTEXTO DE SISTEMAS DE ICP

Florianópolis(SC)

2012

Lucas Vinícius da Rosa

COMPARAÇÃO E ESPECIALIZAÇÃO DE METODOLOGIAS DE SEGURANÇA EM APLICAÇÕES WEB PARA O CONTEXTO DE SISTEMAS DE ICP

TCC submetido ao Curso de Ciências da Computação para a obtenção do Grau de Bacharel em Ciências da Computação.
Orientador: Jean E. Martina, Ph. D
Coorientador: Prof. Ricardo Felipe Custódio, Dr.

Florianópolis(SC)

2012

Catalogação na fonte elaborada pela biblioteca da
Universidade Federal de Santa Catarina

A ficha catalográfica é confeccionada pela Biblioteca Central.

Tamanho: 7cm x 12 cm

Fonte: Times New Roman 9,5

Maiores informações em:
http://www.bu.ufsc.br/design/Catalogacao.html

Lucas Vinícius da Rosa

COMPARAÇÃO E ESPECIALIZAÇÃO DE METODOLOGIAS DE SEGURANÇA EM APLICAÇÕES WEB PARA O CONTEXTO DE SISTEMAS DE ICP

Este TCC foi julgado aprovado para a obtenção do Título de "Bacharel em Ciências da Computação", e aprovado em sua forma final pelo Curso de Ciências da Computação.

Florianópolis(SC), 23 de Novembro 2012.

Prof. Vitório Bruno Mazzola, Dr.
Coordenador

Banca Examinadora:

Jean E. Martina, Ph. D
Orientador

Prof. Ricardo Felipe Custódio, Dr.
Coorientador

Anderson Luiz Silvério

À minha família; ao LabSEC; e aos meus amigos.

AGRADECIMENTOS

À todos, direta ou indiretamente, envolvidos no desenvolvimento do presente Trabalho de Conclusão de Curso.

É preciso ter um caos dentro de si para dar à luz uma estrela cintilante.

Friedrich Nietzsche

RESUMO

Aplicações Web estão cada vez mais presentes na Internet. Seja fornecendo soluções específicas ao ambiente corporativo, ou no modelo de oferta de serviços ao usuário final. Dessa forma, faz-se necessário avaliar sua segurança de forma abrangente, contudo detalhada. Essa avaliação tem como suporte metodologias de testes de intrusão, que fundamentam os procedimentos de testes. São exemplos de metodologias: OSSTMM, ISSAF, OWASP e WASC. Dada a pluralidade dos modelos de testes disponíveis, será efetuada uma análise conceitual das abordagens existentes. Ademais, este trabalho procurará eleger uma, ou mais, dentre as metodologias abordadas, de modo a testar intrusivamente uma aplicação web que envolve Infra-estrutura de Chaves Públicas. O resultado da execução do teste de intrusão será registrado em um relatório, para posterior análise e mitigação dos riscos levantados.

Palavras-chave: Metodologia, Segurança, Aplicação Web, OSSTMM, ISSAF, OWASP, WASC, ICP, SGCI.

ABSTRACT

Web applications are increasingly present in the Internet. Either by providing enterprise environment with specific solutions or by offering services to the end user. Thus it turns out necessary to assess security comprehensively but in detailed form. This assessment is supported by penetration testing methodologies which justify the testing procedure. Examples of such methodologies are: OSSTMM, ISSAF, OWASP and WASC. Considering the plurality of available testing models a conceptual analysis on existing approaches will be conducted. Moreover this work will seek to choose one, or more, among the debated methodologies for intrusively test a web application involving Public Key Infrastructure. The penetration test result will be registered on a report for future analysis and mitigation of the raised risks.

Keywords: Methodology, Security, Web Application, OSSTMM, ISSAF, OWASP, WASC, PKI, SGCI.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Assumimos, hoje, posto de observação privilegiado frente ao avanço tecnológico computacional, particularmente no ecossistema de aplicações (ou softwares). Os antigos e estáticos programas de computador, antes meros conjuntos de instruções executados localmente ou em estações de execução dedicadas (mainframes), mostram-se atualmente transformados e inseridos em uma categoria especializada: as Aplicações Web. Com a popularização da Internet, ocorrida a partir da segunda metade de 1990, houve, gradualmente, a migração do ambiente corporativo e seus ativos para a conjuntura virtual. Consequentemente, conforme os anos passavam e a rede mundial de computadores evoluía, as empresas curvavam-se, mais e mais, aos benefícios da digitalização de seus processos operacionais, assim como o armazenamento de dados sensíveis em localizações remotas, porém acessíveis através da Web.

Desse modo, as aplicações Web tornaram-se onipresentes na Internet, trazendo, juntamente com a ubiquidade dos dados e a melhoria da eficiência no acesso, questões frágeis no tocante à sua segurança. Uma vez substituídos, ou adicionados, novos vetores de ataque – aqueles relacionados às aplicações Web modernas –, observou-se uma migração do interesse do atacante, que passou a voltar sua atenção às falhas em sistemas baseados na Web.

Como transições de natureza tecnológica não ocorrem, comumente, em caráter imediato, porém em progressões paulatinas, assim o foi com as aplicações Web e suas ameaças inerentes. Ainda hoje, é observada uma significativa incidência de falhas antigas, datadas de décadas atrás, como os famigerados ataques de injeção de SQL, por exemplo.

Ao longo deste trabalho, será possível constatar que, embora as tecnologias convirjam, dispersem-se, enfim, evoluam, ainda há um fator determinante no estabelecimento da segurança em torno de um sistema: o agente humano. A inabilidade do desenvolvedor de software, seja por negligência ou falta de conhecimento, em resguardar aspectos de codificação e de modelagem seguros, acarreta na aparição de brechas de segurança, normalmente referidas como vulnerabilidades.

O aumento da oferta de aplicações Web, portanto, alavancou a quantidade de ataques voltados a essa área. Em virtude da preocupação que adorna o tema, criaram-se diversas iniciativas e/ou projetos cujo objetivo é, essencialmente, compor metodologias de endereçamento das questões de segurança de uma aplicação Web. Neste trabalho, são atendidas as seguintes metodologias: OSSTMM, ISSAF, OWASP e WASC.

Por certo, em se tratando de aplicações Web e sua diversidade, não há metodologia de teste definitiva. Posto como um guia de orientação ao testa-

dor, um subconjunto de diretrizes de uma metodologia pode ser mais conveniente ao escopo testado que outro subconjunto, pertencente a outra abordagem. Em face disso, e levando-se em consideração os requisitos do software SGCI (Software de Gerenciamento do Ciclo de Vida de Certificados da ICPEdu), serão destacados, de uma metodologia ou a partir de uma composição desta com outra, os testes mais aderentes ao domínio em que se insere a aplicação.

O teste de intrusão ratifica na prática, dessa forma, a descoberta e eventual exploração das fraquezas previstas pela metodologia escolhida; permite diagnosticar regiões de acesso não autorizado, vazamentos de dados, violação do esquema de privilégios, etc. Tal diagnóstico é registrado, conforme os testes são executados, em um relatório. Quando constatadas vulnerabilidades, serão fornecidas suas estratégias de mitigação.

1.1 OBJETIVOS

Realizar-se-á um estudo individual e comparado das metodologias de segurança em aplicações Web hoje disponíveis; nomeadamente: OSSTMM, ISSAF, OWASP e WASC. Uma vez compreendidos os pontos máximos e mínimos de cada metodologia, será eleita uma abordagem de teste de intrusão adequada ao âmbito de aplicações utilizadoras de ICPs.

1.1.1 Objetivos específicos

Aplicar-se-á um teste de intrusão, correspondente às metodologias adotadas, contra o sistema SGCI. Em seguida, um relatório de vulnerabilidades será confeccionado, enumerando as condições de ataque testadas, sejam elas confirmadas ou não. A partir do relatório, será estimada a superfície de ataque da aplicação testada, com base em métricas de segurança operacional.

1.2 JUSTIFICATIVA E MOTIVAÇÃO

Tradicionalmente, a indústria de segurança da informação ocupava-se do endereçamento de vulnerabilidades sobre a infra-estrutura de redes de computadores. No entanto, com a ascensão da Web 2.0, juntamente a onipresença das aplicações Web na Internet, o foco do atacante foi significativamente alterado; as falhas em aplicações Web consolidaram-se como alvo predileto do atacante moderno. Conclusivamente, segurança passou a ser um requisito relevante não somente às redes de computadores, mas ao nível de

aplicação (CROSS, 2006).

Entretanto, ataques em nível de aplicação, diferentemente daqueles voltados às redes de computadores – como *Distributed Denial of Service* (DDoS) e ameaças de vírus –, podem ser conduzidos por essencialmente qualquer usuário online (CROSS, 2006).

Aplicações Web, por sua natureza, costumam ser publicamente acessíveis. E assim também o são os servidores Web que as propulsionam, em virtude da necessária disponibilidade dos mesmos a usuários legítimos. Em consequência disso, há maiores chances de comprometimento dessa classe de sistemas de computador (GRAVES, 2010).

A entrada de um atacante em território virtual restrito, potencialmente, abre portas para outras regiões da infra-estrutura de rede subjacente à aplicação. A título de exemplo, e especialmente de alerta, uma simples não filtragem de um parâmetro cujo valor é definido pelo usuário pode, dependendo da magnitude da falha, representar a violação de toda uma rede de computadores.

Segundo o *Imperva's Web Application Attack Report*, atacantes têm expandido a escala de seus ataques. Em um intervalo de seis meses, de Julho a Dezembro de 2011, foi registrado, para 40 aplicações Web monitoradas, um aumento de 25000 para 38000 ataques por hora, ou 10 tentativas a cada segundo (IMPERVA, 2010). Um panorama estatístico deste cenário, em termos de volume de ataque mensal, pode ser visualizado na figura 1. Observe-se a predominância dos ataques de injeção, SQLi e XSS, quando comparados aos demais, assim como o ponto máximo de investidas durante o mês de Agosto.

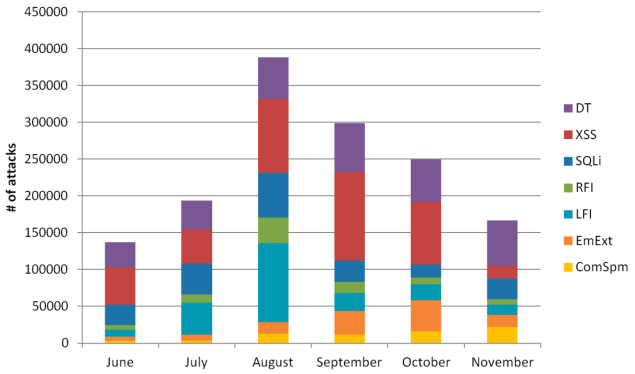

Figura 1: Volume de ataques mensal (IMPERVA, 2010)

Ainda conforme este mesmo relatório (IMPERVA, 2010), a exploração de falhas em apli-cações Web remete, majoritariamente, às cinco seguin-

tes classes de vulnerabilidades: *Remote File Inclusion* (RFI), *SQL Injection* (SQLi), *Local File Inclusion* (LFI), *Cross Site Scripting* (XSS) e *Directory Traversal* (DT); XSS e DT posicionam-se como as modalidades de ataque mais prevalentes.

Finalmente, diante das especificidades de um aplicação utilizadora de ICP, como é o caso do SGCI, incorre-se na necessidade de estudo das metodologias existentes. Seja pela adaptação dos procedimentos de teste à disposição, ou pela formulação de novas diretrizes de ataque como, por exemplo: tentativa de comprometimento da cadeia de certificação, pela personificação de uma entidade, ou agente, da infra-estrutura pelo atacante; corrupção ou ameaça ao diretório que armazena listas de certificados revogados, LCRs; forjar uma requisição de certificado em nome de um usuário, etc.

1.3 METODOLOGIA

Inicialmente, foram levantadas as metodologias atuais voltadas ao campo de segurança em aplicações Web. Após tal levantamento, selecionou-se um subconjunto destas para estudo aprofundado: OSSTMM, ISSAF, OWASP e WASC.

Uma vez individualmente estudadas, cada metodologia teve seus aspectos contrastados com os requisitos específicos encontrados em aplicações Web que utilizam ICPs. Tal análise contrastada é complementar às características básicas comumente encontradas em aplicações Web tradicionais, como gerenciamento de sessão do usuário, esquemas de autenticação e autorização, tratamento de dados oriundos do usuário, robustez de diretórios de armazenamento, etc.

O estudo individual, posteriomente estendido para uma análise coletiva das metodologias, fomentou um processo de eleição da abordagem mais adequada de teste de intrusão contra o sistema SGCI.

A realização do teste de intrusão culminou na composição de um relatório das vulnerabilidades testadas, classificando-as quanto a viabilidade de exploração e, quando confirmadas, oferecendo estratégias de mitigação correspondentes.

1.4 LIMITAÇÕES DO TRABALHO

Este trabalho não contempla todos os testes propostos pelas metodologias. Todavia, e oportunamente, apenas aqueles relacionados às características de aplicações Web e ao escopo de ICPs.

Stuttard, Dafydd Pinto e Marcus (STUTTARD DAFYDD, 2008) propõem-nos, no capítulo *"A Web Application Hacker's Methodology"*, uma metodologia própria para a condução de testes de intrusão. Por conseguinte, longe de exaurir todo o espectro de falhas de uma aplicação, ressalta-se o objetivo crucial da metodologia: fornecer garantia suficiente de que as regiões fundamentais da superfície de ataque da aplicação foram abrangidas, assim como encontrar tantas fraquezas quanto possíveis, considerando os recursos à disposição do testador.

A ideia imediatamente supracitada reflete a abrangência, e sua limitação de raio, do teste de intrusão neste trabalho documentado. Ou seja, o fato de um conjunto de vulnerabilidades ter sido descoberto não impede o aumento do tamanho deste conjunto.

Determinados ataques, como os de *Cross Site Scripting*, podem ser dirimidos por mecanismos intermediários de filtragem de dados, o que não significa, de forma alguma, a impossibilidade de evasão de tais medidas. A astúcia do atacante, conjuntamente ao seu nível de conhecimento, pode ser o ponto de minerva perante o comprometimento de uma aplicação.

1.5 ESTRUTURA DO TRABALHO

Divido em cinco partes, este trabalho desenvolve, inicialmente, no capítulo 2, os conceitos teóricos necessários à compreensão das metodologias expostas e da aplicação a ser testada. São tratados fundamentos de criptografia simétrica e assimétrica, infra-estrutura de chaves públicas, e aspectos do software SGCI.

No capitulo 3, desenrola-se sobre quatro metodologias de segurança em aplicações Web, sendo elas: OSSTMM, ISSAF, OWASP e WASC.

O capítulo 4, por sua vez, estipula uma metodologia adequada para a realização do teste de intrusão no SGCI. Seguidamente, descrevem-se as atividades práticas do teste de intrusão, cuja descrição segmenta-se em: delimitação do escopo, planejamento, execução, relatório de resultados e, finalmente, determinação da superfície de ataque da aplicação.

Relativamente às considerações finais, o capítulo 5 encerra o trabalho, trazendo as motivações que fundamentaram a condução deste; assim como um apanhado sobre os procedimentos realizados e às conclusões deles derivadas.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 CRIPTOGRAFIA

Criptografia é o processo de transformação de um mensagem, de modo a deixá-la ilegível. Ou, ainda, de acordo com Schneier (SCHNEIER, 1996), é a arte e a ciência de manter mensagens seguras.

Sua origem exata é incerta. Porém, tem-se registros de sua utilização pelo egípicios, em 1900 a.c.. Ao longo do tempo, as técnicas de transformação evoluíram, de acordo com as necessidades de ocultação de mensagens. Atualmente, tornou-se uma ciência avançada, amparada substancialmente por recursos computacionais.

Uma mensagem é um texto plano, ou em claro. Schneier (SCHNEIER, 1996) nos informa que o processo de dissimular uma mensagem a fim de ocultar seu conteúdo original chama-se cifragem. De maneira inversa, o processo de retornar o texto cifrado para plano denomina-se decifragem.

Além da confidencialidade – ocultação da mensagem para todos, senão para os destinatários pretendidos –, restam-nos três propriedades essenciais à criptografia (SCHNEIER, 1996):

- **Autenticação**: deve ser possível para o receptor de uma mensagem determinar sua origem; um intruso não deve ser capaz de personificar outro indivíduo.

- **Integridade**: o receptor de uma mensagem deve ser capaz de verificar que ela não foi modifica em trânsito; deve ser inviável a um intruso substituir uma mensagem legítima por outra falsa.

- **Não-repúdio (ou irretratibilidade)**: um emissor não deve ser capaz de negar o fato dele próprio ter enviado a mensagem.

Ademais, são pertinentes à criptografia os conceitos de algoritmo criptográfico e chave. O primeiro, também chamado de cifra, é a função matemáti-ca usada para cifragem e decifragem. Chave, por conseguinte, designa uma valor numérico, computacionalmente representado por uma cadeia de bits.

Algoritmos e chaves criptográficas são itens inerentes a dois paradigmas de cifragem – criptografia simétrica e assimétrica.

2.1.1 Criptografia simétrica

Na criptografia simétrica, conforme Polk (POLK, 2001), tanto emissor quan-to receptor fazem uso do mesmo valor de chave. Em consequência disso, sistemas de chave simétrica são, às vezes, referenciados como sistemas de chave de segredo compartilhado.

Quando Alice deseja enviar uma mensagem privada para Bob, ela seleciona um algoritmo de cifragem e uma chave de conhecimento de ambos. Alice cifra a mensagem em claro usando o algoritmo e chave selecionados, gerando o texto cifrado, que é encaminhado para Bob. Este, então, usa o algoritmo de decifragem e chave para recuperar o texto original a partir do texto cifrado.

2.1.2 Criptografia assimétrica

Criptografia assimétrica, ou criptografia de chave pública, utiliza duas chaves distintas. Uma delas mantida privada, e outra que pode ser tornada pública. As duas chaves, como afirmado por Polk (POLK, 2001), são complementares. Todavia, o valor da chave privada não pode ser obtido a partir da chave pública.

Adicionalmente, chaves públicas podem ser distribuídas abertamente, dado que não necessitam serem mantidas em sigilo.

Quando Alice deseja enviar uma mensagem para Bob, valendo-se de criptografia assimétrica, ela obtém a chave pública deste e a utiliza na cifragem. Bob, em seguida, decifra o conteúdo cifrado usando sua chave privada.

O método acima exposto ilustra uma aplicação de criptografia assimétrica voltada à confidencialidade. Ou seja, remete ao transporte seguro de um mensagem de Alice para Bob.

Entretanto, como destacado por Polk (POLK, 2001), criptografia de chave pública não é normalmente usada para cifragem de dados do usuário. Algoritmos de chave pública costumam ser dispendiosos, exigindo significativamente mais poder computacional que algoritmos de cifragem simétrica.

Dessa maneira, sobrepondo a utilidade supracitada, a criptografia assimétrica encontra sua preponderante aplicação na assinatura digital e nas Infra-estruturas de Chaves Públicas, as ICPs.

2.2 INFRA-ESTRUTURA DE CHAVES PÚBLICAS

ICPs constituem-se de uma variedade de componentes. Cada um deles é projetado para realizar tarefas consistentes e bem definidas. O arranjo dos componentes é hierárquico, estabelecendo uma cadeia de confiança, cujo ponto mais alto é denominado âncora de confiança.

Conforme Polk (POLK, 2001), os seguintes componentes funcionais concretizam a ICP:

- A autoridade certificadora
- A autoridade de registro
- O repositório
- O arquivo

Para uma melhor compreensão das estruturas funcionais e objetivos de uma ICP, definiremos alguns conceitos primordiais a ela.

2.2.1 Certificados de chave pública

Para Polk (POLK, 2001), um certificado de chave pública é um objeto puramente digital. Contém campos referentes ao indivíduo, ou entidade, a que pertence e uma chave pública a este respectiva. Assim sendo, um certificado agrega informações de atributos do indivíduo, período de validade e, não obstante, nome e assinatura do seu emissor, dentre outros dados.

Normalmente, em soluções utilizadoras de criptografia assimétrica, são empregados certificados codificados no formato X.509, cuja especificação é definida pela ITU-T (*ITU Telecommunication Standardization Sector*).

2.2.2 Lista de certificados revogados

Certificados de chave pública possuem um ciclo de vida e, dado isso, alterações em seu estado precisam estar disponíveis aos demais componentes da ICP. Uma possível alteração de estado no certificado é a sua revogação.

Um certificado pode ser revogado por uma série de razões, destacando-se: comprometimento da cadeia de certificação, alteração nas informações do indivíduo, perda ou furto da chave privada correspondente à chave pública do certificado, etc.

A ferramenta básica de ICP para distribuição de informações de estado do certificado é a LCR – Lista de Certificados Revogados. Segundo Polk (POLK, 2001), a LCR contém uma lista de números seriais correspondentes a certificados não expirados que, no entanto, não são mais confiáveis. Sendo um objeto digital, sua distribuição ocorre por meio eletrônico. Além disso, sua autenticidade é garantida pela assinatura digital de seu emissor.

2.2.3 Autoridade Certificadora

A autoridade certificadora, AC, é o bloco de construção base da ICP. É visualizada como uma coleção de *hardware*, *software*, e pessoas que a operam. Polk (POLK, 2001) aponta-nos dois atributos pelos quais a AC é identificada: seu nome e sua chave pública.

Uma AC realiza quatro operações fundamentais; são elas:

- Emite certificados (isto é, cria e os assina);
- Mantém informação de estado dos certificados e emite LCRs;
- Publica seus certificados atuais, e não expirados, para que usuários obtenham a informação que necessitam para implementar serviços de segurança;
- Mantém arquivos de informação de estado, no que se refere a certificados expirados e revogados por ela emitidos.

2.2.4 Autoridade de Registro

A autoridade de registro, AR, tem por responsabilidade verificar os conteúdos de um dado certificado para a AC. Esses conteúdos podem refletir informações apresentadas pela entidade que está requisitando o certificado, como documentos de identificação oficiais (RG, CPF, Carteira Nacional de Habilitação, etc).

Como a AC, a AR também é uma coleção de *hardware*, *software* e pessoas que a operam. Sua operação é realizada por um ou mais indivíduos, denominados agentes de registro – AGRs.

AC e AR vinculam-se uma a outra através de relações de confiança, que atestam a confiança mútua entre as partes. Operações com base nas chaves pública e privada de cada entidade permitem sedimentar a confiança entre uma AR e uma AC, pelas vias da assinatura digital.

2.3 SGC

Um Sistema de Gestão de Certificados (SGC), conforme definido por Kapidzic e Davidson (KAPIDZIC, 1995), é um sistema em rede voltado à geração, distribuição, armazenamento e verificação de certificados para uso em uma variedade de aplicações de segurança. Para a realização dessas operações, por conseguinte, atuam entre si entidades cooperantes, como Autoridades Certificadoras (ACs) e Autoridades de Registro (ARs). Desse processo de interação decorrem as relações de confiança entre as entidades componentes de uma ICP.

De modo mais judicioso, as operações de um SGC são: criação e operação de ACs e ARs, emissão e revogação de certificados digitais, e emissão de LCRs.

A série de recomendações CCITT 1988 X.500 determina que a criação de certificados deve ser realizada por ACs. E o formato dos certificados comumente estabelecido para SGCs apoia-se no padrão X.509.

A criação e verificação de certificados, funções que pertencem ao escopo de um SGC, são discutidas detalhadamente no documento RFC1422 (KENT, 1993).

2.3.1 SGCI

O acrônimo SGCI significa Sistema de Gerenciamento de Certificados Digitais ICPEDU (Infra-estrutura de Chaves Públicas para Ensino e Pesquisa). Ele contempla as características e as operações tradicionais de SGCs, comentadas na seção 2.3.

De acordo com a página oficial do projeto (LABSEC, 2012), as principais características do sistema SGCI são:

- Criação e gerenciamento de ACs e ARs
- Utilização de entidades online e offline
- Utilização de ACs de resposta automática
- Utilização de ACs com e sem AR
- Suporte a HSM (*Hardware Security Module*)
- Suporte aos algoritmos ECDSA e RSA
- Suporte aos algoritmos de hash SHA-1 e SHA-2

- Suporte a um subconjunto do CMP (RFC4210 e RFC4211)
- Instalação simples (via apt em sistemas Debian)
- Código internacionalizado

Módulo de Segurança Criptográfico, MSC, é um dispositivo voltado ao processamento criptográfico e gestão de chaves criptográficas que mantém conformidade com normas adequadas de construção de hardware. Seu projeto e implementação deve, em face da sensibilidade das operações que realiza, levar em consideração os mais diversos ataques que ameaçam os processos criptográficos oferecidos (RNP, 2004).

Em sua versão mais recente, v2.0.0, o SGCI oferece suporte a dispostivos HSM, como mesmo demonstra a lista de principais funções acima exibida.

O SGCI divide-se em cinco módulos, definidamente (SILVéRIO, 2011):

- **Criador**: responsável pela criação de entidades e usuários, assim como pelas configurações da aplicação. Neste módulo, ACs, ARs e usuários podem ser cadastrados ou removidos, além da síntese de vínculos entre usuários e entidades;
- **Administrador de AC**: responde pelo gerenciamento de ACs. Neste módulo, são realizadas as operações de cadastro e remoção de operadores, definição de modelos de certificados, gestão de relacionamentos de confiança e determinação do período de emissão de LCRs;
- **Administrador de AR**: responde pelo gerenciamento de ARs. Neste módulo, tomam corpo as funções de cadastro e remoção de operadores e, também, o gerenciamento de relacionamentos de confiança;
- **Operador de AC**: responsável pela operação de ACs. Neste módulo, é possível aprovar ou rejeitar requisições de certificado e de revogação. Caso uma requisição seja aprovada, há a possibilidade de emissão do certificado a ela respectivo. Além disso, o Operador de AC emite LCRs.
- **Operador de AR**: reponsável pela operação de ARs. Neste módulo, o operador pode aprovar ou rejeitar requisições de certificado e de revogação. No entanto, diferentemente do Operador de AC, ele não emite LCRs.

Diversas versões foram lançadas pelo projeto SGCI, desenvolvido no LabSEC – Laboratório de Segurança em Computação, UFSC. A primeira, v1.3.0 beta1, foi divulgada em 16/12/2010; a última, 2.0.0 RC2, em 07/12/2012.

3 METODOLOGIAS DE SEGURANÇA EM APLICAÇÕES WEB

3.1 OSSTMM

A OSSTMM (*Open Source Security Testing Methodology Manual*) (ISECOM, 2010) fornece uma metodologia para a realização de testes de segurança minuciosos, também referenciados como auditorias OSSTMM. Uma auditoria OSSTMM define-se como uma avaliação precisa da segurança em um nível operacional, isentando-se de considerações não factuais e evidências anedóticas; é essencialmente baseada em métodos científicos. Além disso, tem entre seus princípios a consistência e a possibilidade de repetição.

Sendo amparada pela ISECOM(*Institute for Security and Open Methodologies*), e um projeto *open source*, a metodologia fomenta a contribuição da comunidade de segurança da informação, visando sua melhoria constante quanto a precisão, acionamento e eficiência dos testes de segurança.

3.1.1 Características gerais

Esta metodologia divide-se em quatro grupos chave: escopo, canal, índice e vetor.

O escopo, conforme Ali e Tedi (ALI, 2011), define um processo de coleta de informações dos ativos presentes no ambiente alvo. Um canal, por conseguinte, determina o tipo de comunicação e interação com esse ativos. Canais são subdivididos em três classes: física, espectro e comunicação. Cada classe descreve um conjunto único de componentes de segurança a serem avaliados e testados.

Quando da identificação específica de ativos, Ali e Tedi (ALI, 2011) ressaltam a utilidade do índice, terceiro grupo chave da metodologia; são exemplos de índices: endereços MAC (*Media Access Control*) e IP (*Internet Protocol*). Em último lugar, vetores consistem em direções a partir das quais um auditor, ou testador, orienta-se para avaliar e analisar cada ativo funcional.

3.1.2 Tipificação dos testes de segurança

A partir da metodologia OSSTMM, Ali e Tedi (ALI, 2011) nos fornecem seis diferentes tipos de testes de segurança. Essa tipificação, neste texto, foi mantida em nomenclatura inglesa. E é classificada, de maneira padroni-

zada, da seguinte maneira:

- ***Blind***: não exige quaisquer conhecimentos prévios sobre o sistema alvo. Porém, notifica-se o alvo antes da execução do teste de segurança; são exemplos de testes *blind*: hacking ético e *war games* – simulações, na forma de desafios, de ambientes virtuais de rede ou de sistemas operacionais.

- ***Double blind***: assim como no tipo *blind*, este não requer informação alguma do alvo. Entretanto, inversamente, não emite aviso antes da execução do teste; são exemplos de testes *double blind*: testes de intrusão e auditoria *black-box*, ou caixa-preta.

- ***Gray box***: há, por parte do testador, conhecimento pouco a respeito do alvo, que é informado antes da realização do teste de segurança; avaliação de vulnerabilidade é um exemplo de teste *gray box*.

- ***Double gray box***: apresenta semelhanças ao teste *gray box*. No entanto, estipula um intervalo de tempo para o teste. Adicionalmente, não compreende canais e vetores; *white-box* – avaliação caixa branca – é o exemplo mais tradicional de teste *double gray box*.

- ***Tandem***: aqui, o testador detém um conhecimento mínimo para violar o sistema alvo, sendo este notificado antes da execução do teste; são exemplos de testes *tandem*: teste *crystal box* e auditoria *in-house*.

- ***Reversal***: diferentemente das outras cinco categorias de teste, o reversal prevê conhecimento integral do sistema alvo. Além disso, os responsáveis pelo sistema atacado não serão informados sobre como e quando o teste conduzir-se-á; *Red-teaming* é uma instância desta modalidade de teste.

Red Teaming, de acordo com o *SANS Institute* (ROOM, 2003), é um processo projetado para detectar vulnerabilidades de redes e sistemas. Complementarmente, tem a incumbência de testar, a partir da perspectiva do atacante, o acesso ao sistema, redes e dados.

Dada sua flexibilidade, a metodologia OSSTMM mostra-se capaz de derivar casos de testes, os quais são logicamente segmentados de acordo com a classificação de canais supracitada.

Ali e Tedi (ALI, 2011) indicam que os casos de testes, usualmente, ocupam-se do exame do alvo perante os seguintes aspectos: segurança de controle de acesso, segurança de processo, controles de dados, localização

física, proteção de perímetro, nível de consciência da segurança, nível de confiança, proteção de controle a fraude, dentre outros.

Pela utilização de métricas de segurança, a captura do estado atual de proteção de um sistema alvo é consideravelmente útil e valiosa (ALI, 2011). A OSSTMM, para efeitos de medição de tais métricas, propõe valores conhecidos como RAV (*Risk Assessment Values*), descritos em mais detalhes na próxima seção.

3.1.3 Métricas de segurança operacional

Antes de definirmos métricas de segurança operacional, tratemos das métricas operacionais.

Métricas operacionais, de acordo com o OSSTM v3 (ISECOM, 2010), podem ser visualizadas quando medimos coisas corriqueiras acerca de um objeto, como altura, largura, ou massa. Da mesma forma, quando perguntamos o resultado de um jogo, ou registramos uma data de aniversário, estamos fazendo uso de métricas operacionais.

Assim sendo, formalmente, uma métrica operacional condensa-se em uma constante, que nos informa uma contagem factual em relação ao mundo físico que nos rodeia. São operacionais uma vez que são números sobre os quais podemos trabalhar. No entanto, para que possamos medir aspectos muito variáveis, afinal assim o é nosso mundo, necessita-se padronizar o processo de medição. Essa padronização pode ser alcançada por reducionismo, definida pelo OSSTMM v3 (ISECOM, 2010) como a extração de elementos mensuráveis a partir dos objetos mensurados.

No domínio de métricas de segurança operacional, por conseguinte, tem-se a superfície de ataque como instância mensurada e a porosidade, os controles e as limitações como elementos adequadamente mensuráveis.

Ademais, retornando aos RAVs, eles são definidos como uma medição de escala da superfície de ataque. Ou seja, a quantidade de interações não controladas estabelecidas com um alvo, que podem ser calculadas pelo balanço quantitativo entre operações, limitações e controles (ISECOM, 2010).

RAVs determinam, por uma escala, a proporção exposta da superfície de ataque. Dito isso, 100 RAVs representariam baixíssima exposição, enquanto que valores inferiores a este designariam um maior superfície disponí-vel ao atacante, segundo o OSSTMM v3 (ISECOM, 2010). Por outro lado, mais que 100 RAVs sinalizariam mais controles que o necessário. Tal excedente pode, eventualmente, acarretar no aumento dos problemas de manutenção e complexidade, à medida que mais interações são adicionadas ao escopo nesta conjuntura.

Há, porém, uma ressalva quanto aos RAVs, como afirmado pelo OSSTMM v3 (ISECOM, 2010). A pontuação não é capaz de conjecturar sobre se um alvo em particular será ou não atacado; contudo, ela indica em que regiões o alvo pode ser comprometido, contra quais ataques pode se defender, qual o alcance do atacante e, finalmente, quanto dano pode ser feito. Com isso, torna-se possível avaliar as confianças e riscos muito mais precisamente.

O valor de segurança final, obtido pelo equacionamento de diversos RAVs, é conhecido como RAV Score. O RAV Score muni o testador com a capacidade de extrair e definir marcos baseados na postura de segurança atual do alvo, consolidando, dessa forma, uma melhor proteção (ALI, 2011).

3.1.3.1 Grandezas envolvidas na composição do RAV

Minuciosamente, a medição da superfície de ataque contempla a quantificação de uma miríade de aspectos. Entretanto, muitos deles possuem definição dúbia, ou sobrepõem-se entre si. Ei-los a seguir:

- **Segurança operacional** - divide-se em visibilidade, acesso e confiança.
  - **Visibilidade**: refere-se ao número de alvos no escopo. Cada alvo deve ser contado uma única vez, associando-se-lhe um índice unívoco e consistente em relação aos demais alvos.
  - **Acesso**: diferentemente da visibilidade, aqui é contabilizado o número de acessos por ponto de interação único.
  - **Confiança**: diz respeito ao número de relações de confiança por ponto de interação único.
- **Controles** - são os mecanismos de segurança que fornecem garantia e proteção durante as interações. Classificam-se em:
  - **Autenticação**: número de instâncias de autenticação necessárias para obter acesso.
  - **Compensação**: número de instâncias de métodos empregados na respon-sibilização e compensação de todos os ativos do escopo.
  - **Subjugação**: número de instâncias de Acesso ou Confiança no escopo, as quais estritamente não permitem que controles sigam o arbítrio do usuário.
  - **Continuidade**: número de instâncias de Acesso ou Confiança no escopo, as quais asseguram a não interrupção na interação através de um canal ou vetor; mesmo em situações de falha total.

  - **Resiliência**: número de instâncias de Acesso ou Confiança no escopo, as quais não fornecem novos acessos diante de uma falha de segurança.
  - **Não-repúdio**: número de instâncias de Acesso ou Confiança no escopo, as quais fornecem um mecanismo de não-repúdio para cada inter-ação, garantindo que ela ocorreu em dado instante entre as partes identificadas.
  - **Confidencialidade**: número de instâncias de Acesso ou Confiança no escopo, as quais fornecem meios de manutenção do conteúdo de inter-ações não divulgadas entre as partes envolvidas.
  - **Privacidade**: número de instâncias de Acesso ou Confiança no escopo, as quais fornecem meios de sustentar o método de interações não divulgadas entre as partes envolvidas.
  - **Integridade**: número de instâncias de Acesso ou Confiança no escopo, as quais garantem que o processo de interação e Acesso aos ativos possui finalidade, e não pode ser corrompido, continuado, redirecionado, ou revertido sem o consentimento das partes envolvidas; é uma mudança no processo de controle.
  - **Alarme**: número de instâncias de Acesso ou Confiança no escopo, as quais registram ou emitem notificação diante de um aumento de porosidade não intencionado, ou não autorizado; ou, ainda, quando restrições e controles são comprometidos ou corrompidos.

- **Limitações** - são calculadas com base na porosidade e controles do alvo.

  - **Vulnerabilidade**: contabiliza cada falha ou erro capaz de subverter as proteções existentes.
  - **Fraqueza**: contabiliza cada falha ou erro nos controles de interatividade: autenticação, identificação, resiliência, subjugação, e continuida-de.
  - **Interesse**: contabiliza cada falha ou erro nos controles de processo: não-repúdio, confidencialidade, privacidade, integridade, e alarme.
  - **Exposição**: contabiliza ações não justificadas, falhas, ou erros que fornecem visibilidade direta ou indireta de alvos ou ativos pertencentes ao escopo.

– **Anomalia**: contabiliza elementos desconhecidos ou não identificáveis, os quais não são levados em consideração em operações normais.

3.1.4 Relatório de avaliação de segurança

Os resultados da avaliação de segurança são formalizados em um documento chamado *Security Test Audit Report*, STAR. A valia deste artefato repousa nas vantagens oferecidas à equipe técnica, que, para Ali e Tedi (ALI, 2011), poderá fortuitamente analisar os objetivos do teste, valores de avaliação de risco e saída de cada fase da auditoria.

3.2 ISSAF

O ISSAF, ou *Information System Security Assessment Framework*, é um framework estruturado, e profissionalmente revisto, que categoriza a avaliação de segurança de sistemas de informação em variados domínios. Ademais, são contempladas as especificidades da avaliação, ou critérios de teste, de cada um desses domínios (OISSG, 2006).

Ainda segundo o ISSAF (OISSG, 2006), constitui-se em objetivo primário do framework preencher os requisitos de avaliação de segurança de uma dada organização. Em um segundo momento, entretanto, ele pode ser empregado como uma referência perante outras necessidades de segurança da informação.

No tocante aos processos de segurança, inclui os fundamentos necessários a avaliação e fortalecimento da infra-estrutura de segurança, viabilizando a obtenção de um retrato completo das vulnerabilidades que possam existir.

No ISSAF, a informação sofre uma categorização baseada em critérios bem definidos de avaliação. Dentre tais critérios, destacam-se:

- Uma descrição dos critérios de avaliação
- Suas intenções e objetivos
- Os pré-requisitos para a condução de avaliações
- O processo de avaliação
- Amostra dos resultados esperados
- Contramedidas recomendadas
- Referências para documentos externos

3.2.1 Características gerais

O desenvolvimento do ISSAF, de acordo com Ali e Tedi (ALI, 2011), estabeleceu-se sobre duas áreas de teste de segurança: técnica e administrativa.

Quando da criação de um processo de avaliação de segurança adequado, Ali e Tedi (ALI, 2011) esclarecem-nos que a área técnica designa o conjunto de regras e procedimentos referencial. Em contrapartida, os aspectos administrativos são contemplados através da gestão de compromissos, assim como pelas melhores práticas a serem consideradas integralmente no processo de teste.

Ali e Tedi (ALI, 2011) ressaltam, contudo, que um ISSAF define a avaliação como um processo, e não como uma auditoria.

Quando tratada como uma iniciativa, a segurança prescinde de uma abordagem sistemática em sua realização. Para isso, o ISSAF (OISSG, 2006) fornece um modelo de quatro fases, o qual estrutura as iniciativas de segurança. Esse modelo deriva pacotes de trabalho, concebidamente chamados de atividades, que podem, posteriormente a sua derivação, serem atribuídos a pessoas pertencentes ao time de projeto.

As quatro fases são, respectivamente: planejamento, avaliação, tratamento e acreditação. O framework permite que pacotes de trabalho específicos sejam associados a cada fase do modelo de teste, independentemente do tamanho, áreas de resultado chave, ou localidade de uma organização; isso ocorre pelo usufruto da flexibilidade inerente a um ISSAF.

O documento ISSAF (OISSG, 2006) prevê o sequenciamento dos pacotes de trabalho particulares a cada uma das fases mencionadas, resultando em artefatos entregáveis, os *deliverables*, ou, então, um estado de negócios desejado. Em seguida, as saídas dessas fases são sucedidas por atividades operacionais, as quais buscam a integração do artefato entregável ou a manutenção, viável e eficaz, do estado de negócios atingido.

O framework ISSAF é abrangente, tendo sido projetado para fornecer tanta informação quanto possível. Esse escopo partiu da premissa de que, para os usuários do framework, seria mais fácil remover material que desenvolvê-lo (OISSG, 2006).

No entanto, embora pareça um objetivo ousado contemplar particularidades de diferentes tecnologias, surge, em decorrência deste fato, o problema da manutenção de uma fonte de informação demasiado volumosa. Conforme exposto por Ali e Tedi (ALI, 2011), pode ser um processo trabalhoso manter o framework atualizado, de modo que reflita novos ou atualizados critérios de avaliação de tecnologia.

Por conseguinte, desvincilhando-se da questão da obsolescência de

conteúdo, a ISSAF pode ser combinada a outras metodologias, como a OSSTMM (ALI, 2011). O casamento entre metodologias pode originar uma abordagem de testes mais robusta, estabelecida pelas forças de cada uma, e subtraída das fraquezas.

3.3 OWASP

O *Open Web Application Security Project*, OWASP, é uma organização mundial 501(c)(3) (SERVICE, 2012), sem fins lucrativos, projetada para o aprimoramento da segurança de software. Ao aumentar a visibilidade da segurança de aplicações, torna-se possível que, em escala mundial, indivíduos e organizações tomem decisões mais acertadas quanto aos verdadeiros riscos de segurança de software (OWASP, 2001).

A fundação OWASP surgiu, online, em primeiro de Dezembro de 2001, estabelecendo-se como uma entidade sem fins lucrativos anos depois, em Abril de 2004. Desde então, tem ramificado sua atuação mundialmente, através de capítulos locais e inúmeros projetos (OWASP, 2001).

Os capítulos locais ajudam a disseminar os conhecimentos agregados pelos projetos OWASP, fixando bases regionais em torno do assunto de segurança em aplicações web.

3.3.1 Principais projetos OWASP

Dentre os projetos da fundação OWASP mais relevantes, destacam-se:

- ***OWASP AntiSamy Project***: uma API para validação de entrada HTML e CSS rica oriunda dos usuários, sem, consequentemente, expor a aplicação a ataques de *Cross Site Scripting* e *Phishing*.
- ***OWASP Enterprise Security API (ESAPI) Project***: uma coleção, livre e aberta, dos métodos de segurança que um desenvolvedor necessita para construir aplicações web seguras.
- ***OWASP WebScarab Project***: uma ferramenta de proxy para a realização de uma variedade de testes em aplicações web e *web services*.
- ***OWASP Zed Attack Proxy (ZAP) Project***: é uma ferramenta de teste de intrusão, que permite a descoberta de vulnerabilidades em aplicações web; fornece tanto scanners automáticos como um conjunto de utilitários que permite a realização de testes manuais.

- ***OWASP WebGoat Project***: uma plataforma de treinamento online a respeito de segurança em aplicações.

- ***OWASP Testing Guide***: descreve os procedimentos e tarefas para testes de segurança em aplicações web; este projeto é melhor detalhado em 3.3.2.

- ***OWASP Top Ten Project***: um documento que revela as dez mais significativas vulnerabilidades em aplicações web; este projeto é melhor detalhado em 3.3.3.

3.3.2 OWASP Testing Guide

O *OWASP Testing Guide*, em sua versão número 3, descreve a metodologia OWASP de teste de intrusão em aplicação web. Além disso, fornece instruções sobre como testar cada vulnerabilidade elencada.

Um teste de intrusão é um método de se avaliar, através da simulação de ataques, a segurança de um sistema de computador ou rede. No entanto, o guia atém-se às aplicações web e suas falhas de segurança inerentes. Ao final do teste de intrusão, apresenta-se um documento descritivo acerca das vulnerabilidades identificadas e, idealmente, estratégias para a adequada mitigação das mesmas (OWASP, 2008).

O método de teste de intrusão proposto pela OWASP adota a abordagem *black-box*, ou caixa-preta. Ou seja, o testador sabe pouco, ou nada, sobre a aplicação alvo. Dessa forma, o modelo de testes compreende três itens/agentes:

- **Testador**: quem realiza as atividades de teste
- **Ferramentas e metodologia**: artilharia e guia de procedimentos
- **Aplicação**: a caixa-preta a ser testada

No que se refere aos testes, eles são divididos em duas fases: modo passivo e ativo.

O modo passivo, conforme descrito pelo *OWASP Testing Guide v3* (OWASP, 2008), integra a etapa em que o testador busca entender a lógica da aplicação. Isso normalmente é alcançado pela manipulação das funções oferecidas pelo sistema alvo. Nesta etapa, fortuitamente, para obtenção de informações, são empregadas ferramentas como um proxy HTTP, que permite a interceptação de requisições e respostas HTTP. Encerrada esta fase, o testador deve ser capaz de visualizar todos os pontos de acesso – em inglês, *gates* – da aplicação; por exemplo: cabeçalhos HTTP, parâmetros e cookies.

Na segunda fase, chamada de modo ativo, recorre-se à metodologia classificada nas seguintes 9 subcategorias, que abraçam 66 controles:

- Teste de gerenciamento de configuração
- Teste de lógica de negócio
- Teste de gerenciamento de sessão
- Teste de autorização
- Teste de validação de dados
- Teste de negação de serviço
- Teste de *Web Services*
- Teste de ambientes AJAX

3.3.3 *OWASP Top Ten Project*

Em face dos desafios impostos pela presença de software inseguro, a fundação OWASP, em 2003, lançou o projeto *OWASP top ten* (OWASP, 2010). O objetivo deste projeto é alavancar a conscientização sobre segurança de aplicações, destacando, para isso, os riscos mais críticos que acometem as organizações.

Sua última versão foi concretizada em 2010. Nesta data, fora assinalado o oitavo ano de projeto, que é atualmente referenciado e respaldado por diversos padrões, livros, ferramentas e entidades, incluindo MITRE, PCI DSS, DISA e FTC (OWASP, 2010).

O *OWASP Top Ten Project* (OWASP, 2010), ao endereçar as falhas mais substanciais em aplica-ções web, permite que organizações aprendam a partir de erros prévios, dirimidos em estratégias de teste e prevenção previstas pelo documento do projeto.

A seguir, serão descritas as características de cada uma das 10 categorias de testes sugeridas pelo *OWASP Top Ten Project*. Os nomes das classes foram mantidos em língua inglesa, em virtude da necessidade de preservação semântica.

- ***A1 - Injection***: falhas de injeção, exemplificadamente injeção de SQL, SO (Sistema Operacional) e LDAP, surgem a partir do envio de dados não confiáveis utilizados na construção de um comando ou consulta. Sua mitigação envolve a filtragem dos dados oriundos do usuário, através da filtragem ou substituição de caracteres especiais (*character escaping*).

- ***A2 - Cross-Site Scripting (XSS)***: falhas de XSS decorrem do envio, indevido, de dados não confiáveis pela aplicação para o navegador,

possibilitando que scripts maliciosos sejam executados no contexto do usuário. Nesta categoria encontram-se os ataques de sequestro de sessão, furto de cookies, adulteração de páginas web, etc.

- ***A3 - Broken Authentication and Session Management***: normalmente, as funções de autenticação e gerenciamento de sessão do usuário não são adequedamente implementadas, potencialmente viabilizando o comprometimento de senhas, chaves, tokens de sessão, ou personificação ilegal de outros usuários na aplicação.

- ***A4 - Insecure Direct Object References***: quando um desenvolver descuidadosamente expõe uma referência para um objeto de implementação interno, como um arquivo, diretório, ou chave da base de dados, abre-se um vetor de ataque na aplicação. Sem uma verificação de controle de acesso ou outra proteção, mostra-se possível a manipulação dessas referências com vistas ao acesso de dados não autorizados.

- ***A5 - Cross-Site Request Forgery***: consiste, basicamente, na reprodução de uma série de requisições HTTP em nome de um determinado usuário. Essas requisições podem, uma vez constatado um ponto de ataque desse gênero na aplicação, personificar ações, ou operações, no sistema como se elas fossem legítimas.

- ***A6 - Security Misconfiguration***: em prol da segurança, uma aplicação deve contemplar configurações adaquedas, sejam elas vinculadas a um dado framework, servidor de aplicação, servidor web, servidor de banco de dados, ou plataforma; ou seja, faz-se necessária uma adaptação das configurações da infra-estrutura da aplicação, que, para ser segura, não deve apoiar-se cegamente em definições padronizadas.

- ***A7 - Insecure Cryptographic Storage***: diante da ausência ou má definição de esquemas robustos de criptografia para informações sensíveis, dados como números de cartão de crédito, credenciais de acesso, dentre outros, são expostos pela aplicação. O impedimento deste tipo de violação ocorre pelo uso de bons algoritmos de resumo criptográfico (*hashing*) e criptografia, para a cifragem das informações confidenciais.

- ***A8 - Failure to Restrict URL Access***: muitas vezes, verificando-se o lapso do esquema de controle de permissões baseado em URL, torna-se possível acessar páginas privadas, ou não autorizadas na aplicação. A contramedida para isso, assim sendo, é a imposição de um controle rígido de acesso, vinculado às URLs, para páginas privadas.

- ***A9 - Insufficient Transport Layer Protection***: aplicações web, frequentemente, falham no processo de autenticação, cifragem ou proteção da confidencialidade e integridade de tráfego de rede sensível. As causas que promovem essa condição são, tradicionalmente, suporte a algoritmos criptográficos fracos, certificados de segurança expirados ou inválidos, ou, mais seriamente, a ausência integral de mecanismos de proteção da camada de transporte.

- ***A10 - Unvalidated Redirects and Forwards***: redirecionamentos são funções corriqueiramente empregadas em aplicações web, conduzindo o usuário a outras localidades pelo uso, em determinados casos, de parâmetros dinâmicos. Quando não há a filtragem do valor deste parâmetro de redirecionamento, observa-se a possibilidade de direcionar o usuário para um destino controlado pelo atacante; uma página falsificada – *phishing scam* –, ou um servidor de arquivos malicioso, por exemplo.

3.4 WASC

A metodologia WASC (*Web Application Security Consortium*) compreende procedimentos de identificação e classificação de ataques e fraquezas inerentes às aplicações Web. À semelhança do projeto OWASP, define-se como um padrão aberto de avaliação de segurança de aplicações.

3.4.1 WASC *Threat Classification*

Uma vez enderaçados ataques e fraquezas, as ameaças a eles associadas são sistematicamente categorizadas pelo *WASC Threat Classification* – WASC-TC (WASC, 2010). Essa classificação é um esforço colaborativo para clarificar e organizar as ameaças de segurança de um site web.

A classificação WASC-TC dispõe de três visões, ou perspectivas: enumeração (*enumeration view*), desenvolvimento (*development view*) e referência cruzada taxonômica (*taxonomy cross reference view*) (ALI, 2011).

- **Visão de enumeração**: fornece as bases para ataques e fraquezas de aplicações web. Nesta visão, cada ataque e fraqueza foi individualmente discutido, levando-se em consideração definições, tipo e exemplos concisos oriundos de múltiplas plataformas de programação. Além disso, são utilizados identificadores únicos para o referenciamento. Há, atualmente, 49 ataques e fraquezas, identificados por um número estático

chamado WASC-ID, que varia de 1 a 49. Contudo, ressalta-se que esse esquema de identificadores cumpre apenas a função de referência, e não determina a severidade de cada risco.

- **Visão de desenvolvimento**: o processo de desenvolvimento pode ser segmentando em três etapas; fase de projeto, de implementação e de implantação. Desse modo, a visão de desenvolvimento do WASC-TC converge o conjunto de ataques e fraquezas em grupos de vulnerabilidades, que podem erodir em uma das três fases de desenvolvimento citadas. Na etapa de projeto, primeiramente, são introduzidas vulnerabilidades quando, durante a obtenção de requisitos da aplicação, não são observados aspectos de segurança. As vulnerabilidades advindas da implementação, por sua vez, afloram de princípios e práticas de codificação inseguros. Finalmente, na terceira etapa, de implantação, vulnerabilidades vêm à tona assoaciadas a má configuração da aplicação, servidor web e outros sistemas externos.

- **Visão de referência cruzada taxonômica**: existe, no domínio de segurança em aplicações web, uma pluralidade de padrões de nomenclatura. Tal fato acarreta em critérios distintos para avaliação de segurança, sintetizados a partir de diferentes ângulos. No entanto, a visão que aqui se descreve busca absorver essas nuanças entre sistemas de referência heterogêneos. Segundo Ali e Tedi (ALI, 2011), os ataques e fraquezas classificados pela WASC-TC estão mapeados com o *OWASP top ten*, *Mitre's Common Weakness Enumeration* (CWE), *Mitre's Common Attack Pattern Enumeration and Classification* (CAPEC) e a lista *SANS-CWE Top 25*.

4 UMA METODOLOGIA PARA ATAQUE AO SGCI

A etapa de elaboração do escopo do aplicação levará em consideração princípios de análise de segurança e de confiança.

Conforme enunciado no OSSTMM v3 (ISECOM, 2010), a análise de segurança refere-se a habilidade de transformar informação em inteligência de segurança. Essa habilidade, no entanto, estende-se além da informação em si. Leva em consideração, também, aspectos como: origem da informação, como e quando foi coletada, e restrições do processo de coleta.

O resultado do processo de análise deve gerar inteligência acionável. Ou seja, informação e fatos dela derivados que permitam, por parte da organização, uma tomada de decisão mais apurada.

Antes de prosseguirmos, contudo, faz-se necessária a distinção entre análise de segurança e análise de risco, conceitos comumente confundidos. A primeira se refere à produção de fatos, ainda que estes se refiram a algo que não pode ser conhecido a partir da informação fornecida. Por outro lado, a análise de risco especula e deriva opiniões com base na informação.

Assim sendo, a análise de risco pode valer-se, para fornecer respostas mais precisas, da análise de segurança; conversamente, entretanto, o mesmo não se aplica.

Uma vez delineadas quatro metodologias de teste de segurança (OSSTMM, ISSAF, OWASP e WASC), e tratados alguns conceitos importantes de análise, passaremos a definição de uma abordagem pertinente ao sistema SGCI. Este capítulo está dividido em escopo, planejamento, execução e resultados.

Por conseguinte, a metodologia neste trabalho sintetizada irá, inicialmente, fundamentar-se nos princípios de análise de segurança. A partir disso, será possível determinar adequadamente a superfície de ataque do sistema SGCI. Escopo, planejamento e execução compreenderão, portanto, está fase. Em uma etapa final, porém, a metodologia apoiar-se-á nos preceitos da análise de risco, para exame das ameaças descobertas e suas correspondentes consequências.

4.1 DELIMITAÇÃO DO ESCOPO

A finalidade preponderante do SGCI é a criação e gestão de uma ICP. Com isso, a delimitação do seu escopo, que passa pelas operações básicas descritas na seção 2.3, fundamenta-se, antes de mais nada, nas relações de confiança definidas entre ARs, ACs e usuário final.

Concebidamente, o SGCI é o coração de gerenciamento dos sensíveis relacionamentos de uma ICP. Dessa forma, na perspectiva das aplicações Web, faz-se necessária uma administração segura e adequada das sessões dos usuários do sistema. Um Operador de AR, por exemplo, não deve, em hipótese alguma, ser capaz de sequestrar a sessão de um Administrador de AC. Isso representaria uma comprometimento da cadeia de confiança, cuja integridade é prevista para o correto e seguro funcionamento de uma ICP. O atacante, neste cenário exemplificado, poderia emitir listas de certificados revogados de maneira ilegítima.

Segundo Polk (POLK, 2001), uma AC pode emitir certificados para usuários, para outras ACs, ou para ambos. Consequentemente, ainda em se tratando de sequestro de sessão, um atacante que, com sucesso, tenha forjado determinado perfil do SGCI, poderá eventualmente criar, validar, outras ACs; e não somente emitir certificados perversa e arbitrariamente.

Como mencionado na seção 2.2, os quatro componentes funcionais básicos de uma ICP – AC, AR, repositório e arquivo – não necessariamente acumulam-se em um único componente. Em outras palavras, as responsabilidades da infra-estrutura podem ser distribuídas em diferentes módulos, ou partes. Essas partes, no que se refere ao SGCI, nem sempre são implementadas na mesma estação de trabalho; neste texto, estação de trabalho designa um sistema operacional implementado em máquina física ou virtual.

Por conseguinte, a descentralização da implantação do SGCI permite que instâncias de aplicação distintas sejam atribuídas aos papéis de AC e AR. Para isso, são estabelecidos relacionamentos de confiança entre as entidades distribuídas. A idoneidade desses relacionamentos, no entanto, deve ser garantida pelo SGCI. É responsabilidade da aplicação implementar corretamente os controles de segurança associados às relações de confiança da ICP em questão.

Em sua versão 2.0.0, o SGCI possui suporte a comunicação com dispositivos HSM. De acordo com Kohler e Júnior (KOHLER, 2007), "*o suporte a HSM aumenta a segurança nas operações realizadas. Isto ocorre devido ao acesso às chaves privadas ser controlado por hardware. Assim, as operações devem ser executadas pelo HSM. A confiança na segurança é movida para a confiança depositada no fabricante do HSM. O SGCI ainda oferece suporte para trabalhar com diferentes HSMs ao mesmo tempo*".

Eis que, em um cenário cuja implantação do SGCI ocorre dentro do próprio HSM, e adicionalmente adotado o modelo de AC online, vetores de ataque da aplicação web podem suplantar os benefícios do uso do HSM. Imagine que um atacante obteve acesso a linha de comando da máquina executora do SGCI. Isso pode ter sido alcançado pela exploração de uma falha de *SQL Injection*, ou RFI (*Remote File Inclusion*), por exemplo. Em seguida, uma vez

controlando o interpretador de comandos do sistema operacional, o atacante poderá fazer uma série de capturas de imagem de memória (*memory dump*). Se, no intervalo em que essa série de capturas ocorreu, a chave privada da AC realizava alguma operação criptográfica, poder-se-ia, eventualmente, obter-se tal conteúdo altamente sensível, a chave privada, em texto plano.

Visualizados os aspectos conceituais de ICP empregados pelo SGCI, será efetuada uma seleção de testes consoantes aos módulos que implementam essas particularidades. Ademais, far-se-á também uma extração de testes pertinentes aos componentes Web presentes no SGCI. A metodologia fornecedora das diretrizes de teste será a OWASP.

Na seção seguinte, 4.2, tais testes são elencados. Posteriormente, em 4.3, eles são descritos, tanto no que tange sua teoria subjacente como em relação a instrumentação dos mesmos pelo testador, e depois executados.

4.2 PLANAJEMANTO

A partir das características comumente presentes em aplicações web, juntamente àquelas correspondentes às especificidades do SGCI, elaborou-se uma lista de testes OWASP, abaixo apresentada. A denominação de cada teste segue o padrão *OWASP- categoria de teste-numeração dentro da categoria*.

- (OWASP-IG-003) *Testing: Identify application entry points*
- (OWASP-IG-004) *Testing for Web Application Fingerprint*
- (OWASP-AT-002) *Testing for user enumeration*
- (OWASP-AT-004) *Testing for Brute Force*
- (OWASP-AT-005) *Testing for Bypassing Authentication Schema*
- (OWASP-AT-006) *Testing for Vulnerable Remember Password and Pwd Reset*
- (OWASP-AT-007) *Testing for Logout and Browser Cache Management*
- (OWASP-DV-001) *Testing for Reflected Cross Site Scripting*
- (OWASP-DV-002) *Testing for Stored Cross Site Scripting*
- (OWASP-SM-001) *Testing for Session Management Schema*
- (OWASP-SM-002) *Testing for Cookies Attributes*
- (OWASP-SM-005) *Testing for CSRF*

- (OWASP-AZ-001) *Testing for Path Traversal*

Neste trabalho, embora tenha sido feita pelo próprio testador, não será contemplada a implantação do SGCI. Concebidamente, testes blackbox, abordagem aqui adotada, concentram-se em ataques sob uma perspectiva externa. Assim sendo, pressupõe-se um sistema SGCI disponível no domínio (http://pentest-sgci.com) – esta URL é válida apenas no ambiente de testes configurado, ou seja, não remete a um endereço oficialmente registrado.

4.3 EXECUÇÃO

4.3.1 Testing: Identify application entry points (OWASP-IG-003)

4.3.1.1 Objetivos

- Identificar e mapear os pontos de entrada
- Determinar a superfície de ataque da aplicação

4.3.1.2 Pré-requisitos

- Ferramenta de *proxy* (WebScarab, BurpSuite, etc)

4.3.1.3 Metodologia

Antes de realizar uma bateria exaustiva de testes, é essencial que o testador entenda as interações entre o usuário/navegador e a aplicação web. Essas interações costumam basear-se no protocolo HTTP, particularmente através dos métodos GET e POST. Portanto, uma análise meticulosa das mensagens HTTP revelará pontos de entrada pertinentes ao processo de investigação.

Um mapeamento abrangente da aplicação compreende a identificação de parâmetros, cabeçalhos, campos ocultos (*hidden fields*) presentes nas requisições e respostas HTTP. As informações coletadas neste teste devem, idealmente, ser armazenadas em um planilha ou arquivo de controle.

Entretanto, dependendo do porte e complexidade da aplicação, a quantidade de informações coletadas pode ser volumosa, remetendo a um processo de coleta dispendioso e repetitivo. Quanto a isso, a experiência do testador indicará quais partes da aplicação são mais relevantes, e correspondem a pontos

de entrada a serem investigados.

Especificamente às requisições que não são GET, para sua análise, faz-se necessária alguma ferramenta de captura de tráfego, como um *proxy*, por exemplo.

Quando da investigação de requisições HTTP, enumeram-se os seguintes aspectos a serem observados:

- Pontos em que são utilizados os métodos GET e POST
- Parâmetros de composição de requisições POST
- *Hiden fields* em requisições POST
- Parâmetros utilizados em consultas
- *Headers* adicionais tipicamente não vistos (como, por exemplo, debug=False)

Ademais, referente às respostas HTTP, atenta-se para:

- Definição, modificação e adição de cookies (*Set-Cookie header*)
- Presença de redirecionamentos HTTP (averiguar códigos de estado HTTP)
- Uso de *headers* específicos (tal qual, por exemplo, "Server: BIG-IP")

4.3.1.4 Execução

Com o auxílio da ferramenta ZAP, obteve-se a estrutura de diretórios do sistema SGCI. Além disso, pela mesma ferramenta, foram identificadas cada requisição e resposta referentes às páginas da aplicação. A função de *Spider* do ZAP, adicionalmente, forneceu mais informações a respeito de diretórios eventualmente ocultos, quando da navegação pelo próprio navegador.

A figura 2 ilustra uma captura da tela do ZAP, em uma sessão de navegação pelo SGCI.

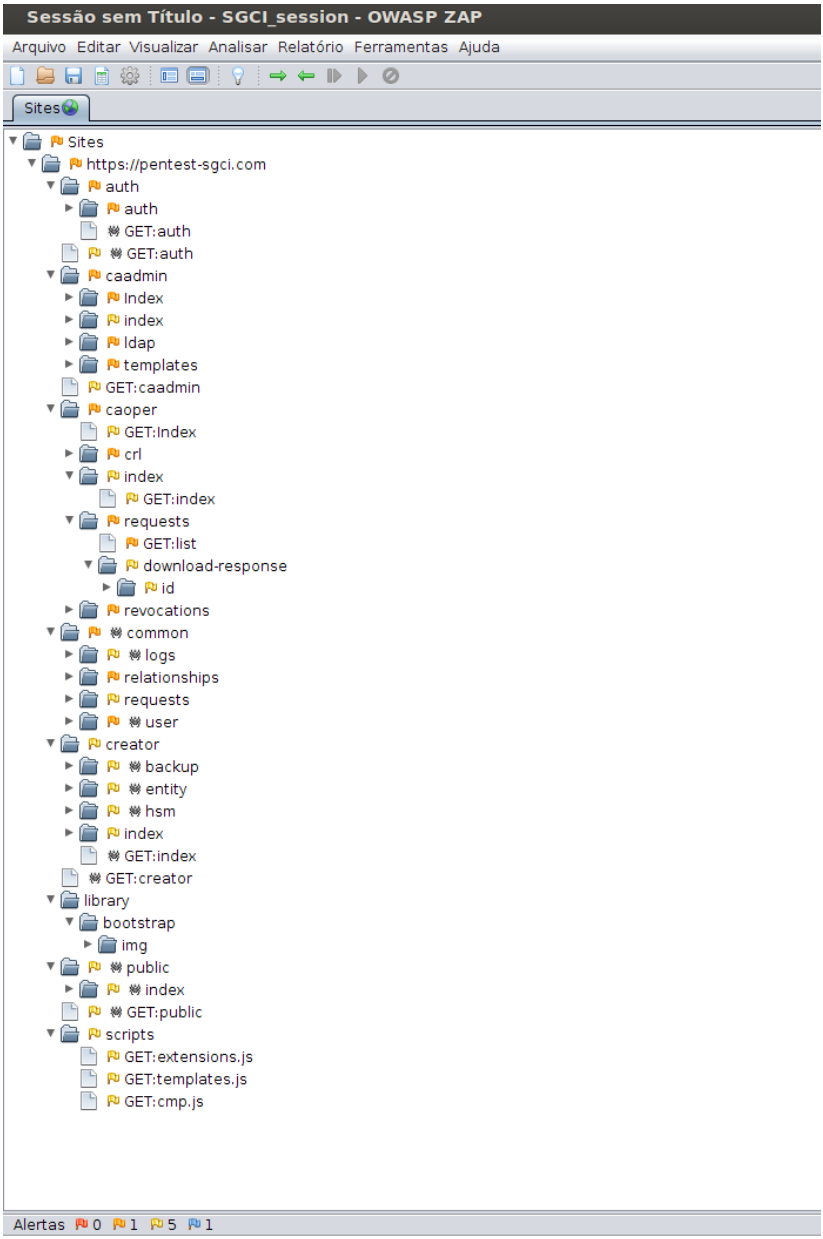

Figura 2: Estrutura de diretórios do sistema SGCI

Uma vez identificados os caminhos, páginas e campos da aplicação, assim como após a geração de um relatório do material descoberto, tem-se uma base sólida para a realização de testes mais específicos. Esse relatório, que contém os pontos de entrada e seus detalhes – como método HTTP utilizado, presença de túnel seguro no tráfego de dados sensíveis, atributos de cookies, etc –, pode ser gerado pelo próprio ZAP.

Por conseguinte, a automatização dessa etapa de teste passa a ser fortuita, se ponderado o porte da aplicação testada. Ressalta-se, no entanto, a necessidade da averiguação manual de determinados pontos; ferramentas têm suas limitações quando comparadas ao cérebro humano.

4.3.2 Testing for Web Application Fingerprint (OWASP-IG-004)

4.3.2.1 Objetivos

- Identificar o tipo e a versão do servidor web

4.3.2.2 Pré-requisitos

- Ferramenta de requisições HTTP (Netcat, por exemplo)

4.3.2.3 Metodologia

Aplicações web são executadas a partir de servidores web, cujas características, portanto, passam a ser pertinentes a um teste de intrusão. Desse modo, durante o teste, importamo-nos especialmente com a versão e o tipo do servidor web. Uma vez sob posse dessas informações, uma investigação mais eficaz acerca de vulnerabilidades existentes torna-se possível.

Tradicionalmente, a aquisição da versão e tipo do servidor web começa pelo envio de uma série de comandos HTTP ao mesmo. Após processados, os comandos geram respostas cujo padrão pode revelar características do servidor sendo testado. Entretanto, servidores distintos podem retornar respostas semelhantes para um dado comando HTTP. Em face disso, quanto maior a quantidade de comandos e respostas avaliados, mais precisa é a inferência sobre a versão e espécie do servidor web.

Durante o teste, para a geração de comandos HTTP, empregam-se ferramentas como o *Netcat* – popular e versátil ferramenta para manipulação de conexões de rede. Através do *Netcat*, o testador conecta-se à aplicação, for-

nece um comando HTTP e, finalmente, examina o campo *Server* da resposta. Normalmente esse campo oferece o tipo e versão do servidor web. Contudo, considerando a possibilidade de adulteração dessa informação pelo servidor, uma análise mais apurada envolveria observar a ordem em que os headers são retornados. Cada fabricante de servidores web mantém uma ordem particular às respostas que emite.

Outra técnica de investigação atém-se ao envio de requisições HTTP malformadas ao servidor web. Em seguida, são diagnosticadas as respostas e suas especificidades inerentes, que anunciam padrões em sua estrutura e conteúdo.

Ademais, alternativamente à utilização do *Netcat*, existem ferramentas de automatização do processo de coleta. Uma delas, nomeadamente, é o *httprint*; ele permite o carregamento de dicionários de assinaturas de servidores web, contra as quais são comparadas as respostas ecoadas pelo servidor.

4.3.2.4 Execução

Conectando-se ao servidor do SGCI, na porta 80, através da ferramenta *Netcat*, podemos observar algumas informações de versão retornadas. Conforme o cabeçalho *Server*, a aplicação está sendo executada sobre um servidor Apache 2.2.22, em uma distribuição *linux* Ubuntu.

```
lucas@lvrosa-labsec:~$ nc pentest-sgci.com 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Tue, 18 Sep 2012 21:22:05 GMT
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Wed, 04 Jul 2012 18:30:48 GMT
ETag: "e3b3c-c-4c4053ae6e4e2"
Accept-Ranges: bytes
Content-Length: 12
Vary: Accept-Encoding
Connection: close
Content-Type: text/html
```

Fornecendo uma versão de protocolo HTTP inexistente, o servidor devolve as seguintes informações:

```
lucas@lvrosa-labsec:~$ nc pentest-sgci.com 80
GET / HTTP/3.0

HTTP/1.1 400 Bad Request
Date: Tue, 18 Sep 2012 21:26:38 GMT
Server: Apache/2.2.22 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 368
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

Complementarmente, a submissão de uma requisição cujo protocolo não está disponível traz-nos bons indícios de que, realmente, trata-se de um servidor Apache 2.2.22.

```
lucas@lvrosa-labsec:~$ nc pentest-sgci.com 80
GET / JUNK/1.0

HTTP/1.1 200 OK
Date: Tue, 18 Sep 2012 21:28:43 GMT
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Wed, 04 Jul 2012 18:30:48 GMT
ETag: "e3b3c-c-4c4053ae6e4e2"
Accept-Ranges: bytes
Content-Length: 12
Vary: Accept-Encoding
Connection: close
Content-Type: text/html
```

4.3.3 Testing for user enumeration (OWASP-AT-002)

4.3.3.1 Objetivos

- Enumerar os usuários válidos da aplicação
- Auditoria do esquema de mensagens de erro e URL

4.3.3.2 Pré-requisitos

- Credinciais válidas (usuários e senhas) do sistema a ser testado

4.3.3.3 Metodologia

O objetivo deste teste é verificar a possibilidade de coleta e enumeração de usuários válidos. Posteriormente, por intermédio de um ataque de força bruta, poderemos descobrir as senhas vinculadas aos usuários enumerados. Isso permitirá o acesso a funcionalidades restritas do sistema. Para tanto, devem ser testadas combinações de credenciais válidas e inválidas, sucedendo-se um exame das mensagens de erro e URLs.

Conforme recomendado pelo *OWASP Testing Guide v3* (OWASP, 2008), o testador deverá seguir as seguin-tes etapas:

1. Realizar o teste somente com o usuário incorreto, sem inserir a senha
2. Realizar o teste com um usuário incorreto e com uma senha qualquer
3. Realizar o teste com um usuário válido e com uma senha incorreta

Ao executar as etapas enumeradas acima, é possível, através das respostas devolvidas pelo sistema testado, concluir quanto à existência ou não de um usuário. Por exemplo, ao testarmos um *login* qualquer, a aplicação pode fornecer indícios, ou mesmo provas, de que aquele *login* de fato existe.

A análise de URLs é outro vetor pelo qual usuários podem ser enumerados. Considere o seguinte exemplo de respostas emitidas pela aplicação:

```
http://www.foo.com/err.jsp?User=baduser\&Error=0
http://www.foo.com/err.jsp?User=gooduser\&Error=2
```

Note que existe um código associado à existência do usuário em questão, o que nos permite constatar sua validade.

4.3.3.4 Execução

- **Teste somente com o usuário incorreto (sem inserção da senha)**: o SGCI não permite autenticação sem digitação da senha;
- **Teste com um usuário incorreto e com uma senha qualquer**: retornada a mensagem ‘Login ou senha incorretos‘;

- **Realizar o teste com um usuário válido e com uma senha incorreta**: retornada a mensagem ‘Login ou senha incorretos‘;

Para os dois últimos testes, a presença da palavra ‘ou‘ na mensagem de resposta obscurece qual campo está incorreto, *login* ou senha.

Adicionalmente, a aplicação não retorna mensagens de erro em URLs. Também não foi possível identificar padrões de resposta correspodentes a validade de um dado usuário.

4.3.4 Testing for Brute Force (OWASP-AT-004)

4.3.4.1 Objetivos

- Classificar tipo de mecanismo de autenticação
- Realizar teste de exaustão (ou força bruta) por dicionário

4.3.4.2 Pré-requisitos

- Uma ferramenta de *brute force*, por exemplo o Hydra ou o FireForce

4.3.4.3 Metodologia

A abordagem a ser utilizada neste ataque é capturar, com o auxílio de uma ferramenta de *proxy*, as requisições HTTP no contexto da comunicação cliente/servidor. Logo após, far-se-á a identificação do mecanismo de autenticação.

- *HTTP Authentication*
  - *Basic Access Authentication*
  - *Digest Access Authentication*
- *HTML Form-based Authentication*

Uma vez identificado o mecanismo, o testador deverá utilizar uma ferramenta de força bruta adequada para descobrir senhas de acesso ao sistema. São exemplos de ferramentas que cumprem o propósito deste teste: *THC-Hydra* e o *plugin FireForce*, para navegadores Firefox.

4.3.4.4 Execução

O mecanismo de autenticação utilizado pelo SGCI é o *HTML Form-based Authentication*. O código cliente da página revela a existência de um formulário de autenticação, que utiliza o método HTTP POST. A requisição HTTP POST relacionada ao processo de *login* pode ser visualizada na figura 3.

```
POST https://pentest-sgci.com:443/auth/auth/authentication HTTP/1.1
Host: pentest-sgci.com
Connection: keep-alive
Content-length: 106
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Origin: https://pentest-sgci.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64)
Content-Type: application/x-www-form-urlencoded
Referer: https://pentest-sgci.com/auth/auth/authentication
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
Cookie: PHPSESSID=10n2gf6nakf9ht36lllbtnk2f4

no_csrf=cd1b85bdcca4940f5719880cc5224c76&role=Creator&entity=1
&username=username&password=password&submit=
```

Figura 3: Requisição HTTP POST para autenticação no SGCI

Foram realizadas tentativas de força bruta por meio das ferramentas *THC-Hydra* e *FireForce*. No entanto, o *FireForce* apresentou demasiados falsos positivos – caso em que constata-se uma senha para um usuário, porém a mesma é falsa – ou falsos negativos – a senha verdadeira para um dado usuário, embora presente no dicionário de entrada, não é detectada.

Quanto ao *THC-Hydra*, a aplicação utiliza um *token* especial de proteção, chamado *no_csrf*. Sua função principal é evitar ataques de *Cross Site Request Forgery*, vistos na seção 4.3.12. Para isso, os valores pelo token assumidos devem ser de difícil previsão, valendo-se de bons algoritmos de geração de *strings* aleatórios.

Desse modo, a automatização do ataque de força bruta pelo *THC-Hydra* mostra-se inviável, uma vez que novos valores para *no_csrf* são gerados a cada tentativa de autenticação.

Abaixo, são exibidos alguns valores para o token *no_csrf*, definido como um campo oculto – *hidden field* – no código HTML.

```
<input type="hidden" name="no_csrf"
      value="df5478a047b2fec7720901310ebc9428"
```

```
            id="no_csrf">

<input type="hidden" name="no_csrf"
      value="2f41f36f766ca03d4a7c4745501aa524"
            id="no_csrf">

<input type="hidden" name="no_csrf"
      value="4b5206491aa62573c2d87ce1aaa97b53"
            id="no_csrf">

<input type="hidden" name="no_csrf"
      value="899e976264ca856490ac0136134fab24"
            id="no_csrf">
```

4.3.5 Testing for Bypassing Authentication Schema (OWASP-AT-005)

4.3.5.1 Pré-requisitos

- Ferramenta de *proxy* (WebScarab, BurpSuite, etc)

4.3.5.2 Objetivos

- Contornar esquema de autenticação
  - *Direct page request (forced browsing)*
  - *Parameter Modification*
  - *Session ID Prediction*
  - *Form authentication SQL Injection*

4.3.5.3 Metodologia

Há, segundo o projeto OWASP, algumas técnicas voltadas à violação do esquema de autenticação. A verificação desse esquema envolve os seguintes testes:

- ***Direct page request (forced browsing) -*** O testador deverá tentar acessar páginas restritas, disponíveis somente após o processo de *login*.

- ***Parameter Modification -*** Se a verificação de autenticação for baseada em algum parâmetro, o testador deve modificá-lo, a fim de autenticar-se. Duas formas são possíveis para este teste. Na primeira delas, o parâmetro é passado via URL; desse modo, o testador pode alterá-lo na própria barra de endereços. Na segunda, são utilizadas requisições HTTP capturadas, cujos parâmetros podem ser adulterados.

- ***Session ID Prediction -*** Em sistemas baseados em *Session IDs*, para controlar a autenticação, o testador tentará descobrir um padrão na formação dos identificadores de sessão. Inicialmente, é avaliado o uso do *Session ID* para autenticação e, depois, seu padrão de formação.

- ***SQL Injection for authentication forms*** - A principal condição para injeção de código SQL é a não filtragem de parâmetros que têm como origem o usuário. Seja em consultas, ou mesmo em formulários de autenticação, se preenchida essa condição, torna-se possível testar variações de querys e, de acordo com elas, observar o comportamento da aplicação.

  Por conseguinte, a constatação da injeção de SQL pode ser trabalhada manualmente, atentando-se a todas as consultas e parâmetros envolvidos nas mesmas no contexto da aplicação, ou automaticamente – abordagem tradicionalmente empregada em situações de *Blind Injection*, cujo esforço, em termos de número de querys necessárias, é muito alto.

4.3.5.4 Execução

Direct page request (forced browsing): as URLs listadas a seguir foram acessadas diretamente, procurando-se contornar o esquema de autenticação. Cada subgrupo de URLs corresponde às operações da aplicação, disponíveis ideal e originalmente apenas no contexto de determinado perfil de usuário. Note-se que a apresentação das URLs é parcial, cujo prefixo deve ser ajustado para o caminho onde reside a aplicação SGCI.

- Contexto do usuário ‘Criador‘
  /creator/index/index
  /creator/entity/list
  /creator/hsm/register
  /common/user/assignrole
  /common/user/createuser
  /common/user/listusers
  /common/user/listauthusers

- Contexto do usuário ‘Administrador‘
  - /caadmin/Index/configure
  - /caadmin/templates/crltemplate
  - /caadmin/templates/create
  - /caadmin/templates/list
  - /caadmin/ldap/configure
  - /caadmin/ldap/synchronize
  - /common/user/createuser
  - /common/user/assignrole
  - /common/user/listauthusers
  - /common/relationships/list
  - /caadmin/relationships/importtrustedrelationship
  - /common/relationships/registeronline
  - /caadmin/logs/export
  - /caadmin/logs/delete

- Contexto do usuário ‘Operador‘
  - /caoper/Certificates/list
  - /caoper/requests/importrequest
  - /caoper/requests/list
  - /caoper/revocations/importrequest
  - /caoper/revocations/list
  - /caoper/crl/issue
  - /caoper/crl/download

A verificação das URLs supracitadas ocorreu manualmente. Sua abordagem complementar, a automática, foi inviabilizada pela presença do *token no_csrf*, gerado aleatoriamente, e utilizado em cada requisição submetida à aplicação.

Parameter modification: alteraram-se, para os efeitos deste teste, os parâmetros da requisição de autenticação, ilustrada na figura 3. No entanto, as tentativas foram frustradas.

Session ID Prediction: através da ferramenta WebScarab, capturou-se uma resposta HTTP com a diretiva Set-Cookie. A partir desta, instrumentou-se a ferramenta de modo a gerar uma série de valores de sessão, *cookies*. A figura 4 exibe a distribuição dos valores de sessão obtidos, para um determinado intervalo de tempo – de poucos segundos. Não obstante à presença do *token no_csrf*, cujos valores são de difícil previsibilidade, os *cookies* apresentam uma variação muito dispersa. Deriva-se dessa análise, portanto, que um bom algoritmo de geração está sendo utilizado na composição de tais valores.

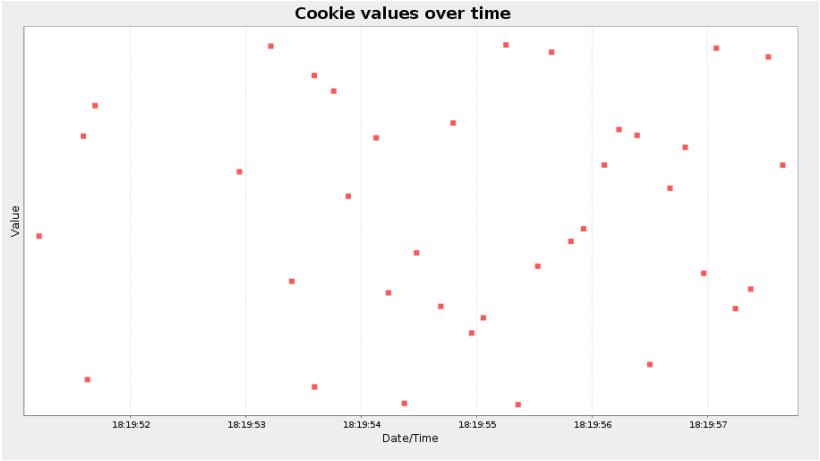

Figura 4: Gráfico de distribuição de *Session IDs* do SGCI

SQL Injection for authentication forms: dividiu-se, aqui, a abordagem de teste em duas etapas. Uma manual, outra automática.

O procedimento manual operou expressões canônicas de injeção SQL, como:

```
1’ or ’1’ = ’1
1’ or ’1’ = ’1’))/*
```

No entanto, ainda que efetuadas alterações nos caracteres da expressão injetada, as tentativas não se confirmaram.

Para o teste automático, por sua vez, utilizou-se a ferramenta SQLmap, cuja entrada fora alimentada pelos seguintes comandos:

```
$ python sqlmap.py -r post_request.txt -p username
$ python sqlmap.py -r post_request.txt -p password
```

A opção ‘r‘ lê o arquivo post_request.txt, que contém uma requisição de autenticação semelhante à disposta na figura 3. Através de ‘p‘, foram testados quanto à injeção os campos *username* e *password*.

Igualmente ao resultado do teste manual, não foram identificados, na condução automática, campos injetáveis na autenticação.

4.3.6 Testing for Vulnerable Remember Password and Pwd Reset (OWASP-AT-006)

4.3.6.1 Objetivos

- Testar função de lembrete de senha
- Explorar funcionalidade de redefinição de senha

4.3.6.2 Pré-requisitos

- Credinciais válidas (usuários e senhas) do sistema a ser testado
- Ferramenta de proxy (WebScarab, BurpSuite, etc)

4.3.6.3 Metodologia

Este ataque consiste em verificar se a aplicação web implementa o mecanismo de lembrete de senha, para tornar o acesso automático em visitas futuras. Além disso, avalia a disponibilidade de um meio de redefinição de senha e sua eficaz segurança. Este teste pressupõe o conhecimento de nomes de usuários válidos.

4.3.6.4 Execução

Teste da função de lembrete de senha: não há recurso semelhante na aplicação.

Teste da função de redefinição de senha: o perfil Criador tem poderes para alterar as senhas dos demais usuários do sistema; obviamente, esse poder se estende a si mesmo.

Entretanto, as redefinições de senha são auto contidas na aplicação. Em outras palavras, para definir uma nova senha, o usuário não depende de mecanismos externos, como notificações enviadas para um *e-mail*.

Adicionalmente, destaca-se o fato da comunicação entre navegador e aplicação usar HTTPS, o que protege a antiga e a nova senha transmitidas na requisição.

4.3.7 Testing for Logout and Browser Cache Management (OWASP-AT-007)

4.3.7.1 Objetivos

- Acessar informações sensíveis após *logout*
- Reproduzir, no sistema, ação prévia após *logout*
- Analisar tempo de expiração de sessão de usuário

4.3.7.2 Pré-requisitos

- Ferramenta de proxy (WebScarab, BurpSuite, etc)

4.3.7.3 Metodologia

Neste teste são feitos alguns diagnósticos, como, por exemplo, se a função de *logout* apaga todas as informações críticas após o encerramento da sessão do usuário. Também é analisado se o servidor faz as devidas verificações sobre o estado da sessão, não permitindo ao invasor reproduzir alguma ação prévia, quando o usuário ainda estava autenticado. Por fim, observa-se se existe algum tempo limite para expirar a sessão automaticamente. Tempo esse que deve também ser verificado se é pré-configurado no servidor.

4.3.7.4 Execução

Baseando-se na resposta HTTP relacionada ao *logout*, ilustrada na figura 5, são observados determinados aspectos do gerenciamento de sessão do SGCI.

O primeiro deles é a diretiva *Expires*, que estipula um período de expiração retroativo para a sessão do usuário. Esta data, 'Thu, 19 Nov 1981 08:52:00 GMT', curiosamente, refere-se ao nascimento de um desenvolvedor, que incluiu este recurso de controle de *cache* na linguagem PHP.

Conforme a RFC2616 (GROUP, 1999), tendo sido a expiração definida para o passado, novas requisições para recursos da aplicação são necessariamente validadas pelo servidor de origem.

Ainda de acordo com a RFC2616 (GROUP, 1999), a diretiva Cache-

Control definida pelo servidor de origem, quando constituída pela expressão *must-revalidate*, obriga a validação de todas as requisições pelo HTTP/1.1 *cache*.

```
HTTP/1.1 200 OK
Date: Sat, 17 Nov 2012 19:16:31 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.2
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
X-Content-Encoding: gzip
Content-length: 1837
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

Figura 5: Resposta HTTP para *logout* no SGCI

Ou seja, os três objetivos deste teste, para o SGCI, não foram frutíferos.

4.3.8 Testing for Reflected Cross Site Scripting (OWASP-DV-001)

4.3.8.1 Objetivos

- Enumerar pontos de entrada de dados do usuário não filtrados
- Executar código cliente malicioso no contexto da enumeração

4.3.8.2 Pré-requisitos

- Ferrementa de varredura de aplicação Web, Spider/Crawler
- Ferramenta de proxy (WebScarab, BurpSuite, etc)
- Codificador/decodificador de conjunto de caracteres

4.3.8.3 Metodologia

Ataques do tipo *Reflected Cross Site Scripting* exploram a não filtragem, cuja responsabilidade geralmente recaí ao lado servidor da aplicação, de entradas fornecidas pelo usuário, convencionalmente, a partir de formulários. Um código cliente, Javascript, por exemplo, fornecido como valor de alguma

variável contida numa requisição HTTP pode ser retornado, pelo servidor de aplicação, em uma resposta HTTP de caráter malicioso.

O procedimento de verificação para esse tipo de XSS consiste em se testar entradas Javascript nos formulários de consulta da aplicação, assim como o envenenamento de variáveis na própria URL associada a uma solicitação HTTP - a alteração dos parâmetros de composição da requisição pode ser realizado, quando assim for necessário, através de uma ferramenta de proxy. Caso o código seja efetivamente executado pelo navegador, significa que a resposta HTTP retornada pelo servidor refletiu o código malicioso. Ou seja, a aplicação está vulnerável.

A constatação da condição de XSS não-persistente, como também é conhecido o *Reflected* XSS, pode ser automatizada por ferramentas. Estas tem como heurística a análise da resposta HTTP correspondente a requisição constituída por dados não confiáveis. Analisa-se, portanto, numa abordagem automatizada, o conteúdo da resposta HTTP, e não sua execução propriamente dita.

4.3.8.4 Execução

Na seção referente ao teste OWASP-AT-005, dispõe-se uma lista de URLs da aplicação SGCI. Para nenhuma das entradas desta lista foram encontrados vetores para o ataque de XSS refletido. Isso se deve, em parte, à ausência de funções de busca na aplicação; característica inerente aos próprios requisitos e funcionamento do SGCI.

4.3.9 Testing for Stored Cross Site Scripting (OWASP-DV-002)

- Enumerar pontos de entrada de dados do usuário não filtrados
- Armazenar e executar código cliente malicioso no contexto da enumeração

4.3.9.1 Pré-requisitos

- Ferrementa de varredura de aplicação Web, Spider/Crawler
- Ferramenta de proxy (WebScarab, BurpSuite, etc)
- Codificador/decodificador de conjunto de caracteres

4.3.9.2 Metodologia

Ataques da classe *Stored XSS*, ou ataques de *Cross Site Scripting* persistente, são caracterizados pelo armezenamento do código malicioso na base de dados da aplicação. Assim sendo, o raio de ataque, em relação ao Reflected XSS, é aumentado. Isso acontece porque as vítimas potenciais passam a ser todos os usuários que acessam a página que faz referência ao conteúdo malicioso. O servidor de aplicações, quando da recuperação e exibição dos dados para o usuário, representa para o navegador uma fonte confiável de dados. Dessa forma, o código inescrupuloso trazido da base de dados pelo servidor de aplicações é efetivamente executado pelo navegador no contexto do usuário.

A detecção dessa categoria de vulnerabilidade de injeção de código se dá de maneira ligeiramente similar ao *Reflected XSS*. Testes caixa-preta envolvendo os parâmetros utilizados em requisições HTTP permitem que seja avaliado, uma vez tendo sido emitida uma requisição HTTP com conteúdo malicioso, o armazenamento conclusivo desse código de entrada na base de dados da aplicação. A confirmação da falha, por sua vez, ocorre pela consulta ao dado supostamente armazenado. Se, ao acessar a página que recupera o código malicioso, este for executado pelo navegador, consagra-se o XSS persistente.

Ressalta-se, entretanto, que, diferentemente do ataque de *Reflected XSS*, este que aqui se descreve concede maior atenção aos formulários de entrada de dados que serão armazenados na estrutura de persistência da aplicação.

Prosseguindo, discutamos sobre um importante aspecto sobre ataques de *XSS*. Ao se testar uma condição de *Cross Site Scripting*, independentemente da categoria de ataque, é pertinente confirmar a vulnerabilidade em diferentes navegadores.

Muitas vezes, uma determinada entrada é renderizada por um navegador, mas não por outro. Isso se deve ao modo particular como cada um lida com a codificação de caracteres. Dado isso, e pressupondo a existência de filtros contornáveis (tanto no lado servidor, como na perspectiva do cliente), cabe ao testador empregar ousadia e criatividade na composição das entradas.

Ferramentas de codificação e decoficação de caracteres são ótimos auxiliares na elaboração de expressões Javascript, especialmente no tocante às suas variadas formas de representação.

4.3.9.3 Execução

Na seção referente ao teste OWASP-AT-005, dispõe-se uma lista de URLs da aplicação SGCI. A partir desta relação, foram selecionados os endereços abaixo exibidos, que apresentam em suas respectivas páginas pelo menos um ponto de entrada de dados externos.

Considerando as características do ataque de *Stored XSS*, a página de inserção do código cliente pernicioso não necessariamente é a mesma que exibe esse código, após persistido. Em determinadas situações, o código malicioso é visualizado por outro perfil de usuário, que não aquele que fora o agente da inserção.

Criação de entidades na ICP:

- Página(s) de inserção:

  /creator/entity/registerrootca

  /creator/entity/registerintermediateca

  /creator/entity/registerra

As três páginas acima listadas comungam os campos a seguir apresentados. Note-se, entretanto, que no contexto da criação de uma AC raiz, devido a uma exceção gerada pela aplicação SGCI, os campos de formulário das seções 'Nome alternativo do emissor' e 'Nome alternativo do sujeito' não puderam ser efetivamente averiguados.

A exceção referida, que faz menção a expiração de um tempo máximo de execução limite, é:

```
Error information:

Message: [ Fatal Error] [client x.x.x.x] Maximum execution
time of 30 seconds exceeded in  /var/www/sgci/library/Labs
ecCL/Security/Crypto/KeyPairBuilderCL.php on line 183
```

Conseguintemente, estes campos foram selecionados como não filtrados, e passíveis de injeção de código malicioso:

- Em 'Cadastrar entidade'

  Nome Comum

  Organização

  Unidade da Organização

  Cidade

- Em ‘Opções avançadas do sujeito‘:

  Nome Comum

  E-mail

  Organização

  Unidade da Organização

  Cidade

  Estado

  Titulo

  Nome dado

  Iniciais

  Pseudônimo

  Gerar qualificação

- Página(s) de visualização:

  /creator/entity/list

  /common/entity/displayentitydetails/id/X

  /common/relationships/visualize/id/Z

Para último item da lista acima, X designa o identificador da entidade recentemente criada; Z refere-se ao identificador de algum relacionamento de confiança que envolva a entidade cujos dados foram subvertidos.

Perfis afetados pela vulnerabilidade: Criador, e todos aqueles que visualizam dados das entidades criadas.

Emissão de certificados digitais:

- Página(s) de inserção:

  /caoper/requests/importrequest

  /raoper/requests/importrequest

A emissão de certificados digitais, no SGCI, tem como primeiro passo a geração de uma requisição de certificado. Essa requisição, por sua vez, pode ser gerada de duas diferentes maneiras.

Na primeira, a requisição é sintetizada pelo próprio sistema SGCI. Caso em que são criadas, por exemplo, novas entidades em uma dada ICP. Todavia, como observado acima, durante sua criação, as requisições vinculadas às entidades não filtram determinados dados que as compõem.

Assim sendo, o dado malicioso referente a um campo da entidade irá fazer parte também da requisição e, perigosamente, do certificado digital depois emitido.

Na segunda forma de geração de requisição, um usuário utiliza ferramentas externas para gerá-la. Para os testes neste trabalho efetuados, utilizou-se o OpenSSL – *Open Secure Sockets Layer* – para tal procedimento. Conforme será ilustrado a seguir, entradas arbitrárias para campos de certificado são aceitas pelo OpenSSL.

Oportunamente, analisemos os dois cenários enunciados.

No primeiro, não existe um controle de filtragem por parte do SGCI, ao se gerar uma requisição. No segundo, o OpenSSL não implementa filtros quanto ao dados que irão compor um certificado digital. Portanto, tem-se uma fonte de dados interna à aplicação SGCI, e outra externa; infelizmente, ambas falham, na medida em que permitem a passagem de código malicioso para o fluxo de execução do SGCI.

- Página(s) de visualização (para Operador de AR):

  /raoper/requests/list

  /raoper/Certificates/list

  /raoper/requests/approve?requests[]=Y

Para último item da lista acima, Y designa o identificador da requisição a ser aprovada.

- Página(s) de visualização (para Operador de AC):

  /caoper/requests/list

  /caoper/Certificates/list

  /common/requests/displayrequestdetails/id/W

Para último item da lista acima, W designa o identificador da requisição cujos detalhes desejam ser exibidos.

Há, contudo, uma ressalva quanto à persistência do código Javascript. Nas visões de operador de AR e de AC, a edição das informações da requisição remete à filtragem das entradas maliciosas, pela aplicação SGCI. Ou seja, a não edição de tais requisições é premissa irrevogável para um cenário de ataque se materializar.

Quando da revogação do certificado com entradas Javascript, as seguintes páginas também executam o código malicioso:

- Página(s) de visualização (para Operador de AC e de AR):

  /caoper/revocations/list

  /raoper/revocations/list

Decididamente, podemos observar que um campo de entidade, ou de requisição de usuário, envenenado alastra-se horizontalmente pelo SGCI. Conforme essa dispersão ocorre, passando pelas etapas da lógica de negócio da aplicação, diversos perfis de usuário sensíveis são afetados. Isso representa uma falha seríssima para sistemas de ICP, podendo acarretar no comprometimento de parte da cadeia de certificação.

Perfis afetados pela vulnerabilidade: Operador de AC, operador de AR, e todos aqueles que visualizam dados de requisições e certificados dissimulados pelo atacante.

Gestão de usuários:

- Página(s) de inserção:

  /common/user/createuser

  /common/user/assignrole

Os perfis Criador e Administrador (de AC e de AR) são capazes de gerir usuários no SGCI. Na página de criação de um novo usuário, no formulário ‘Cadastro de Usuário‘, o campo ‘Nome‘ está vulnerável à injeção de código malicioso. Neste caso, a figura do atacante é restrita aos perfis, neste parágrafo, destacados.

- Página(s) de visualização (para Operador de AC e de AR):

  /common/user/listauthusers

  /common/user/listusers

O Javascript mal intencionado, antes persistido, é satisfatoriamente visualizado por todos os usuários com permissão nos dois endereços imediatamente supracitados – referentes à listagem de usuários e de usuários de autenticação.

Perfis afetados pela vulnerabilidade: Criador, Administrador de AC, Administrador de AR, e todos aqueles que visualizam dados de usuários normais e/ou usuários de autenticação.

Templates:

- Página(s) de inserção:

  /caadmin/templates/create

Ressalva semelhante àquela descrita para as emissões de certificados no SGCI aplica-se aos *templates*. A edição das informações de template remete à filtragem da entrada maliciosa, no lado servidor. Em outras palavras,

a não edição de tais requisições é premissa necessária para um cenário de ataque se concretizar.

- Página(s) de visualização (para Administrador de AC):

  /caadmin/templates/list

  Perfil afetado pela vulnerabilidade: Administrador de AC.

A figura 6 demonstra a renderização, em um navegador Firefox, de uma página que recupera código cliente malicioso anteriormente armazenado.

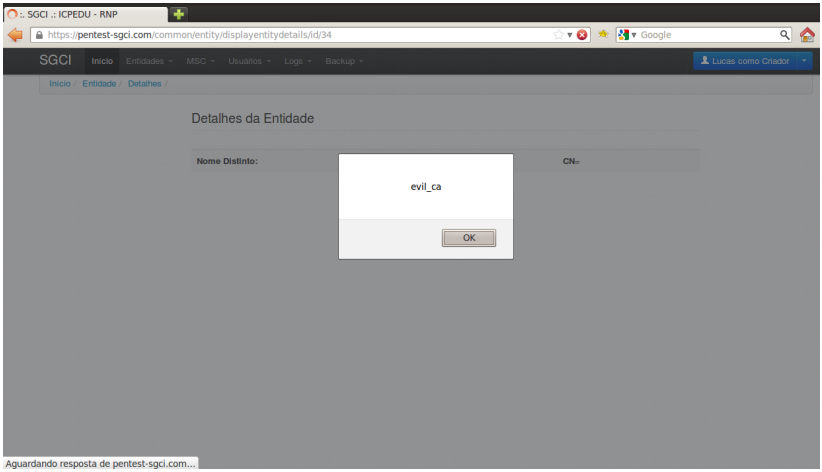

Figura 6: Captura de tela para exibição de XSS armazenado no SGCI

Para o teste de injeção das entradas, foram utilizadas construções em Javascript, da forma:

```
<script>alert("XSS");</script>
"><script>alert("XSS");</script>
<script>alert(document.cookie);</script>
```

A figura 7 ilustra, ordenadamente, os passos de um ataque de XSS persistente possível no SGCI. Desse modo, em seis etapas, mostra-se viável a persistência do código malicioso pelo atacante, assim como sua recuperação e execução pelo navegador da vítima: neste caso, administrador ou operador de AC/AR.

Figura 7: Cenário de ataque XSS sobre o SGCI

4.3.10 Testing for Session Management Schema (OWASP-SM-001)

4.3.10.1 Objetivos

- Analisar atributos associados aos cookies
- Investigar mecanismo de geração de informação de sessão
- Explorar tempo de expiração do cookie
- Classificar tipo de transporte pelos cookies utilizado

4.3.10.2 Pré-requisitos

- Ferramenta de proxy (WebScarab, BurpSuite, etc)
- Ferrementa de análise de sequência de cookies

4.3.10.3 Metodologia

O protocolo de comunicação HTTP não tem, por natureza, a propriedade de estabelecimento de sessão. Assim sendo, para que tal característica seja alcançada, criaram-se mecanismos de manutenção de estado/ses-são que atuam juntamente ao protocolo.

Um desses mecanismos é o *Cookie*, estrutura de dados responsável por conter informação de identificação de sessão. Essa mesma informação identificadora pode, variadamente, ser transmitida de outras formas – por URL ou por campos ocultos, ou *Hidden Fields*.

Quanto ao gerenciamento de sessão, portanto, cabe ao testador averiguar as seguintes diretivas:

- Os atributos dos cookies são adequadamente estabelecidos?
- É possível inferir o procedimento de geração dos identificadores de sessão usando engenharia reversa?
- Cookies de sessão apresentam tempo de expiração coerente?
- Os cookies são transportados por canal de comunicação seguro? Se sim, necessariamente?

4.3.10.4 Execução

Para a execução deste teste, partiremos da análise da resposta HTTP ilustrada na figura 8. Ela corresponde a resposta HTTP que define um valor de cookie para a sessão de usuário, através da diretiva *Set-Cookie*.

A distribuição dos valores de *cookies*, relacionados à variável PHPSESSID, já foi discutida neste trabalho; vide seção 4.3.5. Alguns outros campos, como *Expires* e Cache-Control, também foram, parcialmente, abordados no teste OWASP-AT-007, seção 4.3.7.

Entretanto, com o intuito de aprofundar ainda mais a análise do gerenciamento de sessão, trataremos das perguntas apresentadas na metodologia do teste atual, mas que não foram ainda respondidas. São elas:

Cookies de sessão apresentam tempo de expiração coerente?

Um cookie é atribuído pelo SGCI sempre que outro valor prévio não é encontrado no cache do navegador. Ou, de outro modo, quando ocorre o *logout* na aplicação. Assim sendo, durante o intervalo de validade do *cookie*, o mesmo pode ser reutilizado por um atacante, em um procedimento conhecido como *cookie replay*.

No ataque de replicação de *cookies*, um valor válido de sessão é empregado em requisições emitidas pelo atacante. Em consequência disso, o atacante assume a identidade do perfil de usuário que está, simultaneamente, associado àquele dado *cookie*.

Imaginemos uma situação em que um agente de registro esqueceu seu navegador aberto, com uma sessão do SGCI ativa. Enquanto o navegador permanecer aberto, o cookie associado à sessão será válido, o que pode representar uma janela de insegurança muito grande, ou muito pequena.

Idealmente, recomenda-se ajustar um tempo de expiração suficiente à operação adequada da aplicação; considerar-se-ia, neste caso, o comportamento do usuário no SGCI de acordo com as suas funcionalidades.

Os cookies são transportados por canal de comunicação seguro? Se sim, necessariamente?

Na figura 8, observamos a ausência da flag *Secure*, o que não assegura a transmissão do *cookie*, via de regra, por canal de comunicação seguro; consequentemente, um intruso entreposto na comunicação entre vítima e SGCI poderia capturar este dado.

```
HTTP/1.1 302 Found
Date: Wed, 21 Nov 2012 19:11:57 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.2
Set-Cookie: PHPSESSID=7ra6a3ho49eq5v514fmi631bg5; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Location: /auth/auth/authentication
Vary: Accept-Encoding
X-Content-Encoding: gzip
Content-length: 20
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

Figura 8: Resposta HTTP com diretiva Set-Cookie

4.3.11 Testing for Cookies Attributes (OWASP-SM-002)

4.3.11.1 Objetivos

- Analisar atributos associados aos cookies

4.3.11.2 Pré-requisitos

- Ferramenta de proxy (WebScarab, BurpSuite, etc)

4.3.11.3 Metodologia

Cookies são estruturas concebidamente utilizadas para a manutenção de estado do usuário em aplicações Web dinâmicas. Os critérios pelos quais são manipulados pela aplicação são definidos com base em atributos; acerca destes, alguns relacionam-se com aspectos de segurança que devem ser resguardados mediante o trato dessas estruturas, os *cookies*.

São listados os seguintes atributos necessários ao preenchimento de uma condição segura de gerenciamento de sessão do usuário:

- *Secure*: quando ativado, garante que o *cookie* será estritamente transportado por canal de comunicação seguro (tipicamente SSL/TLS);

- *HttpOnly*: se assinalado, impede que código cliente acesse o *cookie*. Recomenda-se que seja sempre utilizado, ainda que, atualmente, nem todos os navegadores o suporte;

- *Domain*: restringe o domínio, e subdomínios, autorizados a manipular o *cookie*;

- *Path*: normalmente utilizado conjuntamente ao atributo *domain*, este atributo designa o caminho de URL para o qual o cookie é válido;

- *Expires*: define tempo de expiração para cookies persistentes. Caso não seja definido valor algum, o *cookie* passa a expirar diante do término da sessão de navegação.

O teste de atributos de *cookie*, por conseguinte, dá-se a partir da análise de respostas HTTP com o cabeçalho *Set-Cookie* presente. Para essas respostas, verifica-se se há, nelas, a presença ou não dos atributos logo acima descritos.

Para os casos em que se mostram presentes, os atributos têm, do ponto de vista de segurança, sua composição de valores avaliada; quando da ausência daqueles, o testador infere os impactos dessa carência.

4.3.11.4 Execução

Ainda debruçados sobre a figura 8, nota-se a presença apenas dos atributos *Expires*, já discutido na seção do teste OWASP-AT-007, e *Path*.

Path, na forma em que foi definido, não restringe aplicações hospedadas na raiz da aplicação de acessar identificadores de sessão. Este atributo, em conjunto com *Domain*, deve limitar o perímetro a partir do qual *cookies* podem ser acessados.

Quanto aos demais atributos relevantes a este teste, destaca-se que não foram encontrados *Secure*, *HttpOnly* e *Domain*.

A não utilização de *Secure* permite que o *cookie* seja transmitido em conexões não seguras. Caso o *cookie* carregue consigo informações sensíveis em claro, ou mesmo em um padrão sobre o qual a inferência é viável, há a possibilidade de comprometimento da sessão do usuário.

Já a ausência de *HttpOnly* torna possível ao atacante acessar os *cookies* atráves de código cliente, como Javascript.

Finalmente, a carência do atributo *Domain* permite que outros domínios (potencialmente vulneráveis), os quais estejam na na raiz da aplicação '/', recebam os *cookies* do navegador.

4.3.12 Testing for CSRF (OWASP-SM-005)

4.3.12.1 Objetivos

- Carregar ou submeter informações à aplicação em nome de usuários válidos

4.3.12.2 Pré-requisitos

- Ferramenta de proxy (WebScarab, BurpSuite, etc)
- Ferramentas para teste de CSRF (*CSRF Tester*, *Cross Site Requester*, etc)

4.3.12.3 Metodologia

Ataques de CSRF (*Cross Site Request Forgery*) compreendem a realização, em uma aplicação web, de ações em detrimento de um usuário legítimo.

Consequentemente, para que o cenário de ataque se estabeleça, o usuário deve estar autenticado à aplicação. Quanto maiores os privilégios associados ao perfil da vítima, maiores são os eventuais danos de uma investida.

No entanto, detalhando um pouco mais a descrição acima, o CSRF é consolidado perante estes aspectos:

- Quando um usuário se autentica em uma aplicação, os navegadores retornam um valor de identificação, como um *cookie*. Logo após, conquanto o usuário permaneça autenticado, o navegador enviará essa identificação em requisições futuras.

- Em determinados casos, a aplicação não utiliza valores de controle de sessão nas URLs. Isso abre precedente para uma descoberta de URLs válidas pelo atacante; seja pela análise de código ou, como seria em um teste *black-box*, percorrendo a aplicação buscando identificar formulários e URLs embarcadas em código HTML/JavaScript.

- Aplicações web inadequadamente, muitas vezes, contam apenas com informações conhecidas pelo navegador quando do gerenciamento da sessão do usuário. Eis que, se utilizados apenas *cookies*, ou mecanismos de autenticação HTTP – como *Basic Authentication* –, o primeiro aspecto acima ressaltado confirma-se.

Os três aspectos supracitados são essenciais para um ataque de CSRF. No entanto, o ataque adquire credibilidade e eficácia devido à existência de tags HTML específicas, cuja função pode ser aproveitada para redirecionar o usuário para componentes da aplicação. A *tag* HTML *img*, por exemplo, permite a inserção de uma URL que designa uma operação válida na aplicação; de outro modo, ainda para a *tag img*, é possível redirecionar o usuário para uma página controlada pelo atacante – que, posteriormente, irá solicitar a requisição forjada.

Há diversas formas pelas quais uma requisição pode ser submetida por um usuário. Para uma requisição GET, por exemplo, o usuário poderia emiti-la enquanto utiliza a aplicação web, quando a digita diretamente no navegador ou, então, a partir de um link externo à aplicação apontando para a URL. Eis que essas submissões têm sua fonte indistinguível pela aplicação. Disso deriva a necessidade de amarrar informações únicas, geradas por mecanismos arbitrários, às URLs de uma dada sessão. Um atacante não deve ser capaz de sintetizar URLs legítimas para aplicação.

Quanto aos métodos HTTP que envolvem o CSRF, vale ressaltar a facilidade de exploração com requisições GET. Porém, requisições POST não estão imunes ao ataque, já que podem ter sua geração automatizada por

código Javascript. O emprego do método POST, assim sendo, mitiga parcialmente o ataque de CSRF, ainda que não o solucione de modo integral.

Na prática, para o teste de CSRF, o testador pode utilizar ferramentas de proxy para capturar e identificar requisições HTTP. Em seguida, submeter essas requisições com valores de sessão de outros usuários, e verificar as eventuais alterações nos dados da aplicação. Além disso, existem ferramentas especializadas na geração de requisições voltadas ao CSRF, como o *CSRF Tester*, da OWASP.

4.3.12.4 Execução

O SGCI possui um mecanismo de proteção contra ataques de CSRF. A cada requisição enviada para a aplicação é associado um *token*, cujos valores são dinâmicos e de difícil previsibilidade. Exemplos de valores para *no_csrf* são apresentados na seção 4.3.4.

Assim sendo, induzir uma vítima a realizar uma ação no SGCI, aproveitando-se do fato dela estar autenticada, esbarra na robustez do *token no_csrf* e sua correta validação no lado servidor.

4.3.13 Testing for Path Traversal (OWASP-AZ-001)

4.3.13.1 Objetivos

- Contornar permissões de acesso a arquivo e diretórios

4.3.13.2 Pré-requisitos

- Ferrementa de varredura de aplicação Web, Spider/Crawler
- Ferramenta de proxy (WebScarab, BurpSuite, etc)
- UniScan (testes de falha de atravessamento de diretórios)

4.3.13.3 Metodologia

Tradicionalmente, servidores e aplicações web implementam mecanismos de autenticação para controlar o acesso a arquivos e recursos. Em geral, fazem-no pelo confinamento dos arquivos de um determinado usuário

em um ‘diretório raiz‘, ou ‘raiz do documento web‘, que é mapeado para um diretório físico no sistema de arquivos; os usuários têm que considerar esse diretório como base para a estrutura hierárquica da aplicação web.

Quanto ao esquema de privilégios, sua definição é feita, tradicionalmente, através de *Access Control Lists*, ou ACLs; essas listas de controle de acesso – em tradução livre a partir do inglês – identificam quais usuários ou grupos serão capazes de acessar, modificar ou executar um arquivo específico no servidor.

Tais mecanismos de permissão são projetados para impedir o acesso a arquivos confidenciais por parte de usuários mal-intencionados (por exemplo, o arquivo comum /etc/passwd, em uma plataforma Unix); ou, ainda, para evitar a execução não autorizada de comandos do sistema.

Dado o exposto acima, passemos a uma descrição do ataque proposto pelo teste.

O atravessamento de diretórios, ou *Path Traversal*, envolve essencialmente duas etapas. A primeira delas é a enumeração dos vetores de validação de entrada. São vetores dessa categoria aqueles que permitem o envio de arquivos pelo usuário, formulários HTML, consultas HTTP GET e POST, etc.

A segunda etapa do teste se refere a tentativa de acessar arquivos, e diretórios, partindo-se dos vetores elencados na primeira fase. Esse processo de verificação realiza-se de forma manual – tarefa custosa quando de uma aplicação web complexa ou de grande porte – ou, alternativamente, de modo automático – através de ferramentas como o Uniscan. Ele analisa diferentes possibilidades de acesso a arquivos e diretórios. Além disso, emprega variados modos de codificação que, em determinados casos, viabilizam contornar filtros existentes no lado servidor.

Ressalta-se, contudo, em relação a constatação manual desta categoria de vulnerabilidade, a necessidade de se levar em consideração a maneira como ocorre o percorrimento de diretórios e arquivos, que está associada intrinsecamente ao Sistema Operacional que executa a aplicação Web.

Não confundir ’/’, carácter tipicamente utilizado em sistemas Linux, com ‘\\‘ ou ‘Letra:\caminho\para\arquivo‘, caracteres usualmente vistos em sistemas Windows.

4.3.13.4 Execução

Na primeira etapa do teste manual, foram enumeradas as páginas que contêm a função de envio de arquivos. Quanto aos demais tipos de vetor de entrada, não houve sua detecção.

- Vetores de entrada

/caadmin/relationships/importtrustedrelationship

/caoper/requests/importrequest

/caoper/revocations/importrequest

A função de envio de arquivos, no SGCI, está vinculada a uma requisição HTTP semelhante à ilustrada na figura 9. Atentemo-nos, para efeitos do teste de atravessamento, ao seguinte campo do *payload* da requisição:

```
Content-Disposition: form-data; name="relationshipFile";
filename="" Content-Type: application/octet-stream
```

Os parâmetros ‘name‘ e ‘filename‘ são pontos fortuitos ao teste de percorrimento da estrutura de diretórios. No entanto, embora fornecidas variações de caminhos – obedecendo à sintaxe do Unix para tal, ’/’ –, a aplicação SGCI não efetivou o ataque. Seja em virtude das restrições impostas pelo sistema operacional, ou por simplesmente não reconhecer, ou validar, o comando passado.

O mesmo teste foi efetuado para as três páginas, enumeradas como vetores de entrada, citadas. Em nenhuma tentativa, no entanto, obteve-se confirmação da vulnerabilidade de *Path Traversal*.

```
POST https://pentest-sgci.com:443/caadmin/relationships/importtrustedrelationship HTTP/1.1
Host: pentest-sgci.com
Referer: https://pentest-sgci.com/caadmin/relationships/importtrustedrelationship
Cookie: PHPSESSID=hrjdh0733kk3kp17ahk6f231o5
Content-Type: multipart/form-data; boundary---------------------------198482581117230965449232068
Content-length: 854

-----------------------------198482581117230965449232068
Content-Disposition: form-data; name="no_csrf"

029b058a4833ef7f15b9877a68a41f73
-----------------------------198482581117230965449232068
Content-Disposition: form-data; name="options"

Non registered entity
-----------------------------198482581117230965449232068
Content-Disposition: form-data; name="trustedEntity"

7
-----------------------------198482581117230965449232068
Content-Disposition: form-data; name="MAX_FILE_SIZE"

2097152
-----------------------------198482581117230965449232068
Content-Disposition: form-data; name="relationshipFile"; filename=""
Content-Type: application/octet-stream

-----------------------------198482581117230965449232068
Content-Disposition: form-data; name="submit"

-----------------------------198482581117230965449232068--
```

Figura 9: Requisição HTTP: importação de relacionamento de confiança

A tentativa de ataque automatizada, através do Uniscan, também fracassou. Isso devido a incapacidade de acesso a zonas restritas da aplicação, uma vez que o Uniscan não é capaz de nela autenticar-se.

4.4 RESULTADOS

A tabela abaixo descreve os treze testes realizados, a confirmação de vulnerabilidades a eles associadas, e, quando são estas confirmadas, as contramedidas suficientes para sua mitigação.

Teste	Vulnerável	Descrição	Contramedida
OWASP-IG-003		O teste de identificação de pontos de entrada contempla a enumeração da aplicação; desse modo, não remete ao dignóstico de vulnerabilidades.	
OWASP-IG-004	X	Foram identificados o sistema operacional e o servidor web subjacentes ao SGCI.	Alterar a assinatura retornada pelos serviços, de maneira a camuflar as versões e rótulos originais.
OWASP-AT-002		Não foi possível enumerar usuários válidos ou inválidos no SGCI.	
OWASP-AT-004		A presença do *token no_csrf* impediu a realização do ataque de força-bruta, já que inviabiliza a automatização das tentativas.	
OWASP-AT-005		Os quatro procedimentos que compõe este teste não revelaram vulnerabilidades; ou seja, há robustez no esquema de autenticação do SGCI.	
OWASP-AT-006		Não existe função de lembrete de senha. Além disso, quanto à redefinição de senhas, tal processo ocorre internamente à aplicação (sem uso de e-mail, por exemplo), mediante a utilização de HTTPS.	
OWASP-AT-007		Após o logout, não é possível acessar funções restritas da aplicação. Com respeito à expiração da sessão, ela é corretamente realizada pelo uso da diretiva *Expires*.	
OWASP-DV-001		Não foram diagnosticados pontos de reflexão de código malicioso fornecido pelo usuário.	
OWASP-DV-002	X	Foram detectados diversos	Inserção de filtros

		pontos de injeção de código malicioso, em que se observou sua persistência e posterior recuperação. A gravidade desta falha abrange diversos perfis do SGCI.	no lado servidor, tanto no momento da persistência como da recuperação do dado; assim como realizar o *escaping* dos caracteres indesejados nas entradas.
OWASP-SM-001	X	O tempo de expiração, através de *Expires*, não é definido, o que pode acarretar em uma janela de tempo muito grande para a execução de um ataque de *cookie replay;* constatou-se, também, a ausência da *flag Secure*.	Definir adequadamente o tempo de expiração dos cookies, e acrescentar a *flag Secure* às requisições que os contêm.
OWASP-SM-002	X	*Flag Path* incorretamente definida; ausência das flags *Secure*, *HttpOnly* e *Domain*.	Ajuste da *flag Path*, e acréscimo daquelas faltantes.
OWASP-SM-005		A existência do *token no_csrf* protege a aplicação de ataques de CSRF.	
OWASP-AZ-001		Não foi possível atravessar diretórios ou incluir arquivos remotos nos pontos de entrada identificados.	

Com base na série de testes propostos pela metodologia OWASP, trataremos agora da definição da superfície de ataque do SGCI. Para isso, será empregado um conjunto de métricas de segurança operacional, proposto pela metodologia OSSTMM: RAV, ou *Risk Assessment Value*.

O conceito de RAV foi detalhado na seção 3.1.3.1.

Para automatização do cálculo, foi utilizada uma planilha chamada *rav calculator*, oferecida pela ISECOM (ISECOM, 2012). Os resultados estão ilustrados na figura 10.

Attack Surface Security Metrics

OSSTMM version 3.0

Fill in the white number fields for OPSEC, Controls, and Limitations with the results of the security test. Refer to OSSTMM 3 (www.osstmm.org) for more information.

OPSEC

Visibility	1
Access	1
Trust	0
Total (Porosity)	2

CONTROLS

Class A		Missing
Authentication	1	1
Indemnification	0	2
Resilience	1	1
Subjugation	1	1
Continuity	0	2
Total Class A	3	7

Class B		Missing
Non-Repudiation	1	1
Confidentiality	1	1
Privacy	1	1
Integrity	0	2
Alarm	1	1
Total Class B	4	6

		True Missing
All Controls Total	7	13
Whole Coverage	35,00%	65,00%

LIMITATIONS

		Item Value	Total Value
Vulnerabilities	5	7,500000	37,500000
Weaknesses	2	4,500000	9,000000
Concerns	2	4,000000	8,000000
Exposures	2	5,150000	10,300000
Anomalies	3	4,500000	13,500000
Total # Limitations	14		78,3000

ISECOM
INSTITUTE FOR SECURITY AND OPEN METHODOLOGIES

OPSEC	5,304712
True Controls	3,427157
Full Controls	3,427157
True Coverage A	30,00%
True Coverage B	40,00%
Total True Coverage	35,00%
Limitations	15,161813
Security Δ	-17,04
True Protection	82,96

OSSTMM
Open Source Security Testing Methodology Manual
www.osstmm.org

Actual Security: 83,06 ravs

Figura 10: RAV score calculado para a aplicação SGCI

5 CONSIDERAÇÕES FINAIS

Aplicações web estão cada vez mais presentes na Internet. Quando relacionadas a soluções de ICP, elas podem amplificar o grau de exposição da cadeia de certificação e seus elementos envolvidos.

Com base nisso, buscou-se selecionar um conjunto de metodologias de segurança em aplicações web. Portanto, no capítulo 2, foram apresentadas e analisadas as metodologias: OSSTMM, ISSAF, OWASP e WASC.

Apoiando-se em sua credibilidade e afirmação internacional, foi escolhido o projeto OWASP para os testes de vulnerabilidade. O documento *OWASP Testing Guide v3* (OWASP, 2008) serviu como base para treze testes, ramificados em ataques específicos, como: gerenciamento de sessão, filtragem de dados fornecidos pelo usuário, permissão de acesso a diretórios e arquivos, robustez do esquema de autenticação, replicação de *cookies*, etc.

Observou-se um mau manuseio dos *cookies*, pelo SGCI; não lhes são atribuídas as *flags Secure*, *HttpOnly* e *Domain*.

A vulnerabilidade mais crítica identificada nesta etapa foi *Stored XSS*. Suas condições necessárias foram detectadas em diversos pontos do SGCI. Se explorados esses vetores de ataque, parte da cadeia de certificação pode ser comprometida por um atacante. O impacto das falhas de XSS armazenado é severo, e medidas tais quais filtros – preferivelmente baseados em *whitelist* – no lado servidor devem ser implementados para mitigar a vulnerabilidade.

Finda a etapa de testes, recorreu-se à abordagem de métrica de segurança computacional sugerida pelo OSSTMM v3 (ISECOM, 2010). O rav (*risk assessment value*) serviu como constante quantificadora da superfície de ataque da aplica-ção SGCI. Utilizou-se, em seguida, a planilha *rav calculator* para a auto-matização do cômputo do *rav socre*.

A aplicação obteve 83,06 rav.

5.1 TRABALHOS FUTUROS

Não há uma forma geral e definitiva para se avaliar a segurança de uma aplicação web. Desse modo, a conjunção de um subconjunto de metodologias, e a execução de um teste de intrusão com base nelas, não assegura todos os aspectos da aplicação. É sugerido, desse modo, a realização de outros tipos de teste de auditoria.

Uma abordagem complementar ao teste de intrusão realizado seria a revisão de código. Ter-se-ia, a partir de um código revisado, averiguado as possíveis fraquezas da aplicação sob um segunda perspectiva, mais interna

(testes caixa branca).

Ainda sobre trabalhos futuros, os treze testes selecionados do projeto OWASP não esgotam todas as funcionalidades da aplicação SGCI. A injeção de LDAP, por exemplo, não foi testada. Além disso, à medida que uma determinada aplicação web evolui, devem evoluir também seus controles de segurança. Para a avaliação da eficácia, ou ausência de determinadas proteções, de tais controles, assim sendo, são necessários novos testes de segurança.

ANEXO A – Artigo

Comparação e especialização de metodologias de segurança em Aplicações Web para o contexto de sistemas de ICP

Lucas Vinícius da Rosa[1]

[1]Laboratório de Segurança em Computação (LabSEC) – Universidade Federal de Santa Catarina (UFSC) – Departamento de Informática e Estatística (INE) – Florianópolis – SC – Brasil

lvrosa@inf.ufsc.br

Abstract. *Web applications are increasingly present in the Internet. Either by providing enterprise environment with specific solutions or by offering services to the end user. Thus it turns out necessary to assess security comprehensively but in detailed form. This assessment is supported by determined penetration testing methodologies that underlie the procedures of tests, such as: OSSTMM, ISSAF, OWASP and WASC. By analyzing them, it will be selected one or more methodologies to execute a penetration test on SGCI web application. The penetration test result will be registered on a report for future analysis and mitigation of the raised risks.*

Resumo. *Aplicações Web estão cada vez mais presentes na Internet. Seja fornecendo soluções específicas ao ambiente corporativo, ou no modelo de oferta de serviços ao usuário final. Dessa forma, faz-se necessário avaliar sua segurança de forma abrangente, contudo detalhada. Essa avaliação, por sua vez, tem como suporte determinadas metodologias de testes de intrusão, que fundamentam os procedimentos de testes, sendo aqui abrodadas as seguintes: OSSTMM, ISSAF, OWASP e WASC. A partir da análise destas, será eleita uma ou mais metodologias para realizar um teste de intrusão na aplicação SGCI. O resultado da execução do teste de intrusão será registrado em um relatório, para posterior análise e mitigação dos riscos levantados.*

1. Introdução

Com a popularização da Internet, ocorrida a partir da segunda metade de 1990, houve, gradualmente, a migração do ambiente corporativo e seus ativos para a conjuntura virtual. Consequentemente, conforme os anos passavam e a rede mundial de computadores evoluía, as empresas curvavam-se mais e mais aos benefícios da digitalização de seus processos operacionais, assim como o armazenamento de dados sensíveis em localizações remotas, porém acessíveis através da Web.

Desse modo, as aplicações Web tornaram-se onipresentes na Internet, trazendo, juntamente com a ubiquidade dos dados e a melhoria da eficiência no acesso, questões

frágeis no tocante à sua segurança. Uma vez substituídos, ou adicionados, novos vetores de ataque -- aqueles relacionados às aplicações Web modernas --, observou-se uma migração do interesse do atacante, que passou a voltar sua atenção às falhas em sistemas baseados na Web.

O aumento da oferta de aplicações Web, portanto, alavancou a quantidade de ataques voltados a essa área. Em virtude da preocupação que adorna o tema, criaram-se diversas iniciativas e/ou projetos cujo objetivo é, essencialmente, compor metodologias de endereçamento das questões de segurança de uma aplicação Web. Neste trabalho, são atendidas as seguintes metodologias: OSSTMM, ISSAF, OWASP e WASC.

Levando-se em consideração os requisitos do software SGCI (Software de Gerenciamento do Ciclo de Vida de Certificados da ICPEdu), serão destacados, de uma metodologia ou a partir de uma composição desta com outra, os testes mais aderentes ao domínio em que se insere a aplicação.

O teste de intrusão ratifica na prática, dessa forma, a descoberta e eventual exploração das fraquezas previstas pela metodologia escolhida; permite diagnosticar regiões de acesso não autorizado, vazamentos de dados, violação do esquema de privilégios, etc. Tal diagnóstico é registrado, conforme os testes são executados, em um relatório. Quando constatadas vulnerabilidades, serão fornecidas suas estratégias de mitigação.

2. Metodologia

Inicialmente, foram levantadas as metodologias atuais voltadas ao campo de segurança em aplicações Web. Após tal levantamento, selecionou-se um subconjunto destas para estudo aprofundado: OSSTMM, ISSAF, OWASP e WASC.

Uma vez individualmente estudadas, cada metodologia teve seus aspectos contrastados com os requisitos específicos encontrados em aplicações Web que utilizam ICPs. Tal análise contrastada é complementar às características básicas comumente encontradas em aplicações Web tradicionais, como gerenciamento de sessão do usuário, esquemas de autenticação e autori-zação, tratamento de dados oriundos do usuário, robustez de diretórios de armazenamento, etc.

O estudo individual, posteriomente estendido para uma análise coletiva das metodologias, fomentou um processo de eleição da abordagem mais adequada de teste de intrusão contra o sistema SGCI.

A realização do teste de intrusão culminou na composição de um relatório das vulnerabilidades testadas, classificando-as quanto a viabilidade de exploração e, quando confirmadas, oferecendo estratégias de mitigação correspondentes.

3. SGCI

O acrônimo SGCI significa Sistema de Gerenciamento de Certificados Digitais ICPEdu (Infra-estrutura de Chaves Públicas para Ensino e Pesquisa). Ele contempla as características e as operações tradicionais de SGCs, tais como criação e operação de ACs e ARs, emissão e revogação de certificados digitais, e emissão de LCRs.

As principais características do sistema SGCI são [LabSEC, 2012]:

- Criação e gerenciamento de ACs e Ars
- Utilização de entidades online e offline
- Utilização de ACs de resposta automática
- Utilização de ACs com e sem AR
- Suporte a HSM (*Hardware Security Module*)
- Suporte aos algoritmos ECDSA e RSA
- Suporte aos algoritmos de hash SHA-1 e SHA-2
- Suporte a um subconjunto do CMP (RFC4210 e RFC4211)
- Instalação simples (via apt em sistemas Debian)
- Código internacionalizado

Em sua versão mais recente, v2.0.0, o SGCI oferece suporte a dispostivos HSM, como mesmo demonstra a lista de principais funções acima exibida.

O SGCI divide-se em cinco módulos, definidamente:

- **Criador**: responsável pela criação de entidades e usuários, assim como pelas configurações da aplicação. Neste módulo, ACs, ARs e usuários podem ser cadastrados ou removidos, além da síntese de vínculos entre usuários e entidades;
- **Administrador de AC**: responde pelo gerenciamento de ACs. Neste módulo, são realizadas as operações de cadastro e remoção de operadores, definição de modelos de certificados, gestão de relacionamentos de confiança e determinação do período de emissão de LCRs;
- **Administrador de AR**: responde pelo gerenciamento de ARs. Neste módulo, tomam corpo as funções de cadastro e remoção de operadores e, também, o gerenciamento de relacionamentos de confiança;
- **Operador de AC**: responsável pela operação de ACs. Neste módulo, é possível aprovar ou rejeitar requisições de certificado e de revogação. Caso uma requisição seja aprovada, há a possibilidade de emissão do certificado a ela respectivo. Além disso, o Operador de AC emite LCRs.
- **Operador de AR**: reponsável pela operação de ARs. Neste módulo, o operador pode aprovar ou rejeitar requisições de certificado e de revogação. No entanto, diferentemente do Operador de AC, ele não emite LCRs.

Diversas versões foram lançadas pelo projeto SGCI, desenvolvido no LabSEC -- Laboratório de Segurança em Computação, UFSC. A primeira, v1.3.0 beta1, foi divulgada em 16/12/2010; a última, 2.0.0 RC2, em 07/12/20-12.

4. Metodologias de segurança em Aplicações Web

4.1. OSSTMM

A OSSTMM (*Open Source Security Testing Methodology Manual*) [OSSTMM v3 2010] fornece uma metodologia para a realização de testes de segurança minuciosos, também referenciados como auditorias OSSTMM. Uma auditoria OSSTMM define-se como uma avaliação precisa da segurança em um nível operacional, isentando-se de considerações não factuais e evidências anedóticas; é essencialmente baseada em métodos científicos. Além disso, tem entre seus princípios a consistência e a possibilidade de repetição.

Sendo amparada pela ISECOM(*Institute for Security and Open Methodologies*), e um projeto *open source*, a metodologia fomenta a contribuição da comunidade de segurança da informação, objetivando sua melhoria constante no tocante a precisão, acionamento e eficiência dos testes de segurança.

4.2. ISSAF

O ISSAF, ou *Information System Security Assessment Framework*, é um *framework* estruturado, e profissionalmente revisto, que categoriza a avaliação de segurança de sistemas de informação em variados domínios. Ademais, são contempladas as especificidades da avaliação, ou critérios de teste, de cada um desses domínios [OISSG 2006].

Ainda segundo o ISSAF [OISSG 2006], constitui-se em objetivo primário do framework preencher os requisitos de avaliação de segurança de uma dada organização. Em um segundo momento, entretanto, ele pode ser empregado como uma referência perante outras necessidades de segurança da informação.

No tocante aos processos de segurança, inclui os fundamentos necessários a avaliação e fortalecimento da infra-estrutura de segurança, viabilizando a obtenção de um retrato completo das vulnerabilidades que possam existir.

4.3. OWASP

O *Open Web Application Security Project*, OWASP, é uma organização mundial 501(c)(3) sem fins lucrativos, projetada para o aprimoramento da segurança de software. Ao aumentar a visibilidade da segurança de aplicações, torna-se possível que, em escala mundial, indivíduos e organizações tomem decisões mais acertadas quanto aos verdadeiros riscos de segurança de software [OWASP 2001].

A fundação OWASP surgiu, online, em primeiro de Dezembro de 2001, estabelecendo-se como uma entidade sem fins lucrativos anos depois, em Abril de 2004. Desde então, tem ramificado sua atuação mundialmente, através de capítulos locais e inúmeros projetos.

Os capítulos locais ajudam a disseminar os conhecimentos agregados pelos projetos OWASP, fixando bases regionais em torno do assunto de segurança em aplicações web.

4.4. WASC

A metodologia WASC (*Web Application Security Consortium*) compreende procedimentos de identificação e classificação de ataques e fraquezas inerentes às aplicações Web. À semelhança do projeto OWASP, define-se como um padrão aberto de avaliação de segurança de aplicações.

5. Teste de intrusão

5.1. Delimitação do escopo

A finalidade preponderante do SGCI é a criação e gestão de uma ICP. Com isso, a delimitação do seu escopo, que passa pelas operações básicas descritas na seção 4, fundamenta-se, antes de mais nada, nas relações de confiança definidas entre ARs, ACs e usuário final.

Concebidamente, o SGCI é o coração de gerenciamento dos sensíveis relacionamentos de uma ICP. Dessa forma, na perspectiva das aplicações Web, faz-se necessária uma administração segura e adequada das sessões dos usuários do sistema. Um Operador de AR, por exemplo, não deve, em hipótese alguma, ser capaz de sequestrar a sessão de um Administrador de AC. Isso representaria uma comprometimento da cadeia de confiança, cuja integridade é prevista para o correto e seguro funcionamento de uma ICP. O atacante, neste cenário exemplificado, poderia emitir listas de certificados revogados de maneira ilegítima.

Visualizados os aspectos conceituais de ICP empregados pelo SGCI, será efetuada uma seleção de testes consoantes aos módulos que implementam essas particularidades. Ademais, far-se-á também uma extração de testes pertinentes aos componentes Web presentes no SGCI. A metodologia fornecedora das diretrizes de teste será a OWASP.

5.2. Planejamento

A partir das características comumente presentes em aplicações web, juntamente àquelas correspondentes às especificidades do SGCI, elaborou-se uma lista de testes OWASP, abaixo apresentada. Cada teste possui denominação de acordo com a seguinte forma: *OWASP-categoria de teste-numeração dentro da categoria*.

- OWASP-IG-003 - Testing: Identify application entry points
- OWASP-IG-004 - Testing for Web Application Fingerprint
- OWASP-AT-002 - Testing for user enumeration
- OWASP-AT-004 - Testing for Brute Force
- OWASP-AT-005 - Testing for Bypassing Authentication Schema
- OWASP-AT-006 - Testing for Vulnerable Remember Password and Pwd Reset

- OWASP-AT-007 - Testing for Logout and Browser Cache Management
- OWASP-DV-001 - Testing for Reflected Cross Site Scripting
- OWASP-DV-002 - Testing for Stored Cross Site Scripting
- OWASP-SM-001 - Testing for Session Management Schema
- OWASP-SM-002 - Testing for Cookies Attributes
- OWASP-SM-005 - Testing for CSRF
- OWASP-AZ-001 - Testing for Path Traversal

Neste trabalho, embora tenha sido feita pelo próprio testador, não será contemplada a implantação do SGCI. Concebidamente, testes *blackbox*, abordagem aqui adotada, concentram-se em ataques sob uma perspectiva externa.

5.3. Resultados

A tabela abaixo descreve os treze testes realizados, a confirmação de vulnerabilidades a eles associadas, e, quando são estas confirmadas, as contramedidas suficientes para sua mitigação.

Teste	Vulnerável	Descrição	Contramedida
OWASP-IG-003		O teste de identificação de pontos de entrada contempla a enumeração da aplicação; desse modo, não remete ao dignóstico de vulnerabilidades.	
OWASP-IG-004	X	Foram identificados o sistema operacional e o servidor web subjacentes ao SGCI.	Alterar a assinatura retornada pelos serviços, de maneira a camuflar as versões e rótulos originais.
OWASP-AT-002		Não foi possível enumerar usuários válidos ou inválidos no SGCI.	
OWASP-AT-004		A presença do *token no_csrf* impediu a realização do ataque de força-bruta, já que inviabiliza a automatização das	

		tentativas.	
OWASP-AT-005		Os quatro procedimentos que compõe este teste não revelaram vulnerabilidades; ou seja, há robustez no esquema de autenticação do SGCI.	
OWASP-AT-006		Não existe função de lembrete de senha. Além disso, quanto à redefinição de senhas, tal processo ocorre internamente à aplicação (sem uso de e-mail, por exemplo), mediante a utilização de HTTPS.	
OWASP-AT-007		Após o logout, não é possível acessar funções restritas da aplicação. Com respeito à expiração da sessão, ela é corretamente realizada pelo uso da diretiva *Expires*.	
OWASP-DV-001		Não foram diagnosticados pontos de reflexão de código malicioso fornecido pelo usuário.	
OWASP-DV-002	X	Foram detectados diversos pontos de injeção de código malicioso, em que se observou sua persistência e posterior recuperação. A gravidade desta falha abrange diversos perfis do SGCI.	Inserção de filtros no lado servidor, tanto no momento da persistência como da recuperação do dado; assim como realizar o *escaping* dos caracteres indesejados nas entradas.
OWASP-SM-001	X	O tempo de expiração, através de *Expires*, não é definido, o que pode acarretar em uma janela de tempo muito grande para a execução de um ataque de *cookie replay;* constatou-se, também, a ausência da *flag*	Definir adequadamente o tempo de expiração dos cookies, e acrescentar a *flag Secure* às requisições que os

		Secure.	contêm.
OWASP-SM-002	X	*Flag Path* incorretamente definida; ausência das flags *Secure*, *HttpOnly* e *Domain*.	Ajuste da *flag Path*, e acréscimo daquelas faltantes.
OWASP-SM-005		A existência do *token no_csrf* protege a aplicação de ataques de CSRF.	
OWASP-AZ-001		Não foi possível atravessar diretórios ou incluir arquivos remotos nos pontos de entrada identificados.	

Tabela 1. Testes de intrusão realizados no SGCI e seus resultados obtidos

6. Considerações finais

Aplicações web estão cada vez mais presentes na Internet. Quando relacionadas a soluções de ICP, elas podem amplificar o grau de exposição da cadeia de certificação e seus elementos envolvidos.

Com base nisso, buscou-se selecionar um conjunto de metodologias de segurança em aplicações web. Portanto, na seção 4, foram apresentadas e analisadas as metodologias: OSSTMM, ISSAF, OWASP e WASC.

Apoiando-se em sua credibilidade e afirmação internacional, foi escolhido o projeto OWASP para os testes de vulnerabilidade.

O documento *OWASP Testing Guide v3* [TestingGuide 2008] serviu como base para treze testes, ramificados em ataques específicos, como: gerenciamento de sessão, filtragem de dados fornecidos pelo usuário, permissão de acesso a diretórios e arquivos, robustez do esquema de autenticação, replicação de cookies, etc.

Observou-se um mau manuseio dos cookies, pelo SGCI; não lhes são atribuídas as *flags Secure*, *HttpOnly* e *Domain*.

A vulnerabilidade mais crítica identificada nesta etapa foi *Stored* XSS. Suas condições necessárias foram detectadas em diversos pontos do SGCI.

Se explorados esses vetores de ataque, parte da cadeia de certificação pode ser comprometida por um atacante. O impacto das falhas de XSS armazenado é severo, e medidas tais quais filtros -- preferivelmente baseados em *whitelist* (em que são aceitos apenas os caracteres desejados, e não o contrário) -- no lado servidor devem ser implementados para mitigar a vulnerabilidade.

Referências

Laboratório de Segurança em Computação LabSEC (2012) "Sistema de Gerenciamento de Certificados Digitais ICPEDU (SGCI)", https://projetos.labsec.ufsc.br/sgci/, Novembro.

ISECOM (2010) "Open Source Security Testing Methodology Manual v3", http://www.isecom.org/mirror/OSSTMM.3.pdf, Dezembro.

OISSG (2006) "Information Systems Security Assessment Framework (ISSAF) draft 0.2", http://www.oissg.org/files/issaf0.2.1.pdf, Abril.

OWASP (2001) "OWASP Project", http://www.owasp.org, Agosto.

OWASP (2008) "OWASP Testing Guide v3", OWASP Foundation.

REFERÊNCIAS BIBLIOGRÁFICAS

ALI, T. *BackTrack 4 : Assuring Security by Penetration Testing*. [S.l.]: Packt Publishing Ltd, 2011. 392 p.

CROSS, M. *Developer's Guide to Web Application Security*. [S.l.]: Syngress Publishing, 2006. 514 p.

GRAVES, K. *CEH : Certified Ethical Hacker Study Guide*. [S.l.]: Sybex, 2010. 439 p.

GROUP, N. W. *Request for Comments 2616*. 1999. Disponível em: <http://www.w3.org/Protocols/rfc2616/rfc2616.html>. Acesso em: 17/11/2012.

IMPERVA. *Imperva Web Application Attack Report*. Janeiro 2010. Disponível em: <http://www.imperva.com/download.asp?id=344>. Acesso em: 25/07/2012.

ISECOM. *Open Source Security Testing Methodology Manual v3*. Dezembro 2010. Disponível em: <http://www.isecom.org/mirror/OSSTMM.3.pdf>. Acesso em: 01/08/2012.

ISECOM. *Security Metrics - Attack Surface Metrics*. 2012. Disponível em: <http://www.isecom.org/research/ravs.html>. Acesso em: 22/11/2012.

KAPIDZIC, A. D. N. A certificate management system: Structure, functions and protocols. p. 1–2, 1995.

KENT, S. *Privacy Enhancement for Internet Electronic Mail - Part II Certificate-Based Key Management*. 1993. Disponível em: <http://www.ietf.org/rfc/rfc1422>. Acesso em: 15/08/2012.

KOHLER, R. B. J. J. G. *Sistema de Gerenciamento de Certificados*. 2007. Acesso em: 21/08/2012.

LABSEC, L. de Segurança em C. *Sistema de Gerenciamento de Certificados Digitais ICPEDU (SGCI)*. 2012. Disponível em: <https://projetos.labsec.ufsc.br/sgci/>. Acesso em: 15/08/2012.

OISSG. *Information Systems Security Assessment Framework (ISSAF) draft 0.2*. Abril 2006. Disponível em: <http://www.oissg.org/files/issaf0.2.1.pdf>. Acesso em: 06/08/2012.

OWASP. *OWASP Project*. 2001. Disponível em: <http://www.owasp.org>. Acesso em: 09/08/2012.

OWASP. *OWASP Testing Guide v3*. [S.l.], 2008.

OWASP. *OWASP Top Ten Project*. 2010. Disponível em: <http://owasptop10.googlecode.com/files/>. Acesso em: 13/08/2012.

POLK, R. H. T. *Planning for PKI - Best Practices Guide for Deploying Public Key Infrastructure*. [S.l.]: New York: Wiley Computer Publishing, 2001.

RNP. *Módulo de HW e Serviços Seguros em Redes de Computadores*. 2004. Acesso em: 17/08/2012.

ROOM, S. I. I. R. Red teaming: The art of ethical hacking. v. 1, p. 1–2, 2003.

SCHNEIER, B. *Applied Cryptography : Protocols, Algorithms, and Source Code in C*. [S.l.]: John Wiley and Sons, 1996.

SERVICE, U. I. R. *501(c)(3) organizations*. 2012. Disponível em: <http://www.irs.gov/charities/charitable/article/0,,id=96099,00.html>. Acesso em: 09/08/2012.

SILVéRIO, A. L. *Análise e implementação de um protocolo de gerenciamento de certificados*. 2011. Acesso em: 15/08/2012.

STUTTARD DAFYDD, M. *Web Application Hackers Handbook : Discovering and Exploiting Security Flaws*. [S.l.]: Wiley, 2008. 770 p.

WASC. *The WASC Threat Classification v2.0*. 2010. Disponível em: <http://projects.webappsec.org/w/page/13246978/ThreatAcesso em: 07/08/2012.