



CENTRO UNIVERSITÁRIO
INSTITUTO DE EDUCAÇÃO SUPERIOR DE BRASÍLIA - IESB

Allison Machado Gonçalves

Humberto Rocha Gonçalves Filho

Autenticação Delegada com Código de Barras Bidimensional

Brasília - DF, Brasil

31 de Outubro de 2014

Allison Machado Gonçalves
Humberto Rocha Gonçalves Filho

Autenticação Delegada com Código de Barras Bidimensional

Trabalho de Conclusão de Curso II, do
Centro Universitário Instituto de Educação
Superior de Brasília, DF.

Centro Universitário Instituto de Educação Superior de Brasília
Ciência da Computação

Orientador: Prof. MSc. Cristiano Lehrer

Brasília - DF, Brasil
31 de Outubro de 2014

Allison Machado Gonçalves
Humberto Rocha Gonçalves Filho
Autenticação Delegada com Código de Barras Bidimensional. – Brasília - DF,
Brasil, 31 de Outubro de 2014-
59 p.

Orientador: Prof. MSc. Cristiano Lehrer

TCC (Graduação) – Centro Universitário Instituto de Educação Superior de Brasília
Ciência da Computação , 31 de Outubro de 2014.
1. Sistema de autenticação. 2. Boas práticas de segurança. I. Cristiano Lehrer. II. IESB. III. Ciência da Computação.

Allison Machado Gonçalves
Humberto Rocha Gonçalves Filho

Autenticação Delegada com Código de Barras Bidimensional

Trabalho de Conclusão de Curso II, do
Centro Universitário Instituto de Educação
Superior de Brasília, DF.

Banca examinadora - aprovado por:

Centro Universitário Instituto de Educação
Superior de Brasília
Ciência da Computação , DF

Centro Universitário Instituto de Educação
Superior de Brasília
Ciência da Computação , DF

Centro Universitário Instituto de Educação
Superior de Brasília
Ciência da Computação , DF

Brasília - DF, Brasil
31 de Outubro de 2014

*Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.*

Agradecimentos

Agradecemos primeiramente ao professor orientador Prof. MSc. Cristiano Leh-
rer, que contribuiu com conhecimento e inspiração para o desenvolvimento teórico e
prático do trabalho. Agradecemos também ao professor Joel Guilherme da Silva Filho
por nos fazer perceber as principais diferenças entre o nosso trabalho e os projetos já
desenvolvidos no mercado atual.

Agradecimentos especiais são direcionados às nossas famílias por nos apoia-
rem durante toda a trajetória de graduação. Certamente todo o apoio e ajuda foram
importantes para chegarmos até aqui.

*O que adquire entendimento ama a sua alma;
o que cultiva a inteligência achará o bem.*

Provérbios 19:8

Resumo

O ato de validar ou autenticar credenciais em um sistema computacional é uma tarefa corriqueira na vida das pessoas que utilizam recursos computacionais dos quais se faz necessário o registro de usuário. O modelo de maior popularidade atualmente para realizar tal tarefa é o de validação de usuário e senha. Um modelo amplamente aceito devido a sua baixa curva de aprendizado e sua sólida bibliografia no que diz respeito a boas práticas de segurança. Porém, este modelo se torna vulnerável quando se faz uso de dispositivos de domínio público como ponto de acesso.

Este trabalho demonstra um modelo baseado no uso de código de barras bidimensional como intermediário no processo de autenticação como uma forma de restringir o acesso mal intencionado às credenciais do usuário e conclui apresentando uma implementação do mesmo.

Palavras-chaves: Autenticação, segurança, código de barras bidimensional.

Abstract

The act of validate or authenticate into a computer system is a common task on life of people who makes use of computational resources that needs user registration. The current most popular model to accomplish this task is the user and password validation. A model widely used due its low learning curve and solid bibliography on best practices of security. However, this model becomes vulnerable when it makes use of public devices as access point.

This work demonstrates a model based on the use of two-dimensional barcode being intermediate on the process of authentication as a way to restrict malicious access to the user credentials and finalize with its implementation.

Keywords: Authentication, security, two-dimensional barcode.

Listas de ilustrações

Figura 1 – Presença de keyloggers em softwares maliciosos	16
Figura 2 – Exemplo de URI	17
Figura 3 – Interação entre Clientes e Servidores	19
Figura 4 – Interação entre Clientes e Servidores a nível de Programação	21
Figura 5 – Exemplo de envio de mensagem por meio da criptografia assimétrica	24
Figura 6 – Exemplificando o envio de assinatura digital	25
Figura 7 – SSL estendendo a API de Sockets	26
Figura 8 – Processo de autenticação utilizado atualmente nos sistemas Web .	30
Figura 9 – Exemplo de código de barras unidimensional	32
Figura 10 – Diversos modelos de códigos de barra bidimensionais	32
Figura 11 – Ilustração do <i>token</i> não autenticado e autenticado respectivamente	34
Figura 12 – Exemplo de formato do endereço de acesso a página de autenticação de <i>token</i>	34
Figura 13 – Processo de Autenticação Delegada	36
Figura 14 – Fazendo a requisição da página ao servidor	37
Figura 15 – Recebendo a página de autenticação via código de barras bidimensional	37
Figura 16 – Capturando os dados via código de barras bidimensional pelo dispositivo móvel	38
Figura 17 – Enviando credenciais para o servidor via dispositivo móvel	38
Figura 18 – Recebendo a resposta de confirmação de login	39
Figura 19 – Enviando o usuário e o identificador de sessão para vinculação . .	39
Figura 20 – Enviando a sessão autenticada para o navegador do usuário	40
Figura 21 – Padrões de detecção do código de barras bidimensional	44
Figura 22 – Tela do site de simulação	45
Figura 23 – Captura do código de barras por meio do aplicativo <i>QR Droid</i>	45
Figura 24 – Página de autenticação delegada	46
Figura 25 – Página de autenticação delegada com credenciais preenchidas . .	47
Figura 26 – Confirmação de autenticação	48
Figura 27 – Tela do site de simulação - antes de acessar a área restrita	49
Figura 28 – Tela do site de simulação - acesso à área restrita	49

Lista de tabelas

Tabela 1 – Impacto financeiro mundial causado por ataques de softwares maliciosos entre 1997-2006	15
Tabela 2 – Alguns protocolos da Internet em suas respectivas camadas	18
Tabela 3 – Sumário dos principais comandos HTTP	20
Tabela 4 – Exemplos de Cookies designados pelo site do jornal New York Times	22
Tabela 5 – Versões do padrão SSL	27
Tabela 6 – Dez maiores tipos de ameaças por número de violações e registros	28

Lista de abreviaturas e siglas

API	<i>Application Programming Interface</i>
DNS	<i>Domain Name System</i>
HTTP	<i>Hypertext Transfer Protocol</i>
IETF	<i>Internet Engineering Task Force</i>
IP	<i>Internet Protocol</i>
ORM	<i>Object-Relational Mapping</i>
QR Code	<i>Quick Response Code</i>
RFC	<i>Request for Comments</i>
SMS	<i>Short Message Service</i>
SQRL	<i>Secure Quick Reliable Login</i>
SSL	<i>Secure Sockets Layer</i>
TCP	<i>Transmission Control Protocol</i>
TLS	<i>Transport Layer Security</i>
URI	<i>Uniform Resource Identifier</i>
WEB	<i>World Wide Web</i>
WWW	<i>World Wide Web</i>

Sumário

1	INTRODUÇÃO	14
1.1	Motivação	14
1.2	Objetivos	15
1.2.1	Objetivo Geral	15
1.2.2	Objetivos Específicos	16
1.3	Organização do Trabalho	16
2	FUNDAMENTAÇÃO TEÓRICA	17
2.1	Request for Comments	17
2.2	A Arquitetura da Web	17
2.2.1	HTTP e a Web	18
2.2.2	Clientes e Servidores	19
2.2.3	Gerência de Estados	21
2.3	Privacidade e Segurança	22
2.3.1	Criptografia Assimétrica	23
2.3.2	Conexões TCP Seguras	26
2.3.3	Keystroke Loggers	28
2.4	Processo de autenticação	29
2.4.1	Vantagens e Desvantagens	31
2.5	Código de Barras Bidimensional	31
3	O MODELO DE AUTENTICAÇÃO DELEGADA	33
3.1	Pré-Requisitos do Modelo	33
3.2	Arquitetura	33
3.2.1	Token de Autenticação	33
3.2.2	Delegação da Autenticação	34
3.2.3	Código de Barras Bidimensional	35
3.2.4	Conexão Segura	35
3.2.5	Fluxo do Processo	35
3.3	Trabalhos Correlatos	40
3.3.1	Autenticação de Múltiplos Fatores	40
3.3.2	SQRL - Secure Quick Reliable Login	40
3.3.3	BB Code	41
4	IMPLEMENTAÇÃO E RESULTADOS	42
4.1	Tecnologias Utilizadas	42

4.1.1	A Linguagem de Programação Python	42
4.1.2	O Framework Django	42
4.1.3	O Sistema de Controle de Versão Git	43
4.1.4	QRCode	43
4.2	Simulação do Modelo	44
4.3	Discussão e Resultados	50
5	CONCLUSÃO E TRABALHOS FUTUROS	51
	Referências	52
6	APÊNDICE A - FUNÇÕES RESPONSÁVEIS PELA TELA INICIAL DE LOGIN E AUTENTICAÇÃO CONVENCIONAL	56
7	APÊNDICE B - FUNÇÕES RESPONSÁVEIS PELA AUTENTICA- ÇÃO DELEGADA	58

1 Introdução

1.1 Motivação

Cada vez mais as pessoas estão conectadas à internet, armazenando seus dados em servidores situados na “nuvem”, acessando redes sociais, realizando transações bancárias online e acessando a serviços na rede. Segundo Statista (2014), o número mundial de usuários da internet registrado em 2014 foi de 2,92 bilhões, 213 milhões de usuários a mais do que em 2013.

Com isso, a necessidade de autenticar-se em diversos serviços, de diversos dispositivos e em lugares diferentes é cada vez maior, e fazer isso de forma segura é um processo cada vez mais complexo para o usuário. Memorizar múltiplas senhas, garantir a segurança de cada máquina utilizada, lembrar de fechar a sessão toda vez que se ausentar da máquina são tarefas cansativas que encorajam os usuários a adotar maus comportamentos como o uso de senhas pequenas de fácil memorização, repetição de senhas para diversos serviços e o uso da opção de manter-se conectado ao sistema para não ter que entrar com suas credenciais novamente.

Além disso, outra ameaça significativa aos usuários de serviços Web é a perda de credenciais por programas maliciosos como *Spywares*. Existem diversas variações destes softwares maliciosos, entretanto seus objetivos são semelhantes. *Keyloggers*, ocultamente, capturam informações digitadas pelos usuários do sistema e os armazem no sistema de arquivos (SIKORSKI; HONIG, 2012). Eles têm a opção de criptografar as informações roubadas, incluindo a opção de enviar o arquivo para um destino por meio da Internet. Estes softwares podem ser sofisticados ao ponto de se esconderem do gerenciador de tarefas, configurando uma ameaça difícil de combater.

Estes softwares maliciosos vêm gerando grandes prejuízos financeiros para empresas que utilizam tecnologia em seu dia a dia (Tabela 1).

Ano	Custo em Dólar Americano
2006	\$13.3 Bilhões
2005	\$14.2 Bilhões
2004	\$17.5 Bilhões
2003	\$13.0 Bilhões
2002	\$11.1 Bilhões
2001	\$13.2 Bilhões
2000	\$17.1 Bilhões
1999	\$13.0 Bilhões
1998	\$6.1 Bilhões
1997	\$3.3 Bilhões

Tabela 1 – Impacto financeiro mundial causado por ataques de softwares maliciosos entre 1997-2006 (COMPUTER ECONOMICS, 2007)

A existência dos problemas citados motivou a elaboração deste trabalho, que busca alcançar um modelo de autenticação capaz de aliar segurança e praticidade para o usuário e, ao mesmo tempo, que possa coexistir com o modelo atual. Deseja-se tornar a sua adoção fácil, sem gerar um grande impacto de mudança em relação ao modelo vigente, encorajando sua implementação.

1.2 Objetivos

1.2.1 Objetivo Geral

A ideia principal deste modelo é possibilitar, através do uso de um código de barras bidimensional, a delegação do processo de autenticação da máquina alvo para o dispositivo móvel do usuário quando a máquina em questão não é confiável. Desta forma, as credenciais de acesso ao serviço Web são inseridas no dispositivo móvel, evitando o roubo de informações por programas maliciosos possivelmente escondidos na máquina alvo.

Segundo a empresa de soluções de segurança Symantec, 76% dos softwares maliciosos que ameaçam roubar informações confidenciais utilizam alguma versão de *keystroke logger* (Figura 1). Portanto, conforme será detalhado, esta metodologia é capaz de escapar desta espécie de programa malicioso.

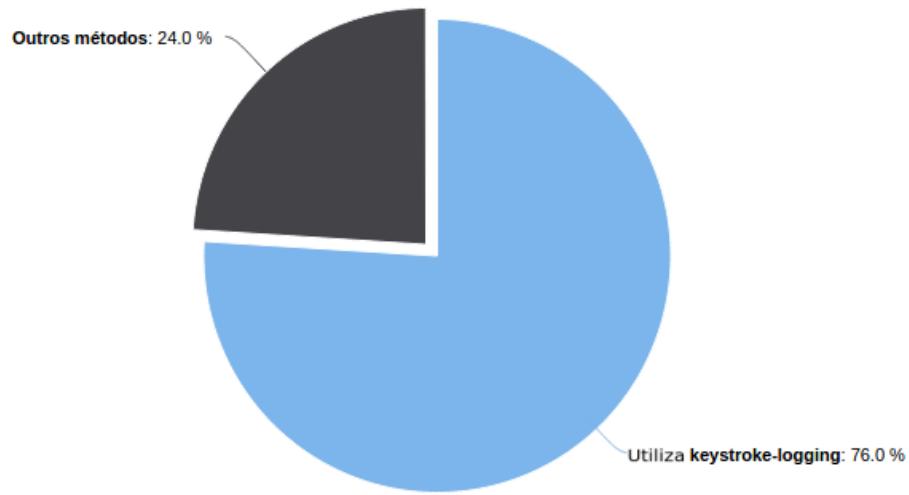


Figura 1 – Presença de keyloggers em softwares maliciosos (SYMANTEC, 2014)

1.2.2 Objetivos Específicos

- Apresentar uma arquitetura de autenticação segura, prática e que possa coexistir com o modelo vigente;
- Possibilitar que usuários possam se autenticar de forma segura em estações de trabalho não confiáveis;

1.3 Organização do Trabalho

Este trabalho está organizado nos seguintes capítulos descritos a seguir:

- O Capítulo dois apresenta a fundamentação teórica necessária para entendimento do trabalho. Neste Capítulo são abordados conceitos importantes como a Arquitetura Web, Protocolos Web e boas práticas de segurança usadas na Internet.
- O Capítulo três descreve o projeto deste trabalho. Para maior entendimento de sua estrutura e funcionamento.
- O capítulo quatro apresenta os aspectos relativos a implementação do modelo.
- O capítulo cinco conclui este trabalho apresentando seus resultados, seus pontos fortes e fracos.

2 Fundamentação Teórica

O objetivo deste capítulo é apresentar a fundamentação teórica necessária para o entendimento do estudo deste trabalho. Será apresentada uma breve descrição dos conceitos da *World Wide Web* relacionados ao trabalho, como uma visão geral da arquitetura Web, protocolos principais, características das conexões e segurança de redes.

2.1 Request for Comments

Uma das principais fontes no que se refere à padronização, implementação e evolução da arquitetura do maior conjunto interligado de redes do mundo, comumente denominado Internet, são os RFCs (*Request for Comments* - Pedido para Comentários). Os RFCs são documentos técnicos criados e/ou regulamentados pelo IETF (*Internet Engineering Task Force*), que é uma instituição formada por uma grande comunidade aberta de empresas, pesquisadores e estudiosos da área.

Os RFCs são escritos por engenheiros e cientistas da computação, que descrevem métodos, comportamentos, pesquisas e inovações aplicáveis a Internet e sistemas interconectados (IETF, 2014). Estes documentos serão extensivamente referenciados neste trabalho como fonte de pesquisa e/ou aprofundamento técnico.

2.2 A Arquitetura da Web

A Web ou WWW (World Wide Web - Grande Rede Mundial) é um dos serviços oferecidos pela Internet. Este serviço se baseia na disponibilização de recursos vinculados que podem ser encontrados por qualquer usuário conectado à Internet através de seu endereço (TANENBAUM, 2011).

O identificador de um recurso da Web é denominado URI (Uniform Resource Identifier – Localizador Uniforme de Recurso) (IEFT, 2005) (Figura 2).

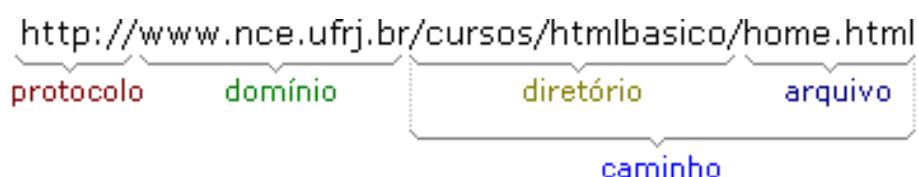


Figura 2 – Exemplo de URI (NCE/UFRJ, 2014)

A parte deste identificador conhecida como domínio (IETF, 1987a) é a responsável por referenciar uma máquina ou um conjunto de máquinas na rede que hospedam os recursos em questão. Este domínio é uma forma de facilitar o endereçamento na web, retirando do usuário a responsabilidade de saber o endereço de IP (*Internet Protocol*) (IETF, 1981a), que é o protocolo base da estrutura de comunicação em rede.

Quando um recurso é acessado por seu domínio, uma requisição ao DNS (*Domain Name System*) é feita pelo endereço IP do recurso. O DNS (IETF, 1987b) é um sistema responsável pela resolução dos nomes de domínio para endereços IP.

2.2.1 HTTP e a Web

O HTTP (*Hypertext Transfer Protocol* - Protocolo de Transferência de Hipertexto) é definido no RFC 1945 e RFC 2616. O protocolo funciona por meio da troca de mensagens de texto que acontece entre os agentes Web (Seção 2.2) que implementam o protocolo. O protocolo HTTP define como os clientes Web requisitam objetos aos servidores e a forma como os servidores tratam e respondem às requisições. O HTTP usa o TCP (Transmission Control Protocol - Protocolo de controle de transmissão) como seu protocolo de transporte, portanto o HTTP não precisa se preocupar com perda de informação. A Tabela 2 apresenta as camadas do modelo TCP/IP juntamente com alguns de seus respectivos protocolos.

Camada	Protocolos
Aplicação	HTTP, SMTP, FTP, SSH, DNS
Transporte	TCP, UDP
Rede	IP (IPv4, IPv6) , ARP
Enlace	Ethernet

Tabela 2 – Alguns protocolos da Internet em suas respectivas camadas

O HTTP é um protocolo que permite a troca de arquivos de forma bidirecional. Na maioria dos casos, um servidor transfere uma cópia de um arquivo ao cliente após receber uma requisição. Entretanto também é possível que exista a transferência de um arquivo do cliente para o servidor. Neste caso é necessário que o servidor esteja preparado para este recebimento específico, e que ocorre por meio dos formulários Web.

O HTTP permite que os agentes negociem parâmetros que influenciam as características da comunicação. Por exemplo, é possível que navegadores e servidores negoiciem detalhes como a codificação de caracteres da mensagem, informem recursos que podem oferecer ou receber.

2.2.2 Clientes e Servidores

A arquitetura da Web baseia-se na arquitetura conhecida como Cliente e Servidor (Figura 3). A Web, portanto, é um Sistema Distribuído sobre a rede mundial de computadores onde um servidor executa processos capazes de servir requisições realizadas por clientes consumidores dos recursos disponíveis. Os clientes são aplicações com capacidade de realizar requisições na rede e recuperar sua resposta, disponibilizando-a para o usuário e/ou outra aplicação. O cliente mais popular dentre os usuários da internet é o navegador.

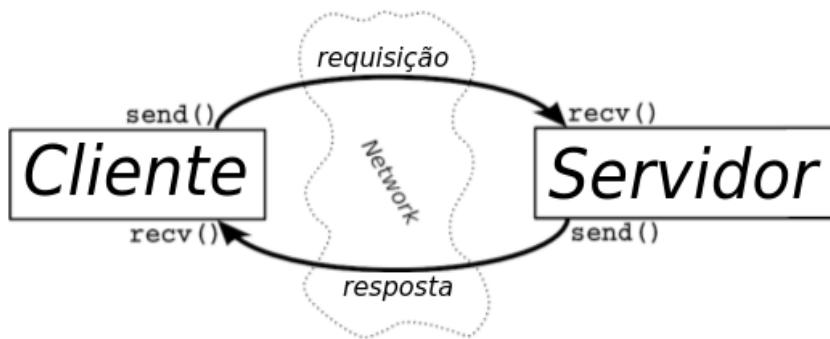


Figura 3 – Interação entre Clientes e Servidores (HALL, 2012)

Uma série de tarefas ordenadas ocorrem toda vez que um usuário deseja carregar uma página Web. Os passos para a obtenção de uma página por um Navegador ou cliente foram descritos por Tanenbaum (2011) da seguinte maneira:¹

- O usuário deseja obter a página <https://www.google.com.br>;
- O cliente navegador obtém a URI por meio de um *link* ou do texto fornecido pelo usuário;
- O cliente realiza uma busca em DNS para obter o IP de www.google.com.br (servidor);
- O cliente recebe um IP, por exemplo 156.106.192.32;
- O cliente estabelece uma conexão TCP (IETF, 1981b) com a porta 80, tipicamente usada pelo protocolo HTTP (Seção 2.2.1) do servidor 156.106.192.32;
- O cliente envia um comando HTTP, solicitando o arquivo index.html;
- O servidor www.google.com.br envia o arquivo index.html;
- O cliente apresenta o conteúdo de index.html estruturando a página de acordo com a linguagem de marcação utilizada.

O servidor Web a todo momento fica à espera de possíveis requisições (Tabela 3) de clientes em busca de páginas Web. Por padrão, o servidor espera por conexões na porta 80 e, depois que a conexão for estabelecida, espera por mensagens contendo requisições HTTP. Junto com a requisição, chega o nome do arquivo que o cliente procura e então o servidor retorna este arquivo em formato de resposta. O resumo dos principais comandos HTTP pode ser encontrado na Tabela a seguir.

Comando	Significado
GET	Solicita documento especificado na URI
HEAD	Solicita informação do documento especificado na URI
OPTIONS	Solicita informação sobre opções de comunicação
POST	Fornecer informações de entrada ao servidor
PUT	Armazena documento em uma URI específica
DELETE	Deleta documento localizado em uma URI específica
TRACE	<i>Loopback request message</i> - auxílio no diagnóstico de problemas

Tabela 3 – Sumário dos principais comandos HTTP (PETERSON; DAVI, 2011)

A execução de um servidor Web foi descrito de forma simplificada por Tanenbaum (2011) da seguinte maneira, onde o servidor Web:¹

- Aceita a conexão TCP do cliente;
- Obtém o nome do arquivo solicitado de acordo com as regras do protocolo HTTP;
- Obtém o arquivo do disco;
- Envia o arquivo para o cliente;
- Encerra a conexão TCP assim que o cliente fecha a instância do navegador.

A nível de programação os agentes Web normalmente utilizam a API (*Application Programming Interface* - Interface de Programação de Aplicações) de *Sockets*, que possibilita a entrada e saída de dados pela rede utilizando-se de descritores de arquivos. Clientes e Servidores têm um comportamento muito característico a nível de *Sockets* (Figura 4).

¹ Listado o funcionamento básico para o entendimento do processo. Existem funcionalidades adicionais que melhoraram a experiência do usuário neste sistema, entretanto não são o foco deste trabalho.

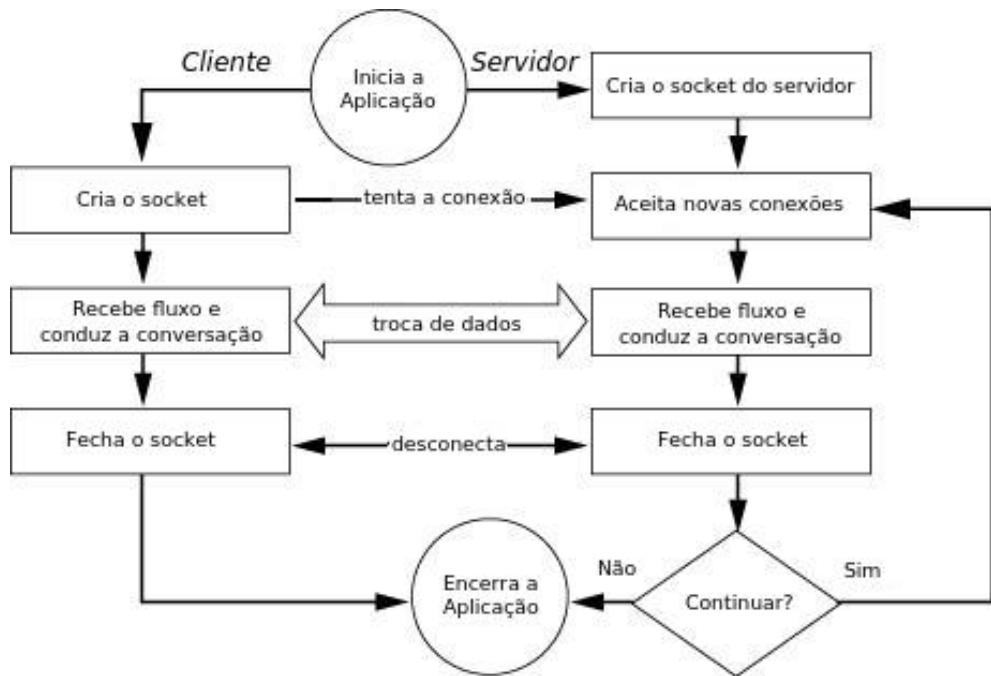


Figura 4 – Interação entre Clientes e Servidores a nível de Programação

2.2.3 Gerência de Estados

Analizando o comportamento típico dos agentes Web (clientes e servidores) observa-se que o sistema não possui estados, ou seja, a princípio um servidor pode esquecer da existência de um cliente específico após o encerramento da conexão. Entretanto, um servidor pode precisar distinguir entre requisições de clientes que estejam cadastrados no sistema que está servindo de outras requisições, para poder dar um tratamento especial a clientes cadastrados. Para resolver este tipo de problema é necessário que exista o conceito de estado, ou sessão, no processo de comunicação.

Imagine que um usuário deseja customizar a aparência ou o comportamento de determinado site, ou então acessá-lo sem precisar digitar a senha todas as vezes que desejar abrir a página. Sem o conceito de estado, o usuário precisará realizar estas operações toda vez que acessar o site.

Para resolver os problemas citados no parágrafo acima foram criados os *cookies*. Os *cookies* são dados enviados pelo servidor a um cliente, como uma informação adicional contida nas respostas de requisições HTTP. O *cookie* é tratado como um arquivo pelo navegador e é armazenado em um diretório no sistema de arquivos.

É importante prestar atenção nos seguintes campos contidos em um *cookie*:

1. Domínio;
2. Conteúdo;

3. Data de validade;
4. Segurança.

Primeiramente um *cookie* informa sua origem por meio do campo domínio, e o cliente valida a autenticidade desta informação. *Cookies* carregam informações no formato “nome=valor”, por exemplo “IdUsuario=334522”, contidas no campo de Conteúdo. A Data de Validade informa a data de expiração do *cookie*, portanto um *cookie* é considerado válido até a data contida neste campo. O campo de segurança indica que o navegador só deve enviar este *cookie* a um servidor se a conexão for segura e criptografada (Seção 2.3). A Tabela 4 mostra os cookies enviados pelo site do jornal New York Times num experimento feito no dia 17 de Setembro de 2014.

Toda vez que um cliente Web vai realizar uma requisição por uma página, primeiramente ele verifica se existe um *cookie* armazenado que pertence ao domínio da página desejada. Caso encontre, os *cookies* são enviados juntamente com a mensagem de requisição ao servidor. Ao receber o *cookie*, o servidor pode interpretar o seu conteúdo da maneira que desejar.

Chave	Conteúdo	Domínio	Validade
RMID	007f0101686b542da7ed0003	.nytimes.com	10-02-2015
adxcs	-	.nytimes.com	-
adxcl	I*3a4de=54c9be4f:1	.nytimes.com	10-02-2015
NYT-wpAB	0002 1&0008 0	.nytimes.com	10-02-2015
NREUM	s=1412278254825&r=2603...	www.nytimes.com	-

Tabela 4 – Exemplos de Cookies designados pelo site do jornal New York Times, Cookie Checker (2014)

2.3 Privacidade e Segurança

Para um bom entendimento de como funcionam os processos de segurança aplicados nas redes de computadores e na Web é fundamental que se entenda o funcionamento de alguns de sistemas criptográficos e protocolos de segurança. Portanto, esta seção trata de assuntos relacionados à segurança de redes.

Um sistema de comunicação seguro é aquele capaz de garantir aos seus usuários confidencialidade, autenticação e integridade da mensagem. A confidencialidade garante que, ao enviar uma mensagem, o remetente tenha a certeza de que só o seu destinatário será capaz de ter acesso à mensagem. Portanto, se outra pessoa tiver acesso à mensagem, ela não será capaz de decifrar o seu conteúdo. Por meio da autenticação, os pares envolvidos na troca de mensagens têm a certeza de que estão

se comunicando com quem desejam, e não com um impostor. Se o sistema garante a integridade da mensagem, ao recebê-la, o usuário pode estar certo de que o seu conteúdo é idêntico ao enviado pelo remetente.

Estas funcionalidades devem fazer parte do sistema de **Autenticação Delegada com Código de Barras Bidimensional**, entretanto não precisam ser reimplementadas, já que existem implementações eficientes e acessíveis. Basta entender os conceitos para aplicá-los de forma correta neste trabalho.

2.3.1 Criptografia Assimétrica

A criptografia assimétrica, também conhecida como criptografia de chave pública, foi embasada na aplicação de funções matemáticas sobre a mensagem que se deseja cifrar, em vez da simples permutação de texto - que era a principal forma de implementação de algoritmos criptográficos anteriores.

A criptografia simétrica utiliza a mesma chave para cifrar e decifrar mensagens, enquanto a criptografia assimétrica utiliza chaves distintas para cifrar e decifrar mensagens. Determinar a chave de decifração utilizada sobre uma mensagem deve ser inviável num sistema assimétrico seguro, tendo-se apenas o conhecimento da chave de cifração e do algoritmo de cifração.

O desenvolvimento da criptografia de chave pública se deu na tentativa de resolver dois problemas importantes. O primeiro é a distribuição chaves por um meio não confiável entre duas entidades que nunca tiveram contato anterior. E o segundo é como garantir que os pares envolvidos na comunicação tenham a comprovação de que determinada mensagem foi enviada por determinada pessoa.

Logo, um sistema de criptografia de chave pública possui os seguintes elementos:

- Mensagem clara (ou texto plano);
- Mensagem cifrada;
- Algoritmo de cifração;
- Algoritmo de decifração;
- Chaves pública e privada - de forma que cada chave usada para cifração tem apenas uma usada para decifração da mensagem.

Segundo Stallings (2008), o processo de troca de mensagem secreta entre dois usuários deste sistema criptográfico (Figura 5) pode ser resumido da seguinte maneira:

1. Os usuários devem gerar um par de chaves;
2. Uma das chaves é mantida sobre sigilo - a chave privada;
3. A outra chave é disponibilizada de forma que qualquer usuário pode obtê-la - a chave pública;
4. Se o usuário **A** deseja enviar uma mensagem para o usuário **B**, o usuário **A** cifra a mensagem usando a chave pública de **B**, e envia a mensagem cifrada para **B**;
5. Neste momento, só é possível decifrar a mensagem por meio da chave privada de **B**;
6. Quando o usuário **B** recebe a mensagem, ele a decifra usando sua chave privada;
7. Não é possível que outro usuário decifre a mensagem pois somente **B** tem acesso a sua chave privada.

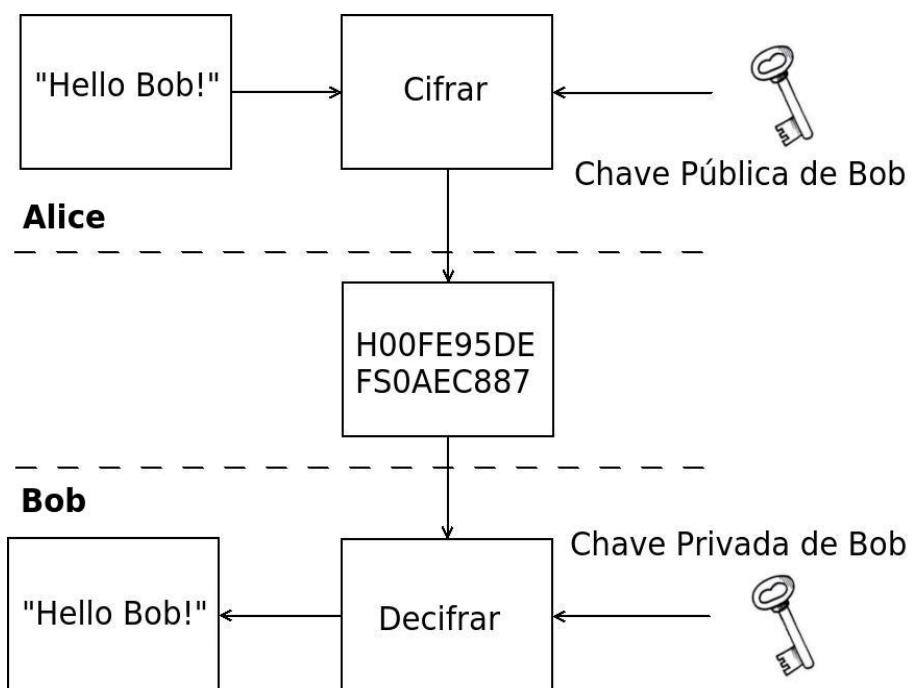


Figura 5 – Exemplo de envio de mensagem por meio da criptografia assimétrica

Observando esta sequência de passos, pode-se concluir que para o envio de mensagens assinadas digitalmente (Figura 6), não é conceitualmente necessário acrescentar novos elementos. Basta inverter a ordem de utilização das chaves neste processo para que um usuário destinatário de uma mensagem possa ter certeza de que uma mensagem foi enviada por determinado destinatário. Observe a pequena mudança no seguinte processo de comunicação:

1. O usuário **A** deseja enviar uma mensagem para **B** de forma que **B**, quando receber a mensagem, tenha certeza de que o remetente da mensagem foi o usuário **A**;
2. Enfim, o usuário **A** cifra a mensagem usando a sua chave privada e envia a mensagem cifrada para **B**.
3. Neste momento, só é possível decifrar a mensagem por meio da chave pública de **A**;
4. Quando o usuário **B** recebe a mensagem, ele a decifra usando a chave pública de **A**;
5. Caso a mensagem decifrada seja igual a mensagem em texto simples recebida, constata-se que a mensagem foi enviada por **A**.
6. Não é possível que outro usuário tenha enviado a mensagem pois somente **A** tem acesso a sua chave privada.

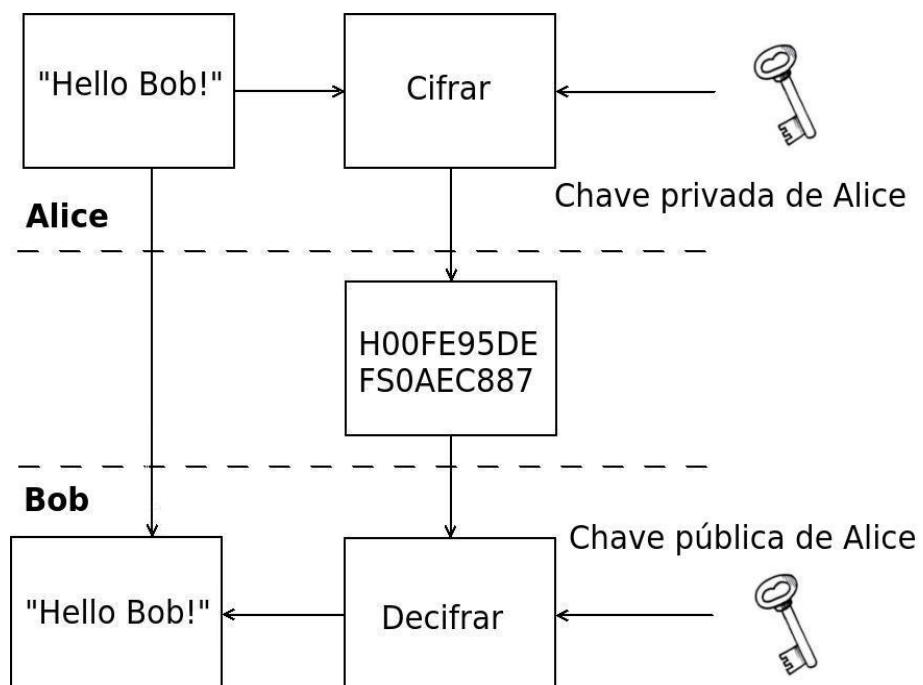


Figura 6 – Exemplificando o envio de assinatura digital

Contudo, é necessário certificar-se de que a chave pública que se está usando para decifrar a mensagem não é uma chave forjada. Para isto existem as Autoridades Certificadoras confiáveis, que atestam a legitimidade do serviço caso a chave pública seja compatível a determinada entidade. As Autoridades Certificadoras devem manter

registros de chaves e entidades, para poder responder à consultas e emitir certificados válidos.

Vale lembrar que se deve manter pares de chaves distintos para cada objetivo específico. Por exemplo, para garantir a identidade de quem está enviando a mensagem, utiliza-se um par de chaves; e para garantir que o conteúdo da mensagem somente será acessado por pessoas autorizadas, utiliza-se outro par de chaves. Utilizar o mesmo par de chaves para as duas funções facilita a criptoanálise sobre mensagens, enfraquecendo a segurança do sistema.

2.3.2 Conexões TCP Seguras

O SSL (Secure Sockets Layer) é um padrão de segurança de comunicação, que tem como objetivo estabelecer uma conexão onde as informações são cifradas. O padrão estende os serviços do protocolo TCP, adicionando a garantia de confidencialidade, autenticação e integridade de dados.

O SSL permite que informações confidenciais, como números de cartão de crédito, e credenciais de login sejam transmitidas de forma segura. Normalmente, os dados enviados entre clientes e servidores Web são textos simples, ou seja, vulneráveis a espionagem. Segundo Kurose e Ross (2010):

“O SSL estende a API de sockets, e quando uma aplicação deseja implementar a segurança fornecida pelo padrão, basta incluir as bibliotecas ou classes SSL.” (Figura 7)

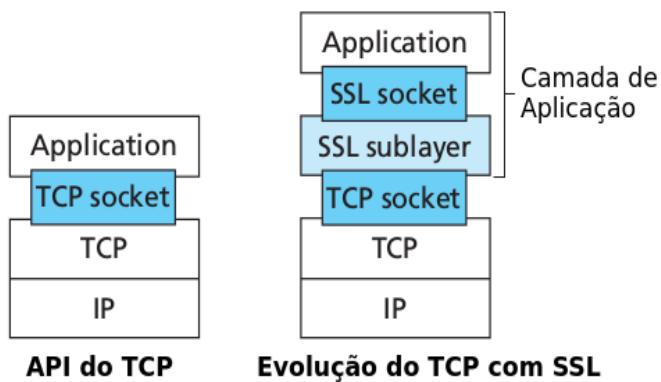


Figura 7 – SSL estendendo a API de Sockets

Os conceitos de Criptografia Assimétrica (Seção 2.3.1) são usados no padrão SSL. Quando um cliente faz uma requisição a um servidor, o servidor, na sua resposta, envia seu certificado digital, contendo a sua chave pública. O cliente pode ter

certeza da autenticidade do servidor se o certificado for constatado por uma Autoridade Certificadora. Em seguida, o cliente gera uma Chave Secreta (a qual deve ser válida somente nesta sessão) que será usada para cifrar as mensagens (e subsequentes chaves geradas) trocadas na comunicação entre os pares. O cliente por fim usa a chave pública do servidor, e envia a chave secreta da sessão. Neste momento, somente os dois possuem a chave da comunicação.

Como a cifragem assimétrica é mais complexa e dispendiosa do que a cifragem simétrica, a criptografia de chave pública não é usada para cifrar as mensagens da comunicação. Mas para gerar chaves secretas e simétricas, estas são responsáveis por cifrar as mensagens que as carregam.

Uma versão modificada do SSL versão 3, chamado *Transport Layer Security* (TLS), foi normatizada pelo IETF (2006) e vem sendo amplamente utilizada pelos servidores e clientes Web modernos. A Tabela 5 exibe a evolução do padrão SSL até o TSL.

Vers.	Fonte	Descrição
SSL v2.0	<i>Vendor Standard (from Netscape Corp.)</i>	Primeira implementação existente do protocolo SSL.
SSL v3.0	<i>Expired Internet Draft (from Netscape Corp.)</i>	Revisões para prevenir ataques de segurança específicos, adiciona cifras não RSA e suporte para cadeias de certificados.
TLS v1.0	<i>Proposed Internet Standard (from IETF)</i>	Revisão do SSL 3.0 atualizando a camada MAC para HMAC, padronizando ordem das mensagens e adicionando mensagens de alerta.
TLS v1.1	<i>Proposed Internet Standard (from IETF)</i>	Atualização do TLS 1.0 para adicionar proteção contra ataques <i>Cipher Block Chaining - (CBC)</i> .
TLS v1.2	<i>Proposed Internet Standard (from IETF)</i>	Atualização do TLS 1.2, adicionando incompatibilidade com SSL para que não haja negociações do tipo SSLv2.

Tabela 5 – Versões do padrão SSL (APACHE SOFTWARE FOUNDATION, 2014)

2.3.3 Keystroke Loggers

Keystroke Loggers, também conhecidos como *keyloggers*, são softwares maliciosos que tem a capacidade de interceptar a entrada de texto digitada pelos usuários do sistema de forma oculta, e armazená-los para obter informações sigilosas. Este software pode ser programado para armazenar os dados capturados na máquina comprometida, ou fazer parte de um ataque maior e enviar as informações diretamente pela rede para a máquina do invasor. Apesar de considerarmos neste trabalho apenas os softwares maliciosos, existem componentes de softwares legítimos que têm a capacidade de interceptar e armazenar a entrada de texto do sistema, porém não são ocultos e nem usados para roubar de informações.

Segundo (VERISON, 2012), no seu relatório de segurança que analisou 855 incidentes de crimes cibernéticos em 2011, os softwares maliciosos do tipo *keylogger/spyware* estão no topo da lista das **dez maiores ameaças** encontradas na análise das violações de segurança. A lista do relatório pode ser observada na Tabela 6. Observa-se que o roubo de informações por meio de *keyloggers/spywares* está presente em 48% das violações, e na terceira posição, como uma consequência do roubo de informações, está o uso das credenciais roubadas para adquirir acesso improprio.

Posição	Ameaça	Categoria	Violações
1	<i>Keylogger/Spyware</i>	Malware	48%
2	Exploração de credenciais dedutíveis	Hacking	44%
3	Uso de credenciais roubadas	Hacking	32%
4	Envio de informações para entidades externas	Malware	30%
5	Ataques de força bruta	Hacking	23%
6	<i>Backdoor</i>	Malware	20%
7	Servidores de comando e controle	Hacking	20%
8	Interferir em controles de segurança	Malware	18%
9	<i>Tampering</i> (acesso físico à máquina)	Físico	10%
10	Exploração de falta de credenciais	Hacking	5%

Tabela 6 – Dez maiores tipos de ameaças por número de violações e registros. Adaptado de (VERISON, 2012).

A detecção deste tipo de ameaça pode ser difícil, pois o objetivo deste tipo de *malware* é permanecer silencioso sendo o mais discreto possível. *Keyloggers* personalizados para determinados ataques podem ser ainda mais difíceis de identificar, sendo imperceptíveis a programas anti-vírus comuns.

2.4 Processo de autenticação

Para fins de identificação ou controle de acesso a recursos em uma rede o processo de autenticação é utilizado. Este processo consiste basicamente do fornecimento de informações que são previamente julgadas como o mínimo necessário para garantir que o usuário em questão é quem afirma ser.

Existem diversas formas de realizar a autenticação em um sistema computacional como biometria, certificado digital entre outros. Entretanto, o modelo de maior adoção é o de usuário e senha, devido seu custo benefício.

Este modelo (Figura 8), aplicado a uma arquitetura cliente servidor consiste dos seguintes passos:

1. O cliente envia a mensagem de requisição de página para o servidor;
2. O servidor responde enviando a página contendo o formulário para autenticação;
3. Após o preenchimento dos dados (usuário e senha) o cliente os envia por meio de uma mensagem HTTP do tipo POST. Os dados são enviados no corpo da mensagem;
4. O servidor inicia a aplicação responsável por gerenciar as páginas do sistema passando os parâmetros enviados pelo usuário;
5. Recebendo os parâmetros, a aplicação verifica a autenticidade dos dados de login;
6. Em caso de usuário com parâmetros inválidos de autenticação:
7. O servidor responde enviando a página de login e informa que a autenticação falhou;
8. Em caso de parâmetros válidos, a aplicação cria um *cookie* de sessão vinculado ao usuário, que permanece em memória no processo do servidor associado às requisições do cliente;
9. O servidor envia a página de acesso interno juntamente com o *cookie* de sessão para que o cliente possa armazená-lo;
10. Nas requisições consequentes o *cookie* de sessão é enviado pelo cliente junto com as requisições de páginas internas do sistema.

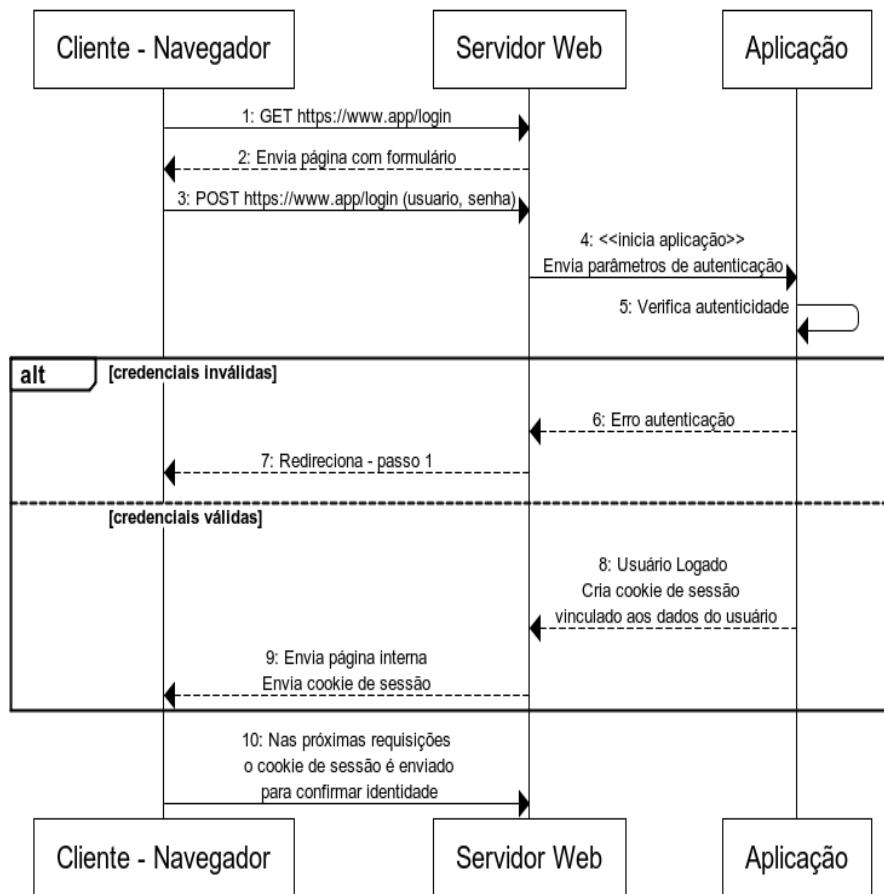


Figura 8 – Processo de autenticação utilizado atualmente nos sistemas Web

Como citado na (Seção 2.2.1), por padrão o protocolo HTTP usa conexões persistentes. Portanto logo após o término do fluxo de eventos citados acima, existirá uma *thread*, linha de execução de um processo cuja área de memória é individual ou compartilhada (COULOURIS et al., 2012), no servidor Web, com uma sessão relacionada a um cliente. A sessão é vinculada por um *cookie* negociado após a confirmação dos parâmetros de autenticação. Assim, quando o cliente for realizar uma requisição por outra página, ele enviará o *cookie* armazenado ao servidor - que verificará sua autenticidade - permitindo que o usuário possa fazer requisições de recursos com restrições de privilégios sem a necessidade de informar o login e senha a cada requisição.

Na listagem de eventos, foi omitido o processo do estabelecimento de conexão TCP, bem como o fato de ser uma conexão segura ou não, pois este processo é o mesmo independente da presença da camada de SSL (IETF, 2011) na comunicação. O projeto apresentado neste trabalho não interfere nas camadas de transporte, nem na criptografia usada para o estabelecimento de uma conexão segura sobre SSL. O projeto propõe uma forma de troca de mensagens alternativa após todas estas camadas terem realizado sua função na arquitetura de redes. Logo, o projeto engloba suas funcionalidades na camada de aplicação (IETF, 1989).

2.4.1 Vantagens e Desvantagens

Este modelo de autenticação tem grandes vantagens como:

- A curva de aprendizado pequena para o usuário;
- Ser um modelo de fácil implementação;
- Quando respeitadas as boas práticas de segurança é bastante efetivo.

Porém, boa parte das boas práticas de segurança são confiadas ao usuário bem como:

- A criação de senhas seguras;
- O armazenamento das senhas em local seguro;
- A avaliação da autenticidade do serviço acessado;
- Garantir a segurança do meio utilizado para acessar o serviço.

Isto permite maior autonomia ao usuário e ao mesmo tempo gera um problema de segurança, pois boa parte dos usuários negligencia um ou mais dos itens citados acima. Por falta de conhecimento ou por opção, devido a complexidade de gerenciar suas credenciais em um número crescente de serviços disponíveis, o usuário acaba adotando práticas de segurança prejudiciais. Como, por exemplo, usar senhas pequenas e repetidas para memorizar com mais facilidade, anotar senhas em pedaços de papel e deixar sobre a mesa, não checar a URI do serviço acessado antes de entrar com suas credenciais, dentre outros.

A fim de solucionar ou ao menos reduzir estes problemas, programas de gestão e armazenamento de senhas foram criados. Eles são responsáveis por criar senhas seguras, armazená-las e disponibilizá-las quando necessário. Isso soluciona uma boa parte dos problemas, mas cria outros dois atualmente bastante recorrentes. O de impossibilitar o usuário de usar suas credenciais em máquinas que não possuem o programa para sincronizá-las, e o de esquecer as senhas salvas em máquinas de acesso público.

2.5 Código de Barras Bidimensional

A utilização de código de barras é algo bastante comum na sociedade moderna para registrar produtos de forma rápida e prática. Seu uso foi popularizado através dos

códigos de barra unidimensionais (Figura 9), os quais são comumente encontrados em estabelecimentos comerciais.



Figura 9 – Exemplo de código de barras unidimensional

Contudo, o código de barras unidimensional, apesar de eficiente, é muito limitado no que diz respeito a quantidade de informação armazenada e a resistência do mesmo ao desgaste vindo do transporte dos produtos.

Com o objetivo de solucionar estes problemas, foram criados os códigos de barras bidimensionais (Figura 10), que além de possuir capacidade de armazenamento superior a de seu antecessor, possuem tolerância a falhas através de suas estruturas redundantes e algoritmos de correção de erros (QR CODE, 2014).



Figura 10 – Diversos modelos de códigos de barra bidimensionais

Outro ganho com o novo modelo foi o da fácil captura por câmeras digitais, presentes em dispositivos móveis, devido sua estrutura mais esparsa em relação a estrutura densa e comprimida dos códigos unidimensionais. Além de seus padrões, em sua maioria utilizarem do auto contraste, que facilita muito o processo de segmentação da imagem para sua futura decodificação.

3 O Modelo de Autenticação Delegada

O modelo desenvolvido neste trabalho tem como principal objetivo tornar possível a delegação da autenticação de credenciais de usuário para o dispositivo pessoal do mesmo com a finalidade de inibir o roubo de credenciais por programas maliciosos presentes em máquinas de acesso público.

Este modelo tem como premissa a coexistência com os modelos tradicionais de autenticação, funcionando como uma camada opcional para usuários que desejarem utilizá-lo; sem impossibilitar ou dificultar o acesso daqueles que não desejam ou não possuem os recursos necessários para utilizá-lo.

Este capítulo apresenta, de forma objetiva, a metodologia utilizada para criar o **Modelo de Autenticação Delegada**, mostrando suas características, arquitetura e funcionamento conceitual. No fim do capítulo serão expostos os trabalhos correlatos.

3.1 Pré-Requisitos do Modelo

Para o uso do modelo são utilizados:

- A máquina pública na qual se quer fazer uso do acesso autenticado;
- Um segundo dispositivo com capacidades de captura de imagem;
- Aplicativo de decodificação de código de barras bidimensionais presente no segundo dispositivo.
- Acesso, por ambos os aparelhos, à rede em que se encontra o recurso no qual será feita a autenticação.

Por exemplo, para autenticar uma máquina em uma página na internet, poderia ser usado um *Tablet* ou um *Smartphone* com câmera, aplicativo de captura de código de barras e acesso à internet.

3.2 Arquitetura

3.2.1 Token de Autenticação

O primeiro item que difere do modelo clássico de usuário e senha e possui papel fundamental no **Modelo de Autenticação Delegada** é o *Token* de autenticação.

Ele é um identificador alfanumérico aleatório, gerado a cada consulta a página de autenticação do recurso que se deseja acessar.

Todo *token*, neste modelo, possui dois estados:

- Não autenticado, que é quando ele é gerado e disponibilizado para o usuário. Neste estado ele não garante nenhum poder de acesso (Figura 11).
- Autenticado, que é quando o usuário valida suas credenciais junto a um *token* não autenticado que, a partir deste momento, torna-se um *token* autenticado. Este estado garante o alcance a uma sessão de acesso aos recursos restritos (Figura 11).

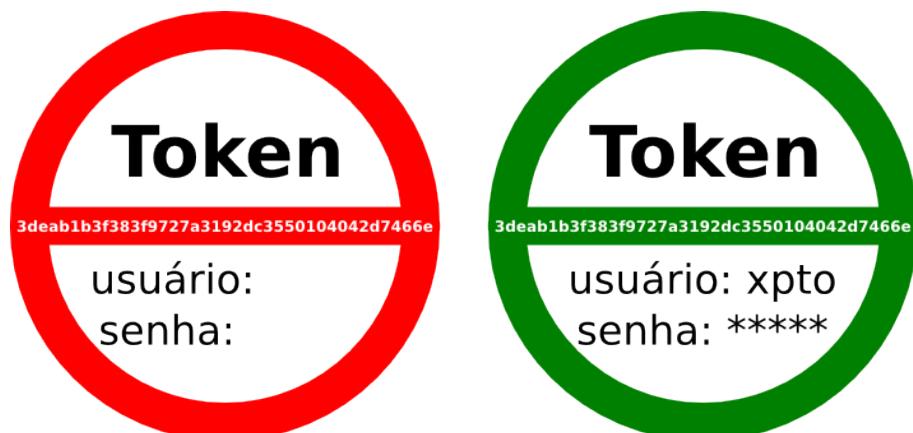


Figura 11 – Ilustração do *token* não autenticado e autenticado respectivamente

Ele deve ser gerado com maior entropia possível, para evitar tentativas de acesso por adivinhação e possuir um tamanho grande o suficiente para inibir ataques de força bruta.

3.2.2 Delegação da Autenticação

Para que seja feita a delegação, o usuário deve acessar a página de autenticação de *token*, passando como parâmetro da URI, o *token* não autenticado fornecido pelo navegador da máquina pública. Porém, inserir manualmente estes dados no dispositivo do usuário é uma tarefa desencorajadora (Figura 12).

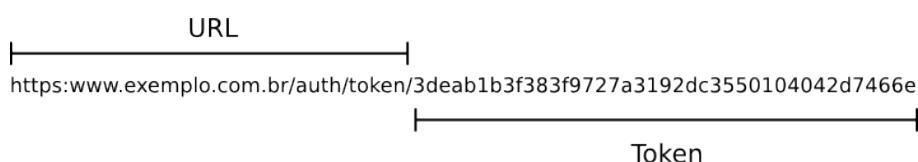


Figura 12 – Exemplo de formato do endereço de acesso a página de autenticação de *token*

3.2.3 Código de Barras Bidimensional

Com o objetivo de solucionar este problema, é usado o código de barras bidimensional para codificar os dados e disponibilizar para o usuário captura-los com seu dispositivo. Esta solução possui vantagem de agilidade no processo em relação a entrada manual pelo usuário, a não imposição de recursos de hardware na máquina pública em relação ao *bluetooth* e a interferência e o incomodo em relação a transmissão por som.

Outras vantagens do código de barras bidimensionais incluem tolerância a falhas - pois seus padrões incluem algoritmos de correção de erros - e a não necessidade de se criar um mecanismo de captura, pois seus padrões são implementados por aplicativos gratuitos em diversas plataformas de dispositivos móveis (QR CODE, 2014).

3.2.4 Conexão Segura

A fim de garantir a segurança do processo, é importante que se use uma conexão segura. Esta segurança pode ser alcançada por meio do uso de protocolos como o HTTPS (Seção 2.3.2).

3.2.5 Fluxo do Processo

O fluxo do processo de autenticação delegada pode ser descrito da seguinte forma:

Considere a hipótese em que um usuário de sistema Web precisa realizar o acesso à sua conta em uma página na Web. Ele deseja fazer isto por meio de um computador público ou compartilhado por muitas pessoas. Esta máquina pode ser considerada não confiável, e nela pode estar rodando algum software malicioso. Digitar a senha diretamente neste computador pode certamente fazer do usuário uma vítima de um *keylogger* (Seção 2.3.3).

Para se proteger ele utiliza a delegação do processo de autenticação (Figura 13):

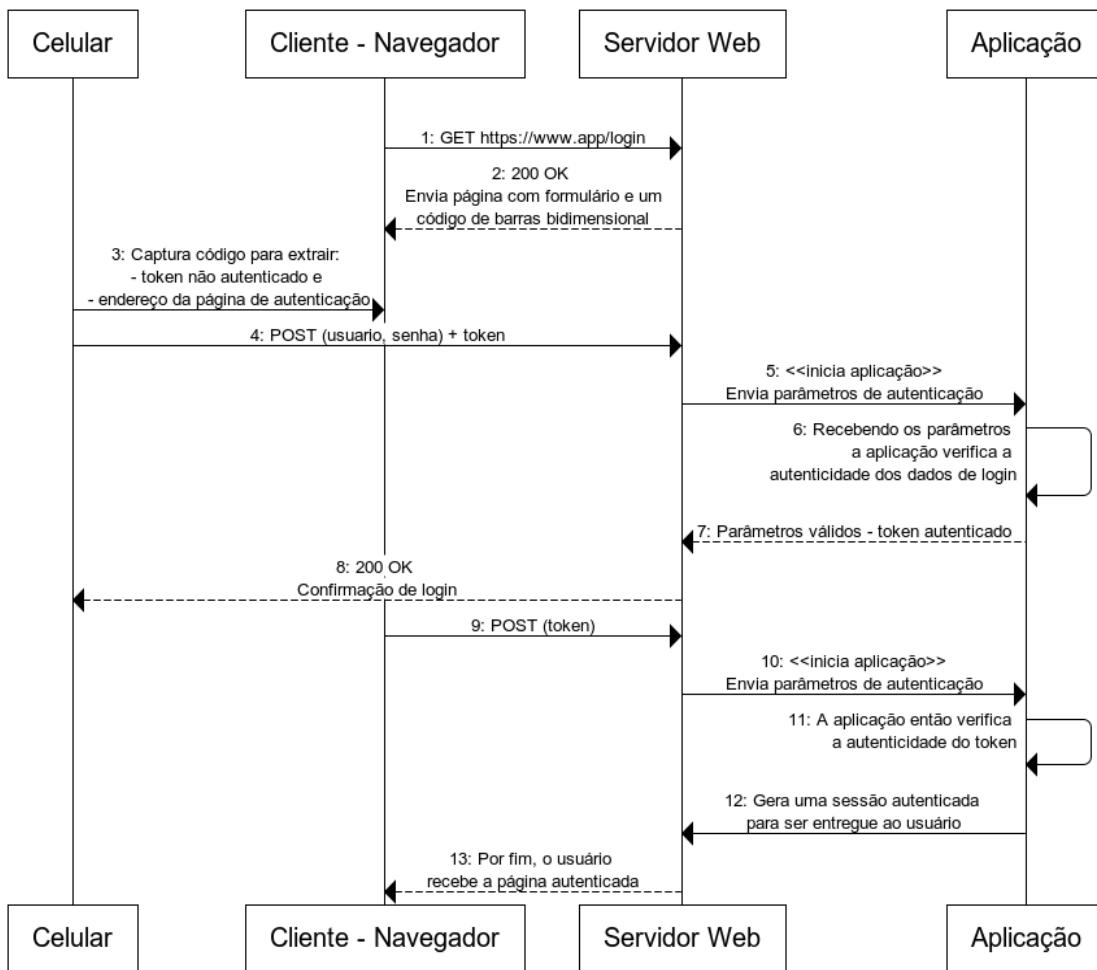


Figura 13 – Processo de Autenticação Delegada

O diagrama (Figura 13) apresenta o panorama geral do processo do qual será detalhado em breve. Entretanto se faz importante observar a presença dos elementos que diferem do modelo clássico de autenticação que são o dispositivo móvel o *token* e o código de barras, descritos previamente.

Também foram incluídas novas etapas de comunicação entre as entidades Web. O objetivo de adicionar novas etapas não é aumentar a complexidade de utilização do sistema e sim aumentar a segurança dos usuários. Entende-se que adicionar complexidade não atrai usuários, o que verdadeiramente atrai é a possibilidade de se obter mais segurança por meio de boas práticas.

O diagrama pode ser dividido nos seguintes passos:

1. O cliente acessa a página de login do serviço fazendo uma requisição ao servidor (Figura 14);



Figura 14 – Fazendo a requisição da página ao servidor

2. O servidor por sua vez, envia a página contendo um código de barras bidimensional e um formulário de *login* tradicional (Figura 15). Este código de barras contém:
 - a) O endereço da página de autenticação de *token* do sistema que fornece o serviço;
 - b) O *token* não autenticado.

Neste ponto, caso o usuário queira realizar a autenticação da maneira convencional, basta informar o usuário e a senha conforme descrito na seção 2.4.

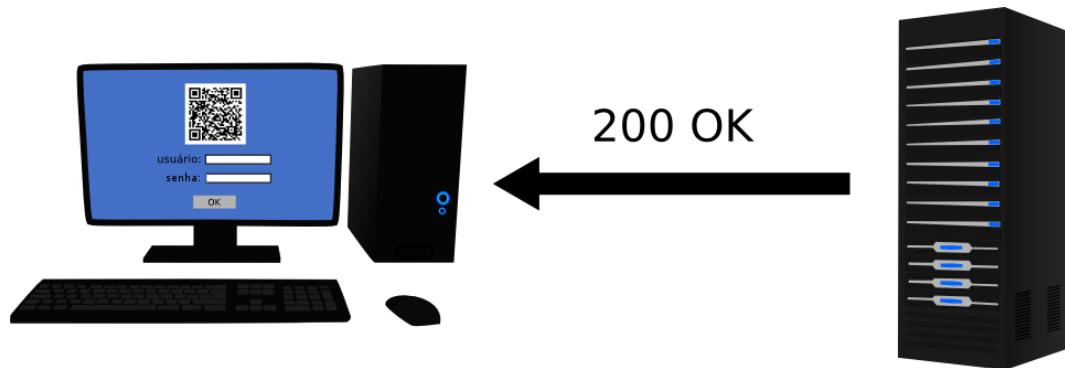


Figura 15 – Recebendo a página de autenticação via código de barras bidimensional

3. O usuário então usa o celular para escanear o código de barras (Figura 16) com um aplicativo de captura de código de barras. O aplicativo, ao decodificar a imagem, irá redirecionar o usuário para a página de autenticação de *token*;

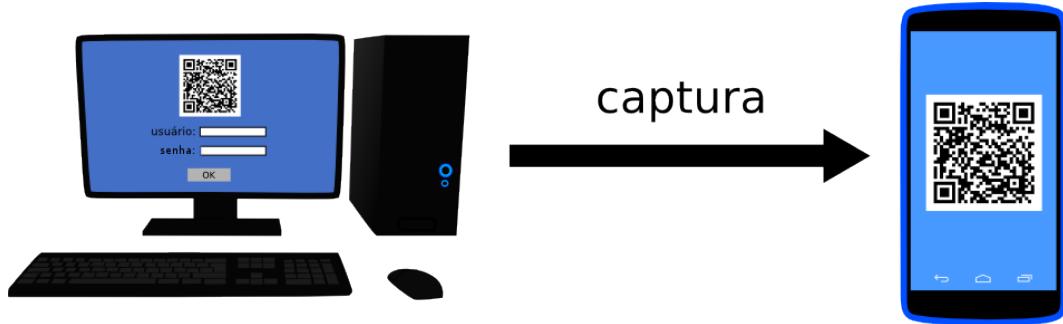


Figura 16 – Capturando os dados via código de barras bidimensional pelo dispositivo móvel

4. Na página de autenticação são preenchidos as credenciais de acesso que, por sua vez, serão enviadas ao servidor junto ao *token* não autenticado (Figura 17);

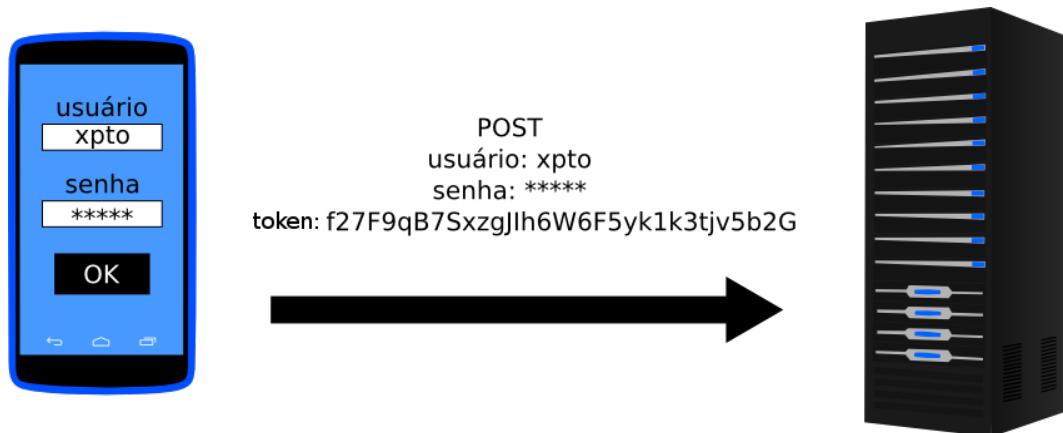


Figura 17 – Enviando credenciais para o servidor via dispositivo móvel

5. O servidor inicia a aplicação responsável por gerenciar as páginas do sistema, passando os parâmetros enviados pelo usuário;
6. Recebendo os parâmetros, a aplicação verifica a autenticidade dos dados de login;
7. Em caso de dados válidos, o *token* é vinculado ao usuário, tornando assim um *token* autenticado;
8. A aplicação do servidor então envia uma resposta de confirmação (Figura 18);

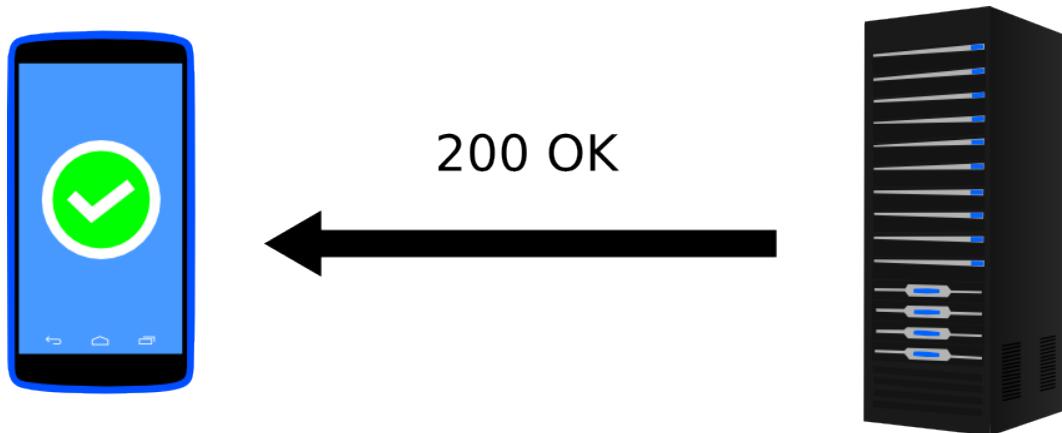


Figura 18 – Recebendo a resposta de confirmação de login

9. Após confirmado o sucesso pelo celular, o usuário submete o formulário sem preenchimento. No envio do formulário o *token* é enviado para a verificação de autenticidade no servidor (Figura 19);



Figura 19 – Enviando o usuário e o identificador de sessão para vinculação

10. O servidor, ao receber a mensagem, repassa-a para a aplicação responsável por gerenciar as páginas do sistema, passando os parâmetros enviados pelo usuário via navegador;
11. A aplicação então verifica a autenticidade do *token*;
12. Após confirmar sua autenticidade, a aplicação gera uma sessão autenticada para ser entregue ao usuário;
13. Por fim, o usuário recebe a página autenticada (Figura 20);

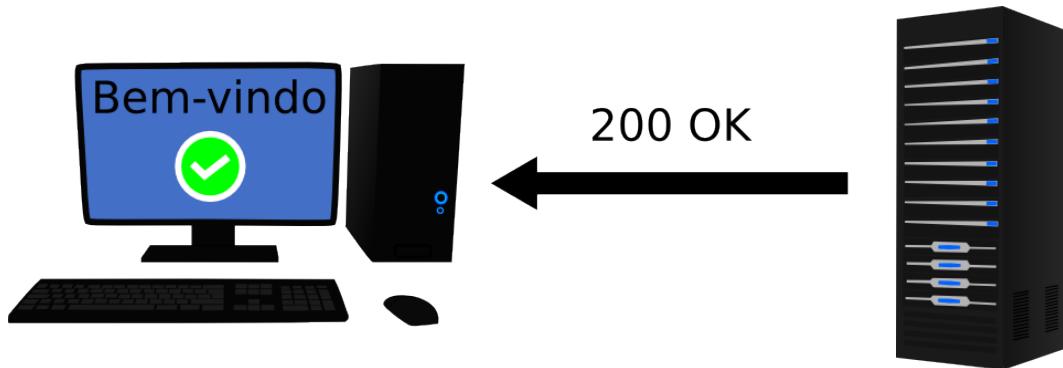


Figura 20 – Enviando a sessão autenticada para o navegador do usuário

14. O navegador neste momento possui acesso ao recurso desejado.

Do ponto de vista do usuário, o que ele deve fazer é acessar o site, escanear o código de barras bidimensional com o celular e informar o usuário e senha. E em seguida, submeter o formulário vazio no navegador para acessar a área restrita.

3.3 Trabalhos Correlatos

Os trabalhos descritos a seguir apresentam semelhanças com o **Modelo de Autenticação Delegada** e tiveram influência no desenvolvimento da pesquisa e desenvolvimento deste documento.

3.3.1 Autenticação de Múltiplos Fatores

Este modelo consiste no uso de dois ou mais fatores dos três fatores independentes de autenticação, algo que somente o usuário sabe, algo que somente o usuário possuí e algo que somente o usuário é. Por exemplo, é muito comum que usuários cadastrem seu número de telefone no provedor de serviços Web e quando for necessário se autenticar, o usuário receberá por meio de SMS em seu telefone, após informar as credenciais, um código que lhe permitirá o acesso ao serviço.

O **Modelo de Autenticação Delegada** não pode ser considerado de múltiplos fatores pois ele não força o uso de algo que o usuário informou ou registrou no serviço. Apesar de que para ser efetivo o usuário tenha que utilizar de seu dispositivo móvel, nenhuma checagem de dispositivo previamente registrado é feita.

3.3.2 SQRL - Secure Quick Reliable Login

O SQRL (GIBSON RESEARCH CORPORATION, 2014) é uma proposta de sistema de autenticação que utiliza de código de barras como meio de transferir o

processo de autenticação para um dispositivo móvel assim como o modelo descrito neste documento. Porém o SQRL usa esta delegação como um recurso de arquitetura para alcançar um objetivo diferente do apresentado pelo modelo de delegação de autenticação.

O objetivo do SQRL é substituir completamente o modelo atual de autenticação, que usa a assinatura de endereço passada via código de barras como desafio criptográfico para a validação da identidade do usuário. É um modelo ainda em desenvolvimento e de alta complexidade e custo de implementação. Além de ser um sistema que impõe para o usuário a presença de seu dispositivo com a aplicação de autenticação no processo, pois as identidades do usuário em diferentes serviços são cifradas com uma chave mestra armazenada no mesmo.

3.3.3 BB Code

O BB Code (BANCO DO BRASIL, 2011) é uma tecnologia utilizada pelo Banco do Brasil para a liberação de operações bancárias no dispositivo móvel do usuário. Diferente da autenticação delegada, no BB Code o código de barras é utilizado para substituir a ida do usuário ao banco para habilitar o seu aparelho a realização das operações. Ao capturar o código pelo *Internet Banking* ou terminal eletrônico o celular é habilitado para operações bancárias por um determinado período.

4 Implementação e Resultados

O presente capítulo busca o detalhamento do processo de implementação do **Modelo de Autenticação Delegada** aplicada a um processo de *login*. O código fonte da implementação do modelo está publicado e disponibilizado na internet, na comunidade de compartilhamento de software *GitHub*¹ podendo ser obtido por meio do seguinte endereço Web: <<https://github.com/humrochagf/bl-server/>>².

4.1 Tecnologias Utilizadas

As tecnologias computacionais utilizadas para o desenvolvimento deste trabalho estão apresentadas nas seções abaixo, juntamente com os motivos para a sua escolha.

4.1.1 A Linguagem de Programação Python

Python é uma linguagem de propósito geral e existem aplicações que a utilizam em diversas áreas da computação, como por exemplo, Desenvolvimento Web, Aplicações Científicas, Desenvolvimento de Jogos, Aplicações de Rede, dentre outros (PYTHON SOFTWARE FOUNDATION, 2014).

A linguagem *Python* foi escolhida por sua ênfase em legibilidade que traz vantagens como fácil manutenção e reuso de código. Por sua vasta biblioteca padrão e suporte da comunidade, que permite o foco na resolução do problema.

Estas características boas, somadas ao fato de os autores já estarem familiarizados com a linguagem, foram elementos importantes na escolha da linguagem para o desenvolvimento deste trabalho.

4.1.2 O Framework Django

Segundo os mantenedores do projeto (DJANGO, 2014):

“*Django* é um *framework* de desenvolvimento Web escrito em *Python* de auto nível, que encoraja o desenvolvimento rápido e um projeto limpo e pragmático.”

¹ GitHub - Build software better, together: <<https://github.com/>> Acessado em: 21/10/2014

² Acessado em: 21/10/2014

O *framework* apresenta várias facilidades para o desenvolvimento Web. Dentre elas estão a facilidade de tratar e gerenciar as URLs do seu serviço, a possibilidade de lidar com banco de dados de maneira homogênea, por meio do seu ORM (Que mapeia dados do Banco de Dados com os Objetos da Linguagem de Programação que os representam). A facilidade de separar estruturas funcionais lógicas por meio dos padrões de desenvolvimento aplicados no *framework*. A simplicidade para implementar as validações de entrada de dados, e também de se realizar os testes das suas funcionalidades.

As qualidades relatadas levaram o *Django* a ser escolhido para o desenvolvimento deste trabalho.

4.1.3 O Sistema de Controle de Versão Git

Para controlar e organizar o desenvolvimento deste trabalho, foi usado o sistema *Git*, tanto para a edição deste documento quanto para a criação do software proposto. Para administrar diferentes versões de documentos de texto, alterados por diferentes pessoas, é fundamental que se tenha um bom sistema de controle.

Git é um sistema de controle de versão descentralizado (GIT VERSION CONTROL SYSTEM, 2014), isto significa que cada pessoa envolvida na alteração do projeto possui uma cópia inteira dos arquivos do projeto. Assim, quando necessário, o *Git* auxilia os seus usuários a associar suas alterações em uma única versão, corrigir conflitos entre alterações de uma mesma parte do documento, manter versões diferentes do projeto, recuperar versões em caso de necessidade e várias outras funcionalidades. Por isso foi escolhido para auxiliar os autores do trabalho.

4.1.4 QRCode

O padrão de código de barras bidimensional utilizado na implementação do modelo de autenticação foi o QRCode (*Quick Response Code* - Código de Resposta Rápida) por apresentar as seguintes vantagens em relação aos demais (QR CODE, 2014):

- Maior popularidade em relação aos demais modelos de códigos de barras, o que consequentemente trás maior número de aplicativos capazes de decodifica-lo.
- Capacidade de representar caracteres complexos como caracteres especiais presentes em URLs e até ideogramas.
- Capacidade de ser escaneado de qualquer direção. Isto é possível por meio dos padrões de detecção de posição localizados nos três cantos do símbolo. Estes padrões garantem leitura estável de alta velocidade (Figura 21).



Figura 21 – Padrões de detecção do código de barras bidimensional. Adaptado de (QR CODE, 2014)

4.2 Simulação do Modelo

A **Autenticação Delegada com Código de Barras Bidimensional** tem o objetivo de auxiliar usuários de serviços Web a se protegerem do roubo de informação, ajudando-os a manter sua privacidade.

A simulação de uso da aplicação, que será apresentada a seguir, está hospedada na Web. Desta forma é possível demonstrar os conceitos deste trabalho de forma mais acessível e prática.

Para testar o serviço, acesse o endereço <<http://bsl-server.herokuapp.com/>> e entre com as seguintes credenciais, Usuário: “teste” e Senha: “teste123”. É possível se autenticar no serviço por meio da autenticação normal, ou por meio do seu dispositivo móvel. Como o objetivo da simulação é testar a autenticação delegada, aplicativo de decodificação de código de barras bidimensionais utilizado será o *QR Droid*¹.

O usuário, na máquina pública onde se quer fazer uso do sistema Web, acessa o site do serviço (Figura 22). As funções responsáveis pelo processamento desta página estão definidas no Apêndice A.

¹ Aplicativo disponível em: <<https://play.google.com/store/apps/details?id=la.droid.qr&hl=en>> Acessado em: 23/09/2014



Figura 22 – Tela do site de simulação: <<http://bsl-server.herokuapp.com/>>

Em seguida, o usuário escaneia o código de barras com o aplicativo *QR Droid* (Figura 23).

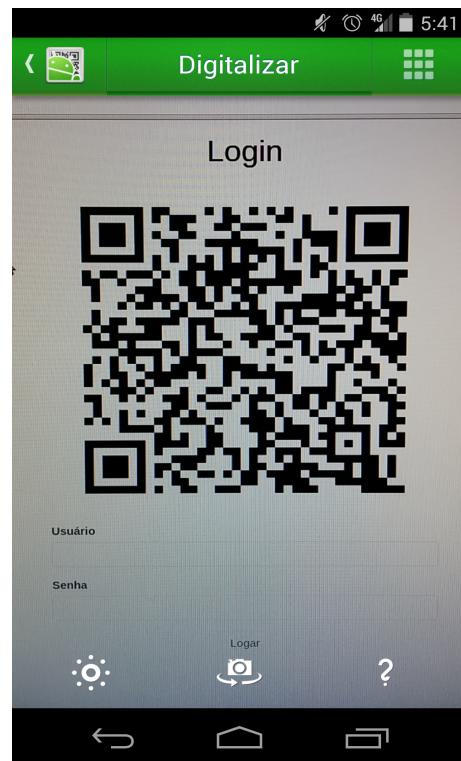


Figura 23 – Captura do código de barras por meio do aplicativo *QR Droid*

Por padrão, o aplicativo de código de barras irá abrir o navegador do dispositivo móvel do usuário. A página contém o formulário para o preenchimento e envio das credenciais (Figuras 24 e 25). O processamento responsável por gerar esta página, e retorná-la para o dispositivo do cliente está impresso no Apêndice B.



Figura 24 – Página de autenticação delegada



Figura 25 – Página de autenticação delegada com credenciais preenchidas

Após preencher as informações e enviar para o servidor, o usuário recebe a mensagem de confirmação. Indicando, assim, que o acesso já pode ser realizado na máquina inicial (Figura 26). O código de implementação da autenticação delegada está contido no Apêndice B.

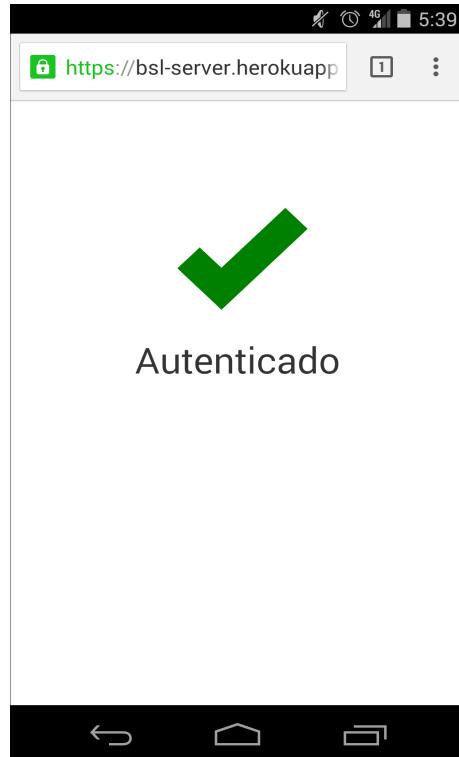


Figura 26 – Confirmação de autenticação

Para realizar o acesso na máquina, o usuário aperta o botão *Logar* e então ele será redirecionado para a página de acesso restrito, possuindo uma sessão válida (Figuras 27 e 28). A implementação da verificação de autenticidade do usuário está contida no Apêndice A.



Figura 27 – Tela do site de simulação: <<http://bsl-server.herokuapp.com/>> - antes de acessar a área restrita

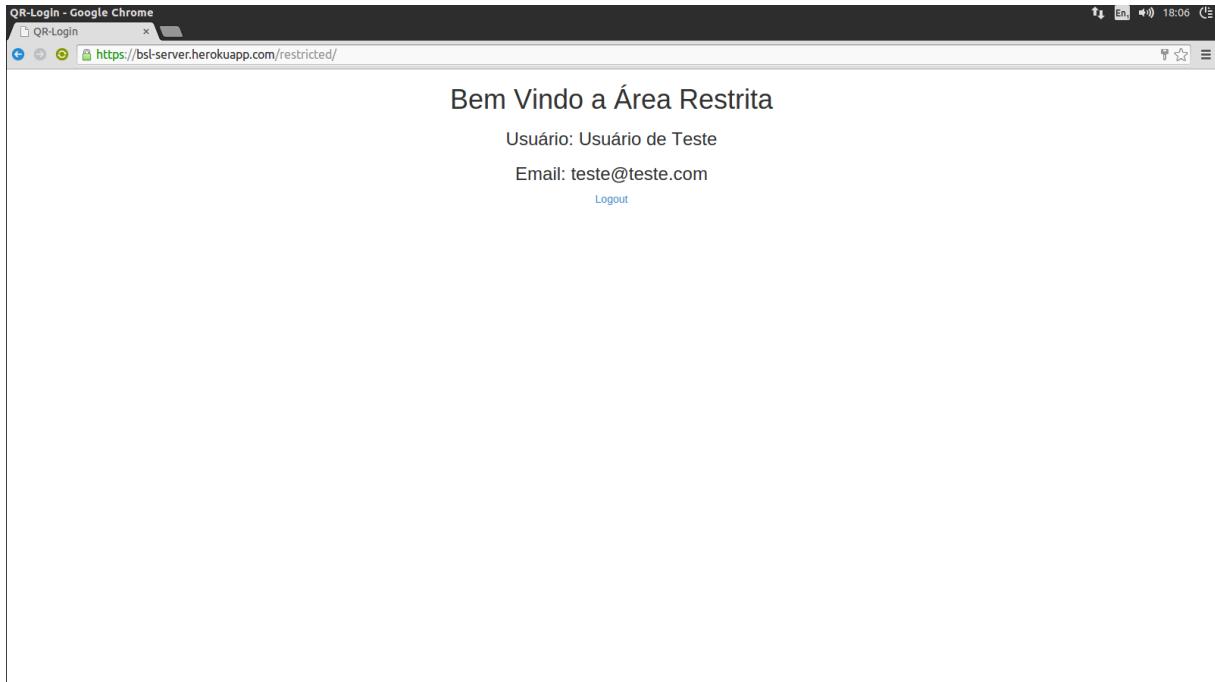


Figura 28 – Tela do site de simulação: <<http://bsl-server.herokuapp.com/restricted>> - acesso à área restrita

4.3 Discussão e Resultados

Conforme apresentado no início deste trabalho, o objetivo de desenvolver um modelo de autenticação capaz de preservar as credenciais de acesso dos usuários foi cumprido. O **Modelo de Autenticação Delegada** é capaz de fornecer uma alternativa aos usuários que desejam se autenticar em sistemas Web, de forma que suas credenciais sejam informadas em um ambiente conhecido pelo usuário da aplicação.

Apresentar uma arquitetura de autenticação que possa coexistir com o modelo vigente também foi um objetivo cumprido. Portanto, o usuário pode escolher se deseja obter os recursos da autenticação delegada. Caso queira se autenticar da maneira convencional, basta realizar o procedimento padrão de autenticação.

O sistema implementado neste trabalho tem o objetivo de ilustrar o funcionamento prático do modelo. É importante notar que a interface exibida é apenas uma das formas de se organizar o formulário de autenticação. Os campos do formulário e a imagem podem estar dispostos de várias outras formas e em tamanhos diferentes, se adequando ao estilo de qualquer site. O importante é que o código de barras esteja disponível e visível para ser escaneado.

O **Modelo de Autenticação Delegada** não resolve todos os problemas relacionados à segurança no processo de autenticação. Ele atende a um objetivo específico que é preservar as credenciais do usuário, entretanto a vulnerabilidade de roubo de sessão não é resolvida na implementação deste trabalho. Observe que o problema de roubo de sessão não é característico da autenticação delegada, e sim de qualquer mecanismo de autenticação. No modelo de autenticação delegada, o roubo de sessão poderia ocorrer logo após o usuário autenticar o *token* por meio do seu dispositivo pessoal. Caso outra pessoa, má intencionada, de alguma maneira, tenha acesso ao *token* e o envie ao serviço antes do usuário que o autenticou, a sessão será roubada. É possível estudar soluções de implementação para prevenir o roubo de sessão na autenticação delegada, mas isto está fora do escopo deste trabalho.

Como a implementação demonstrada neste trabalho utiliza as tecnologias *Python* e *Django*, as organizações ou pessoas que utilizam estas tecnologias como ferramentas de trabalho podem se beneficiar da autenticação delegada de forma simples. Basta observar a implementação feita neste trabalho e reproduzi-la no produto desejado. A implementação exposta neste trabalho é de código aberto e está disponível na internet, conforme descrito no início do Capítulo 4. Futuramente, pode-se modularizar a implementação aqui apresentada para disponibilizá-la como um aplicativo do *Framework Django*. Assim sendo, caso um usuário deseje utilizar a autenticação delegada, basta instalar o aplicativo por meio da interface do *Framework*, e usar os métodos de autenticação conforme o modelo aqui demonstrado.

5 Conclusão e Trabalhos Futuros

Como demonstrado, o modelo de **Autenticação Delegada com Código de Barras Bidimensional** é uma solução a funcional e pode ser considerada em situações onde é preciso realizar a delegação do processo de autenticação para dispositivos de confiança, como por exemplo autenticar se máquinas de acesso público ou de terceiros sem entrar com suas credenciais nas mesmas.

Portanto, o projeto proposto alcançou os objetivos citados no capítulo inicial deste trabalho, apresentando uma arquitetura segura, prática e que pode coexistir com o modelo vigente permitindo que usuários se autentiquem de forma segura em estações de trabalho não confiáveis.

Sugestões de possíveis trabalhos futuros:

- Delegação de permissões específicas. Como por exemplo, um caixa de supermercado ao requerer o cancelamento de determinado produto, pode pedir para o gerente do setor para autenticar o cancelamento de seu dispositivo devidamente registrado.
- Encoraja-se também, o uso deste trabalho como o uso deste modelo sendo a base do controle de acesso a ambientes físicos de acesso restrito.
- Aprofundamento em pesquisas de segurança na exploração de possíveis vulnerabilidades do modelo, como o roubo de sessão.

Referências

APACHE SOFTWARE FOUNDATION. *SSL/TLS Strong Encryption*. Apache HTTP Server, 2014. Disponível em: <http://httpd.apache.org/docs/current/ssl/ssl_intro.html>. Acesso em: 3.10.2014.

BANCO DO BRASIL. *BB Code*. 2011. Disponível em: <<http://www.bb.com.br/portalbb/page3,105,5564,0,0,1,1.bb?codigoMenu=435&codigoNoticia=40038&codigoRet=17051&bread=1>>. Acesso em: 17.10.2014.

COMPUTER ECONOMICS. *Annual Worldwide Economic Damages from Malware*. 2007. Disponível em: <<http://www.computereconomics.com/article.cfm?id=1225>>. Acesso em: 24.09.2014.

COOKIE CHECKER. *Which cookies does a site use?*: Cookie summary for nytimes.com. 2014. Disponível em: <<http://www.cookie-checker.com/check-cookies.php?url=http%3A%2F%2Fwww.nytimes.com%2F>>. Acesso em: 17.9.2014.

COULOURIS, G. et al. *Distributed Systems: Concepts and design*. 5^a. ed. [S.I.]: Pearson Addison-Wesley, 2012.

DJANGO. *The Web framework for perfectionists with deadlines*. 2014. Disponível em: <<https://www.djangoproject.com/>>. Acesso em: 30.09.2014.

GIBSON RESEARCH CORPORATION. *Secure Quick Reliable Login*: Proposing a comprehensive, easy-to-use, high security replacement for usernames, passwords, reminders, one-time-code authenticators. 2014. Disponível em: <<https://www.grc.com/sqrl/sqrl.htm>>. Acesso em: 14.6.2014.

GIT VERSION CONTROL SYSTEM. *Git - distributed even if your workflow isn't*. 2014. Disponível em: <<http://git-scm.com/about/distributed>>. Acesso em: 29.09.2014.

HALL, B. J. *Beej's Guide to Network Programming*: Using internet sockets. 3^a. ed. [S.I.: s.n.], 2012.

IEFT. *Uniform Resource Identifier (URI)*: Generic syntax. Internet Engineering Task Force, 2005. Disponível em: <<http://tools.ietf.org/pdf/rfc3986.pdf>>. Acesso em: 13.10.2014.

IETF. *Internet Protocol*. Internet Engineering Task Force, 1981. Disponível em: <<http://tools.ietf.org/pdf/rfc791.pdf>>. Acesso em: 15.04.2014.

IETF. *Transmission Control Protocol*. Internet Engineering Task Force, 1981. Disponível em: <<http://tools.ietf.org/pdf/rfc793.pdf>>. Acesso em: 15.04.2014.

IETF. *Domain Names: Concepts and facilities*. Internet Engineering Task Force, 1987. Disponível em: <<http://tools.ietf.org/pdf/rfc1034.pdf>>. Acesso em: 26.05.2014.

IETF. *Domain Names: Implementation and specification*. Internet Engineering Task Force, 1987. Disponível em: <<http://tools.ietf.org/pdf/rfc1035.pdf>>. Acesso em: 26.05.2014.

IETF. *Requirements for Internet Hosts: Communication layers*. Internet Engineering Task Force, 1989. Disponível em: <<http://tools.ietf.org/pdf/rfc1122.pdf>>. Acesso em: 23.05.2014.

IETF. *The Transport Layer Security (TLS) Protocol Version 1.1*. Internet Engineering Task Force, 2006. Disponível em: <<http://tools.ietf.org/pdf/rfc4346.pdf>>. Acesso em: 2.10.2014.

IETF. *The Secure Sockets Layer (SSL) Protocol Version 3.0*. Internet Engineering Task Force, 2011. Disponível em: <<http://tools.ietf.org/pdf/rfc6101.pdf>>. Acesso em: 23.04.2014.

IETF. *Request for Comments (RFC)*. Internet Engineering Task Force, 2014. Disponível em: <<http://www.ietf.org/rfc.html>>. Acesso em: 11.11.2014.

KUROSE, J. F.; ROSS, K. W. *Computer Networking: A top-down approach*. 5^a. ed. [S.I.]: Pearson Addison-Wesley, 2010.

NCE/UFRJ. *Construção de Páginas Web com HTML - nível básico*. Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, 2014. Disponível em: <<http://www.nce.ufrj.br/ginape/cursohtml/>>. Acesso em: 13.10.2014.

PETERSON, L. L.; DAVI, B. S. *Computer Networks: A systems approach*. 5^a. ed. [S.I.]: Elsevier, 2011.

PYTHON SOFTWARE FOUNDATION. *Python is powerful and fast; plays well with others; runs everywhere; is friendly and easy to learn; is Open*. 2014. Disponível em: <<https://www.python.org/about/>>. Acesso em: 28.09.2014.

QR CODE. *What is a QR Code? You can learn about the QR Code's features and standards here.* 2014. Disponível em: <<http://www.qrcode.com/en/about/>>. Acesso em: 13.08.2014.

SIKORSKI, M.; HONIG, A. *Practical Malware Analysis*: The hands-on guide to dissecting malicious software. 1^a. ed. [S.I.]: No Starch Press, 2012.

STALLINGS, W. *Criptografia e segurança de redes*: Princípios e práticas. 4^a. ed. [S.I.]: Pearson Prentice Hall, 2008.

STATISTA. *Number of worldwide internet users from 2000 to 2014*. 2014. Disponível em: <<http://www.statista.com/statistics/273018/number-of-internet-users-worldwide/>>. Acesso em: 18.11.2014.

SYMANTEC. *What the Latest Symantec Threat Report Means to SMBs*. 2014. Disponível em: <http://www.symantec.com/en/za/solutions/article.jsp?aid=20090512_what_the_latest_symc_threat_report_means_to_smbs>. Acesso em: 14.08.2014.

TANENBAUM, A. S. *Redes de computadores*. 5^a. ed. [S.I.]: Pearson Prentice Hall, 2011.

VERISON. *2012 Data Breach Investigations Report*. 2012. Disponível em: <http://www.verizonenterprise.com/resources/reports/rp_data-breach-investigations-report-2012_en_xg.pdf>. Acesso em: 4.10.2014.

6 APÊNDICE A - Funções responsáveis pela tela inicial de login e autenticação convencional

```
def login(request):
    template = 'core/login.html'
    token = generate_token()
    error = None

    if request.method == 'POST':
        form = LoginForm(request.POST)

        if form.is_valid():
            user = auth.authenticate(
                username=form.cleaned_data['username'],
                password=form.cleaned_data['password'],
                token=form.cleaned_data['token'])

            if user:
                if user.is_active:
                    user.token = None
                    user.save()

                    auth.login(request, user)

                    return redirect('/restricted/')
                else:
                    error = 'Erro de autenticação: \
                            Usuário inativo.'
            else:
                error = 'Erro de autenticação: \
                        Credenciais inválidas.'

            form.update_token(token)
        else:
            form = LoginForm()
            form.fields['token'].initial = token

    auth_token_page = request.build_absolute_uri(
        reverse('auth_token', args=(token, )))

    context = dict(
        qrcode=make_qrcode_base64(auth_token_page),
        form=form, error=error)

    return render(request, template, context)
```

```
def generate_token():
    # Generate the nonce
    # TODO: Work on the entropy and size of the nonce
    return sha1(urandom(128)).hexdigest()

def make_qrcode_base64(text):
    # Create a temporary placeholder to the image
    with BytesIO() as stream:
        # Generate the qrcode and save to the stream
        qrcode.make(text).save(stream)

        # Get the image bytes, then encode to base64
        image_data = b64encode(stream.getvalue())

    return image_data
```

7 APÊNDICE B - Funções responsáveis pela autenticação delegada

```
def token_authentication(request, url_token=None):
    template = 'core/token_authentication.html'
    success = False
    error = None

    if request.method == 'POST':
        form = TokenAuthForm(request.POST)

        if form.is_valid():
            user = auth.authenticate(
                username=form.cleaned_data['username'],
                password=form.cleaned_data['password'])

            if user:
                if user.is_active:
                    user.token = form.cleaned_data['token']
                    user.last_token_auth = timezone.now()
                    user.save()

                    success = True
                else:
                    error = 'Erro ao autenticar token: \
                            Usuário inativo.'
            else:
                error = 'Erro ao autenticar token: \
                        Usuário ou senha incorreto.'

    else:
        form = TokenAuthForm()
        form.fields['token'].initial = url_token

    context = dict(success=success,
                  token=url_token,
                  form=form,
                  error=error)

    return render(request, template, context)
```

```
class CustomUserBackend(ModelBackend):
    def authenticate(self, username=None, password=None, token=None):
        user = super().authenticate_
            username=username,
            password=password)

        if not user and token:
            try:
                user = User.objects.get(token=token)
                if user.get_token_age() > 60:
                    user = None
            except ObjectDoesNotExist:
                user = None
            except MultipleObjectsReturned:
                user = None

        return user
```