A evolução da Kryptonita

O objetivo deste documento é esclarecer as escolhas tecnológicas feitas no desenvolvimento da versão para Google Chrome sob Windows da solução Kryptonita de modo a apontar os desafios a serem enfrentados na nossa tentativa de aumentar a abrangência do seu público-alvo. Ele se destina a gestores e desenvolvedores de aplicativos que já utilizam ou que estejam pensando em utilizar a solução e que queiram interferir nos destinos do projeto.

Sumário executivo

Este documento sustenta que a arquitetura corrente da Kryptonita foi a mais simples solução possível para o problema de incluir recursos criptográficos nos navegadores Internet em vista das características do problema, do mercado de navegadores e das escolhas tecnológicas disponíveis. A solução está composta pelos seguintes componentes:

1. Uma interface para programação em Javascript para aplicações web ;
2. Extensões de navegador implementadas em Javascript: Kryptonita Emissão (obrigatória para o SITKN) , Kryptonita Validação de Assinaturas (opcional), Kryptonita Assinatura Digital (obrigatória para o SICNS) e Kryptonita Zip (de uso obrigatório);
3. Uma aplicação nativa, implementada em C e compilada exclusivamente para Windows 8.1 ou mais recente;
4. Um instalador próprio para Windows;
5. Um módulo de segurança (Kiripema), exclusivamente para Mozilla Firefox.

O instalador está disponível na Internet no site da Caixa como download exclusivo para o piloto do SITKN e incorporada ao processo de homologação da nova versão do SICNS e atende somente ao navegador Google Chrome para Windows pelas razões expostas no documento.

O documento propõe também um plano de evolução do produto, detalhado na seção Plano de evolução da Kryptonita, que deverá resultar na expansão da sua base de abrangência no mercado de navegadores, com as características resumidas a seguir:

Navegador	**Versão**	**Sistema**	**Distribuição**	**Acesso**
Google Chrome	84	Windows	Chrome Web Store	Não listado
Mozilla Firefox	58	Windows	Mozilla Addons	Público
Microsoft Edge Chromium	84	Windows	Microsoft Store	Não listado
Opera	70	Windows	Opera Addons	Público

Características da solução

A solução Kryptonita (ver https://bitbucket.org/yakoana/kryptonite/src/master/) é uma biblioteca de programação para aplicações web composta por um conjunto de extensões de navegador Internet

acoplados a um aplicativo nativo[1] que fornece os recursos criptográficos para essas aplicações. Ela endereça especialmente o problema do usuário assinar documentos com certificados digitais compatíveis com os padrões da ICP Brasil numa aplicação web.

O produto resultante atende igualmente aos padrões da ICP Brasil para assinaturas digitais, produzindo documentos no padrão CAdES (CMS *Advanced Electronic Signature*), conforme padronização do IETF publicada na RFC 5126[2]. Os envelopes de assinatura digital resultantes podem facilmente ser validados no Verificador de Conformidade do Instituto de Tecnologia da Informação[3], desde que tenha sido utilizado um certificado digital da cadeia da ICP Brasil.

Dizemos que se trata de um "problema" porque uma aplicação web reside e executa num computador remoto (um *servidor*), acessando somente os recursos disponíveis neste computador e na rede interna onde reside. Ora, as assinaturas digitais são feitas através do acesso a recursos residentes no computador do usuário assinante, em dispositivos periféricos como cartões inteligentes ou arquivos em disco. Para evitar a "falsificação" da assinatura digital, tais recursos (conhecidos como "chaves privadas") não podem, em hipótese alguma, ser enviados ao servidor remoto onde executam as aplicações web.

Não se trata de um problema exclusivo das aplicações criptográficas. Para evitar ataques maliciosos ao computador dos usuários, os navegadores Internet que apresentam e, no caso das aplicações "ricas", executam parcialmente as aplicações web, limitam enormemente os recursos acessíveis a essas aplicações. Em linhas gerais, as aplicações web são capazes de utilizar somente os recursos fornecidos pelos navegadores na execução de instruções em HTML, CSS e Javascript[4]. Isso permite, por exemplo, que uma aplicação web leia um arquivo residente em disco, mas somente se este arquivo estiver desprotegido pelo sistema operacional e se o usuário selecioná-lo explicitamente através de um recurso de interface fornecido pelo navegador. Além disso, para não deixar vazar informações privadas, o navegador jamais fornece à aplicação web a localização física deste arquivo no disco do usuário.

Tais recursos não são capazes de atender à totalidade das demandas por aplicativos "ricos" executando sob hospedagem dos navegadores web, embora suficiente para a imensa maioria dos casos. Outro fator que dificulta esse atendimento integral é o fato da evolução daqueles padrões se dar em ritmo muito mais lento que a evolução da demanda por novos recursos. O fato é que tais padrões requerem o consenso dos fabricantes de navegadores para serem aprovados e, naturalmente, tempo e dinheiro para serem implementados pelos inúmeros produtos de software concorrentes no mercado.

---

1 Um "aplicativo nativo" é uma aplicação compilada especificamente para um sistema operacional e instalada diretamente pelo usuário no seu computador. O navegador Internet através do qual acessamos boa parte das aplicações no nosso dia a dia e o editor de texto no qual este documento está sendo escrito são exemplos de aplicativos nativos.

2 IETF (*Internet Engineering Task Force*) é uma organização internacional destinada a manter os diversos padrões de mais largo uso na Internet. Tais padrões são publicados na forma de RFC (*Request For Comments*). O documento citado pode ser encontrado em https://tools.ietf.org/html/rfc5126.

3 Ver https://verificador.iti.gov.br/verifier-2.5.4/.

4 HTML (*Hipertext Markup Language* – linguagem de marcação de hipertexto) e CSS (*Cascading Style Sheets* – folhas de estilo em cascata) são padrões de linguagens declarativas mantidas pelo W3C (*World Wide Web Consortium*, https://www.w3.org/). Javascript é uma linguagem de programação procedural especificada e mantida pela *Ecma International* (https://www.ecma-international.org/).

De fato, existe uma especificação do W3C que define as características de um objeto para o interpretador Javascript dos navegadores. Este objeto deveria fornecer serviços criptográficos para as aplicações web, conforme definido na *Web Cryptography API*[5]. No entanto, nenhum dos fabricantes de navegadores implementaram essa especificação nem têm planos de implementá-la.

Para endereçar esses inúmeros problemas não atendidos (e, consequentemente, atrair mais e mais usuários para seus serviços), o mercado de navegadores já há vários anos implementou mecanismos de ampliar as capacidades desses produtos de software. Trata-se das "extensões de browser", também chamadas "aditivos" (*add-ons*). As extensões são pequenos aplicativos que executam no próprio navegador (e não num computador remoto) e para os quais são fornecidos serviços e capacidades não disponíveis para as aplicações web. Um exemplo é o Google Tradutor, que permite ao usuário obter tradução automática das páginas web enquanto navega, sem necessidade de acessar um site específico.

Aqui surge um problema. Ao contrário dos recursos fornecidos para as aplicações web, as extensões não são padronizadas. Cada fabricante de navegador fornece suas próprias APIs (*Application Programming Interface* – interface de programação de aplicações), com características distintas, muitas vezes contraditórias com aquelas fornecidas pelos concorrentes. Naturalmente, existem inúmeros pontos de contato entre os diferentes navegadores, já que, na maioria dos casos, eles se baseiam em produtos de software comuns, geralmente desenvolvidos e distribuídos em regime de software livre. No entanto, cada fabricante se esmera em imprimir suas próprias digitais nessas APIs, visando disputar a atenção dos usuários. Na prática, não é possível dar a mesma solução para um problema tão específico e de pouco alcance no mercado como a assinatura digital para todo e qualquer navegador. Assim, as primeiras decisões do projeto Kryptonita se basearam no estudo do mercado de navegadores web.

O mercado de navegadores web

Como todo navegador se identifica por um cabeçalho específico do protocolo de comunicação – o *user agent* – quando efetua uma requisição em qualquer site, a melhor maneira de determinar o *market share* dos diversos navegadores web é simplesmente contar os diferentes identificadores por requisição à página no acesso a um conjunto suficientemente grande de web sites ao longo do tempo. Isso tem sido feito há vários anos pelo site https://gs.statcounter.com/, que alega agregar dados de cerca de 10 bilhões de requisições do protocolo (*page views*) coletadas mensalmente em cerca de 2 milhões de sites web. Durante o mês de julho próximo passado, a distribuição do mercado de navegadores, calculada por aquele site, é vista na Figura 1.

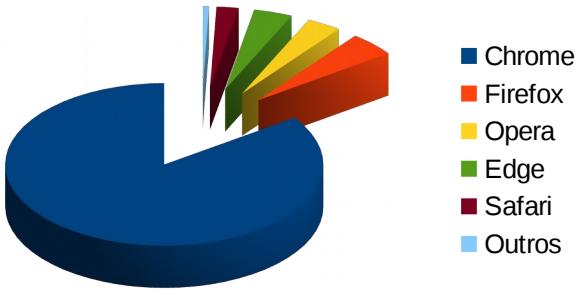

Figura 1: Market share de navegadores web (julho 2020)

A amostra considerou somente os acessos feitos no Brasil por navegadores instalados em computadores *desktop* (ou *notebooks*), já que estes são os requeridos pelas aplicações endereçadas pela

5 Ver https://www.w3.org/TR/WebCryptoAPI/.

solução, que não é adequada a *smart phones* e outros dispositivos portáteis. Como se pode notar, embora possam existir dezenas de navegadores web, o mercado é altamente concentrado em alguns poucos produtos. Aliás, a predominância do Google Chrome beira o monopólio.

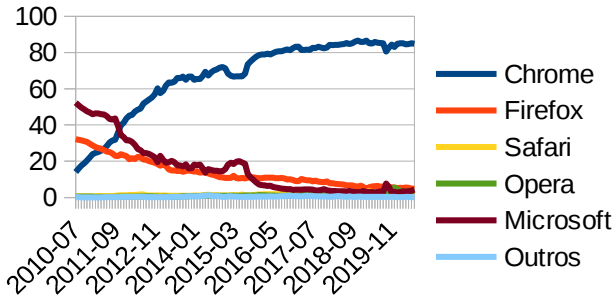

Figura 2: Evolução do market share dos navegadores

Não se trata de acidente no mercado, mas de tendência de longo prazo, nascida com o lançamento da primeira versão do Google Chrome, que rapidamente suplantou o predomínio histórico dos navegadores da Microsoft. Essa tendência pode ser facilmente observável na Figura 2, que, exceto pelo período de abrangência, iniciado em julho de 2010, obedece aos mesmos critérios da anterior. Por si só, a distribuição da pizza dos navegadores e o chamado *princípio de Pareto* nos indica qual solução deveria ser adotada para melhor atender ao *market share* dos navegadores. Portanto, todas as decisões tecnológicas que tomamos obedeceram a essa compreensão.

Essa decisão fez com que a solução a ser apresentada não pudesse se limitar à entrega exclusivamente de uma extensão. Ocorre que somente o Firefox fornece recursos criptográficos às suas extensões e ele não alcança 5% do mercado. Isso significa que a solução precisaria necessariamente ser implementada como um aplicativo nativo, cabendo ao seu componente "extensão de browser" a função exclusiva de intermediar a comunicação desse aplicativo com a aplicação web.

E se o produto deve incluir uma aplicação nativa, a decisão de para qual sistema operacional ela deve ser compilada passa a ser necessária. Essa foi igualmente fácil, dado o predomínio (novamente próximo do monopólio) de um único sistema operacional para computadores de mesa. A distribuição do acesso à Internet no Brasil por sistema operacional em julho passado pode ser vista na Figura 3.

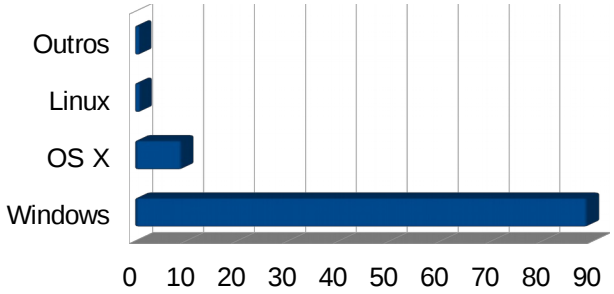

Figura 3: Market share de sistemas operacionais (jul/20)

Embora o OS X (o sistema operacional dos computadores Apple) possa parecer ter alguma relevância no mercado brasileiro, ele geralmente está associado ao uso pessoal. Profissionalmente, os computadores da Apple vêm sendo usados principalmente nas atividades de artes gráficas, fotografia e cinema. Mesmo nesses casos, as empresas do ramo costumam utilizar o Windows como sistema operacional de seus computadores administrativos pela simples razão de que os aplicativos de logística costumam ser implementados para este sistema operacional quando instalados em estações de trabalho de propósito geral. Assim, o Windows é a escolha natural para uma solução voltada para uso empresarial.

Os componentes da solução

Dadas aquelas escolhas, a arquitetura da solução foi então concebida com três componentes: (a) uma API de programação em Javascript, responsável por fornecer os serviços criptográficos à aplicação web; (b) uma aplicação nativa, responsável por implementar esses serviços criptográficos e (c) uma extensão de navegador, responsável pela comunicação entre os dois componentes primários. Ainda que o público-alvo inicial devesse ser composto por usuários do Google Chrome para Windows e as tecnologias utilizadas para a comunicação entre os três componentes precisassem ser suportadas por aquele navegador, onde houvesse escolha nossa preferência recairia por aquela suportada por todos ou pelo menos pela maioria dos concorrentes.

No caso da comunicação entre a aplicação nativa e a extensão não houve escolha disponível: o único mecanismo recomendado pela Google é a tecnologia de *native messaging* (mensagens nativas). É verdade que ainda hoje os navegadores, incluindo o Chrome, suportam outras tecnologias de acesso ao computador do usuário, como o uso de aplicativos baseados na máquina virtual Java e o suporte a *plug-ins* (mini aplicativos) de browser. No entanto, todos os fabricantes que suportavam tais tecnologias já concordaram há algum tempo em descontinuar esse suporte em futuro próximo por razões de segurança e visando a estabilidade de seus produtos. Apenas o mini aplicativo para leitura de documentos em formato PDF será mantido pelos navegadores. Portanto, a solução não pode ser baseada nelas.

Para implementar *native messaging*, os navegadores fornecem uma API que permite que a extensão troque mensagens de texto com aplicativos nativos registrados, usando os dispositivos padrão de entrada e saída – o equivalente computacional à entrada de dados pelo teclado e à exibição de caracteres pela tela. Cabe ao navegador executar e sustentar a aplicação nativa, transferindo os dados originados na extensão para a aplicação e vice-versa. Naturalmente, a aplicação nativa deve ser capaz de executar sem interface própria e se comunicar através do dispositivo de E/S (entrada e saída) padrão.

Adotada para o Google Chrome, essa tecnologia pode ser utilizada para os outros navegadores que a utilizam: o Mozilla Firefox, o Opera e as duas versões coexistentes do Microsoft Edge. O navegador da Apple – o Safari – não suporta essa tecnologia, aliás como quase todos os padrões adotados pelo mercado. Salvo neste caso, o aplicativo nativo desenvolvido poderia ser adotado sem alterações para os demais navegadores.

Infelizmente, quando o assunto é desenvolvimento para o mercado de navegadores, as coisas nunca são tão simples quanto pensamos. O problema que precisamos lidar a seguir é o modo como os navegadores implementam os recursos criptográficos que fornecem – as conexões SSL/TLS[6], em especial aquelas envolvendo a autenticação do cliente. Tanto o Google Chrome quanto o Opera e (naturalmente) o Microsoft Edge, utilizam a infraestrutura criptográfica fornecida pelo próprio sistema operacional. Já o Mozilla Firefox se utiliza da biblioteca NSS (*Network Security Services*) para o fornecimento dos serviços criptográficos, armazenando os certificados digitais e chaves privadas

6 SSL (*Secure Sockets Layer*) e TLS (*Transport Layer Security*) – camada de transporte seguro – são protocolos projetados para fornecer comunicação criptografada em redes de computadores. Esses protocolos cumprem três funções distintas: proteger o canal de comunicação contra espionagem e interferência, identificar o servidor e, em alguns casos, como o Conectividade Social, identificar o cliente da comunicação. Estes protocolos se utilizam de criptografia de chaves públicas e certificados digitais para identificar as partes e iniciar a comunicação criptografada. Presentemente, SSL é considerado obsoleto, tendo sido substituído por TLS.

utilizados em repositórios diferentes daqueles mantidos pelo Windows. Neste caso, o acesso a tais repositórios é feito através da interface fornecida por “módulos de segurança” baseados no padrão PKCS #11[7].

A solução adotada foi acrescentar um elemento arquitetural no caso da distribuição para Firefox: um módulo de segurança PKCS #11 – o Kiripema (https://bitbucket.org/yakoana/kiripema/src/master/). Esse produto faz com que o Firefox passe a suportar a utilização de certificados digitais instalados no repositório do Windows e não somente os instalados nos seus próprios repositórios.

Quanto à comunicação entre a API web e a extensão, a escolha optou pela adoção de padrão uniforme entre os navegadores em lugar da simplicidade e elegância. Cada navegador fornece sua própria API para integração entre a camada web e as extensões, naturalmente incompatível com todos os demais concorrentes. Tais APIs fornecem o serviço com inegável simplicidade e eficiência quando comparados aos padrões. No entanto, são APIs proprietárias, o que implicaria em implementações diferentes da extensão para cada navegador a ser suportado. Assim, optamos por utilizar a tecnologia de troca de mensagens entre documentos (*Web Messaging*)[8] definida nas especificações HTML.

Distribuição da solução

Infelizmente, distribuir a Kryptonita não é algo tão simples quanto efetuar o download de um instalador, embora isso precise ocorrer. Como um dos componentes é uma aplicação nativa, ela precisa ser distribuída como um instalador padrão para o sistema operacional alvo. Tornamos o processo tão simples quanto possível, fornecendo um instalador que requer somente a aprovação da licença de uso (GPL – *GNU Public License*) e permite escolher quais funcionalidades serão instaladas, a saber: emitir requisições de certificados digitais, assinar digitalmente e verificar a assinatura de documentos. O instalador não requer permissões administrativas no computador do usuário, instalando o aplicativo somente para o usuário corrente.

O problema a ser resolvido aqui – que afetou mesmo a arquitetura da solução – é seu componente extensão de browser, que precisa ser instalado no navegador do usuário e não diretamente no seu computador. Isso faz com que as regras e padrões a serem seguidos sejam definidos pelo fornecedor do navegador – e cada navegador tem suas próprias definições de distribuição! O Google Chrome conta com dois modelos distribuição: através da Chrome Web Store, a “loja” de distribuição de produtos para o navegador, e por instalação externa.

Na loja, os produtos ficam expostos em “gôndolas” por tipo de aplicação ou por “promoções”, as extensões “recomendadas” pela Google. As extensões distribuídas pela loja são atualizadas automaticamente pelo navegador, quando necessário, sem necessidade de ação do usuário, o que é uma enorme vantagem para produtos com milhares de usuários. Além disso, a Google tenta emprestar a esses produtos alguma credibilidade, realizando revisões de qualidade e atendendo a reclamações dos

7 *Public-Key Cryptography Standard #11*. Trata-se de uma especificação destinada a definir uma API de acesso a dispositivos criptográficos de alta segurança como cartões inteligentes e HSM (*Hardware Security Module*). Publicada originalmente pelo RSA Laboratories, o padrão é mantido hoje pela organização de padrões abertos OASIS (ver https://www.oasis-open.org/).

8 Ver https://www.w3.org/TR/webmessaging/.

usuários. Porém, como os usuários precisam encontrar o que desejam instalar, este não parece um bom meio de distribuição para um produto que milhares de usuários serão obrigados a usar. Além disso, eles ainda precisarão buscar o instalador da aplicação nativa para download convencional.

A instalação externa não implica, porém, que a extensão deva necessariamente ser hospedada fora da loja da Google. Aliás, o fabricante adverte que "o não cumprimento de um desses métodos de distribuição (descritos na documentação) constitui uma violação da política de extensão do Chrome e pode fazer com que a extensão e/ou o software que o distribui seja sinalizado como software indesejado"[9]. Caso se deseje que a extensão seja atualizada automaticamente pelo navegador nas distribuições para Windows e OS X, a extensão deve estar hospedada na loja, mesmo que não esteja "listada", isto é, acessível à busca manual pelos usuários. Trata-se de um modelo elaborado para permitir que aplicações (como o componente nativo da Kryptonita) possam distribuir direta e facilmente extensões e para instalação em redes corporativas sob controle dos administradores.

No caso do Windows, o instalador da aplicação (ou um script administrativo de rede) insere determinadas chaves e valores no banco de dados de registro (a chamada *Registry*) do sistema operacional. Quando detecta esses valores (ele o faz periodicamente e não apenas quando começa a execução), o navegador efetua o download da extensão. Simplicidade não significa transparência para o usuário. Para prevenir que extensões maliciosas sejam contrabandeadas para o navegador por esse meio, o browser exige que o usuário aceite habilitar a extensão. Caso contrário, ela não é acionada, embora esteja instalada. Além disso, se o usuário remover a extensão pela interface do navegador, ela é marcada como *indesejada* e não poderá mais ser instalada sem uma "cirurgia" nos arquivos do perfil do browser[10].

A simplicidade mencionada nos levou a adotar esse mecanismo de instalação externa. No entanto, para assegurarmos a atualização automáticas das extensões, decidimos por hospedá-la na Chrome Web Store, na categoria de "não listada". As atualizações automáticas são essenciais para uma extensão que, ao que tudo indica, será utilizada por dezenas ou centenas de milhares de usuários, graças aos sistemas da Caixa SICNS e SITKN. Isso garante que não apenas eventuais erros corrigidos estejam disponíveis imediatamente, como também assegura que novas funcionalidades e características sejam incorporadas com a mesma presteza.

A hospedagem na Chrome Web Store, como em quaisquer das lojas dos navegadores, impõe o cumprimento de determinadas regras, que se propõem a proteger os usuários contra software malicioso[11] e violações de privacidade. Particularmente relevante para a arquitetura da Kryptonita foi a regra do "propósito único"[12] a que as extensões devem obedecer. Essa regra fez com que a solução incluísse quatro extensões de navegador: Kryptonita Emissão, Kryptonita Validação de Assinaturas, Kryptonita Assinatura Digital e Kryptonita Zip. As três primeiras estão mapeadas nas opções do instalador; a última é de instalação obrigatória.

---

9 https://developer.chrome.com/extensions/external_extensions.

10 No caso da Kryptonita, se a remoção for feita através do Painel de Controle do Windows, utilizando o desinstalador do aplicativo, esse efeito colateral não ocorre e a extensão poderá ser novamente instalada pelo usuário sem aquela "cirurgia" metafórica.

11 https://www.google.com/about/unwanted-software-policy.html.

12 https://developer.chrome.com/extensions/single_purpose.

O futuro da Kryptonita

A pizza da Figura 1 nos sugere que somos capazes de atender 97% do mercado de navegadores com pouco esforço – porque é esta a soma das fatias do Google Chrome, Mozilla Firefox, Opera e os dois Microsoft Edge[13] hoje disponíveis. Como as tecnologias adotadas pelo Safari são totalmente incompatíveis com as adotadas pelos demais navegadores, é improvável que uma versão do produto para esse browser aproveite sequer uma única linha de código já escrito (e elas são contadas em milhares e não centenas). Portanto, entendemos que o suporte ao Safari não é viável pela Kryptonita, já que teríamos que desenvolver um produto inteiramente novo, atendendo a exigências inteiramente novas e com documentação mínima. Entendemos que se trata de esforço excessivo para atender parcela de mercado não maior que 2,7%.

O atendimento a outros navegadores já está em parte pronto, ainda que não consideremos os produtos aptos para distribuição. No site do projeto Kryptonita estão disponíveis versão tanto para Mozilla Firefox quanto para Microsoft EdgeHTML. Esta última, porém, não recomendamos utilizar. Por força dos padrões adotados então pela Microsoft, fomos obrigados a incluir na solução mais dois componentes a serem instalados no computador do usuário: uma aplicação UMP (*Universal Windows Platform*) e um outro executável nativo, capaz de interagir com aplicações UMP e servir de "adaptador" para a interface *native messaging* adotada pelos outros navegadores. A multiplicação de camadas entre a API web e a aplicação nativa que realmente executa as funções criptográficas aumenta exponencialmente os pontos de falha e, portanto, desaconselha seu uso. Em vista disso, sequer submetemos as extensões à Microsoft.

O problema do Firefox não é a complexidade da solução, mas as restrições do distribuidor, que não permitem mais a adoção do mecanismo simples adotado para o Google Chrome: desde a versão 73 do navegador e alegando razões de segurança[14], o Firefox não suporta mais a instalação através do registro do Windows. Isso significa que o usuário deve instalar não apenas a aplicação nativa como também cada uma das extensões manualmente. Desse modo, não há vantagem alguma em utilizar a distribuição própria das extensões Firefox conforme descrito na documentação[15]. Nossa recomendação, portanto, é publicar cada uma das extensões da Kryptonita e encaminhar os usuários que desejarem usar este navegador ao endereço de distribuição localizado na AMO (*Add-ons Mozilla Organization*), já que essa publicação nos garante a atualização automática das extensões.

O que ainda é possível fazer para facilitar os usuários do Mozilla Firefox é automatizar a instalação do módulo de segurança Kiripema a partir de quaisquer das extensões instaladas. Esse

13 A Microsoft está sustentando atualmente duas versões diferentes do Edge: o EdgeHTML, embarcado no seu sistema operacional e baseado em versão antiga do Mozilla e uma nova versão, recentemente lançada, baseada no Chromium (ver https://www.chromium.org/Home). Como seus anúncios indicam a prevalência da nova versão, podemos esperar que ela acabe por ser embarcada nas novas versões do sistema operacional. Assim, em que pesem as diferenças, estamos contando esses dois produtos como um só e supondo que seu *market share* se manterá constante na soma do dois.

14 Ver https://blog.mozilla.org/addons/2019/10/31/firefox-to-discontinue-sideloaded-extensions/.

15 Ver https://extensionworkshop.com/documentation/publish/self-distribution/.

recurso está disponível desde a versão 58 do Firefox, de resto (e por outras razões técnicas[16]) a versão mais antiga suportada pela Kryptonita.

Com a liberação do novo Microsoft Edge, baseado no Chromium, o problema de adaptar a Kryptonita para execução sob este navegador foi automaticamente resolvido. E mais, como ele adota as mesmas políticas de distribuição que o Google Chrome, inclusive a instalação através de chaves na *Registry*, esse suporte passou a ser trivial, requerendo tão somente a submissão das extensões à loja Microsoft Store e a confecção de instalador próprio. Mesmo as políticas de conformidade adotadas são muito semelhantes às da Google.

Por fim, a distribuição de um sabor da Kryptonita para o navegador Opera obedece às mesmas limitações do sabor Firefox, já que o fabricante não admite instalação através da *Registry*. Como o produto, tal como o Edge e o Chrome, se baseia no projeto de software livre Chromium, as adaptações necessárias são mínimas.

No momento, a base de usuários da Kryptonita para Google Chrome obedece à distribuição mostrada na Tabela 1. Esses números estão ligados, provavelmente, a entrada em piloto do SITKN. Com a entrada em produção da nova versão do SICNS, prevista para os próximos meses, também em regime de piloto, esses números estarão em franca expansão. É, portanto, um bom momento para planejarmos sua expansão para outros navegadores. Caso as recomendações que fizemos até o momento sejam aprovadas, é possível propor o seguinte plano de atividades:

Extensão	Versão	Usuários
Kryptonita Emissão	1.1.14	1.237
Kryptonita Validação de Assinaturas	1.1.12	1.240
Kryptonita Assinatura Digital	1.1.13	1.257
Kryptonita Zip	1.1.12	1.245

Tabela 1: Usuários da Kryptonita em 20 de agosto de 2020

Plano de evolução da Kryptonita

1 Revisão e refatoração do projeto para os novos objetivos
2 Suporte ao Mozilla Firefox versão 58 ou mais recente
  2.1 Implementação de recurso de instalação automática do módulo de segurança Kiripema através das extensões de navegador eventualmente instaladas pelo usuário;
  2.2 Adaptação do instalador Kryptonita para suporte ao navegador;
  2.3 Publicação das extensões na loja AMO;
  2.4 Elaboração de documentação de orientação para a instalação do produto pelos usuários;
  2.5 Testes de aprovação e homologação.
3 Suporte ao Microsoft Edge versão 84 ou mais recente
  3.1 Adaptação das extensões para garantia de suporte ao navegador;
  3.2 Adaptação do instalador Kryptonita;
  3.3 Publicação das extensões na loja Microsoft Store;

16 O fato dessa ser a versão requerida pela Kryptonita fez com que sua distribuição Firefox jamais fosse testada adequadamente, mesmo pelos homologadores, no ambiente da Caixa, onde a versão do produto ainda é a 52.6.0. As razões que nos levaram a utilizar esta versão, além da automação da instalação do módulo de segurança, foi o fato de desejarmos não limitar o tamanho dos arquivos assinados pela Kryptonita, o que só era possível com permissões só disponíveis a partir daquela versão.

3.4 Testes de aprovação e homologação.

4 Suporte ao Opera versão 70 ou mais recente

4.1 Adaptação das extensões para suporte ao navegador;

4.2 Adaptação do instalador Kryptonita;

4.3 Publicação das extensões na loja Opera Addons Store;

4.4 Elaboração de documentação de orientação para a instalação do produto pelos usuários;

4.5 Testes de aprovação e homologação.

Rio de Janeiro, 20 de agosto de 2020

Marco Antonio Gutierrez	Diego de Souza von Sohsten
Spread Tecnologia	CEDES/RJ