



Tor vs FBI: A história, mitigações

EEL840 - Segurança
Miceli - 2022.1

Alunos: Guilherme, Osmar,
Paulo

História do caso



Em 2013 o FBI utiliza um exploit do javascript pelo firefox utilizado pela rede Tor no Windows para identificar usuários de conteúdos ilegais

1

Controle do serviço de hospedagem de sites ilegais

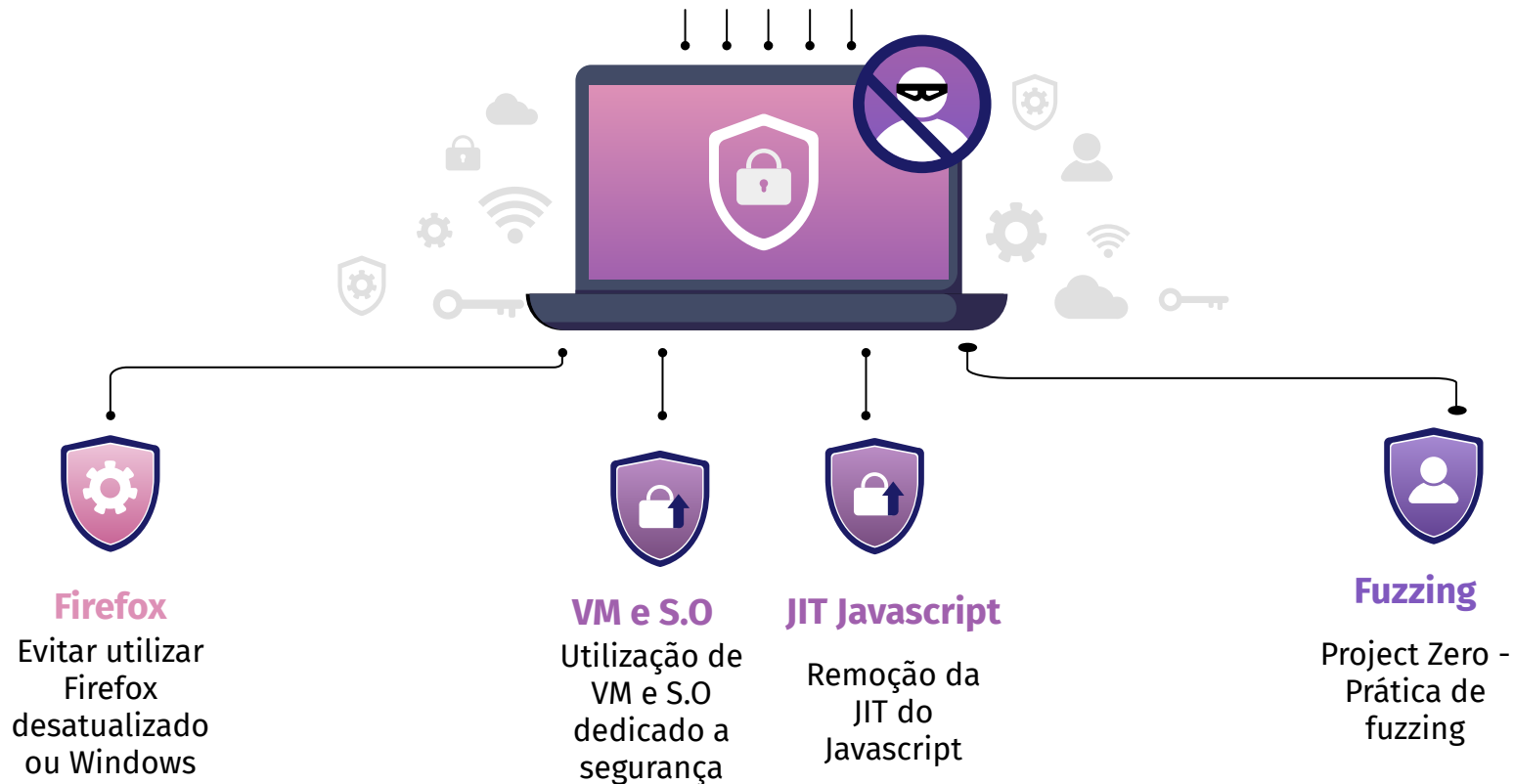
2

Injeção de código malicioso ao consumir o conteúdo alterado

3

Execução de código local arbitrário e transmissão de dados pessoais para o FBI

Mitigações



Como ocorreu o ataque

Etapa 1

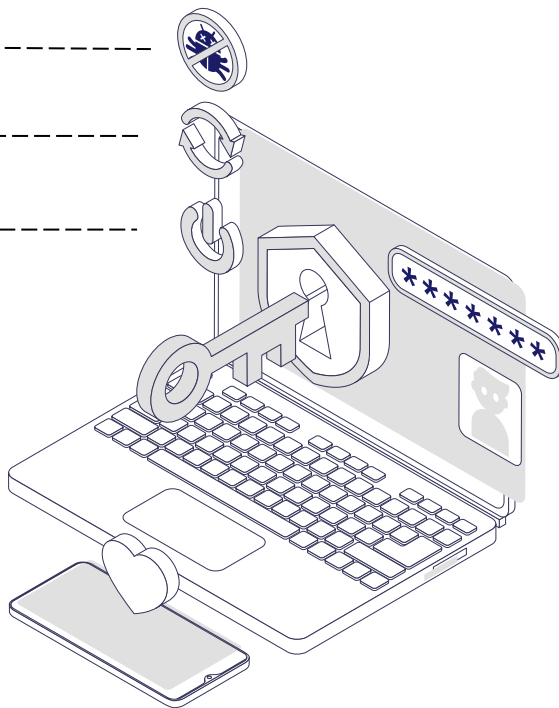
Verificação da execução do exploit e chamada ao iframe que identifica o site acessado

Etapa2

Execução do heap spray por chamadas iterativas de iframes para executar o use-after-free

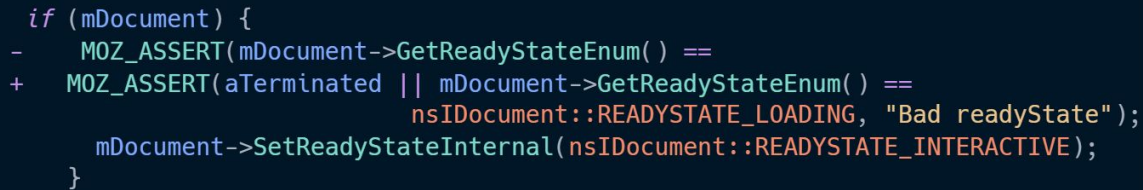
Etapa 3

Corrupção da heap e execução de código arbitrário para envio de informações pessoais fora da rede Tor



Análise do fix

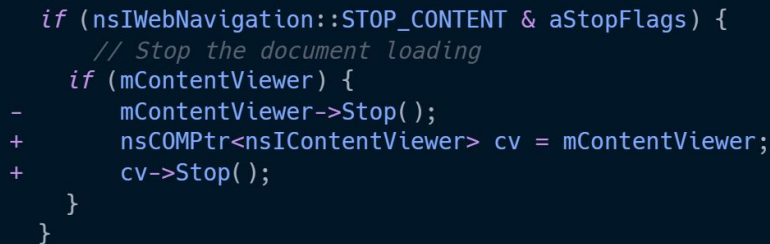
O primeiro foi o arquivo: content/base/src/nsContentSink.cpp

A code editor window with a dark blue background and three colored window control buttons (red, yellow, green) in the top-left corner. It displays a C++ code diff. The code is as follows:

```
if (mDocument) {  
-   MOZ_ASSERT(mDocument->GetReadyStateEnum() ==  
+   MOZ_ASSERT(aTerminated || mDocument->GetReadyStateEnum() ==  
                        nsIDocument::READYSTATE_LOADING, "Bad readyState");  
    mDocument->SetReadyStateInternal(nsIDocument::READYSTATE_INTERACTIVE);  
}
```

Esse MOZ_ASSERT não é uma branch que deve ser executada na prática, mas está aí pra evitar justamente que bugs como esse não evoluam para RCEs, causando um crash controlado, limitando-os a apenas DOSs. O fix do bug ocorre em outro arquivo.

Análise do fix



```
if (nsIWebNavigation::STOP_CONTENT & aStopFlags) {  
    // Stop the document loading  
    if (mContentViewer) {  
-        mContentViewer->Stop();  
+        nsCOMPtr<nsIContentViewer> cv = mContentViewer;  
+        cv->Stop();  
    }  
}
```

- A mudança é bastante simples, em vez de executar a função `Stop()` numa referência fraca, o smart-pointer é copiado para incrementar a contagem de referências. E só então `Stop()` é chamado.
- Isso evita que quando `mContentViewer` for a única referência àquele `nsIContentViewer` interno, a cópia para `cv` fará com que o contador de referências vá para 2 e portanto `Stop()` não dealocará variáveis internas que ainda serão acessadas durante a mudança do estado `readystatechange`.
- Mostrando o porquê do javascript ter chamado `“window.stop()”` duas vezes com um event listener pra `“readystatechange”` ativo, no meio de uma sequência de reloads