UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LOOPHOLE

Ataques temporizados em loops de eventos compartilhados no Chrome

Alberto Andretta
Augusto Cadini
João Victor do Pilar

SEGURANÇA COMPUTACIONAL

INTRODUÇÃO

- Estudo da vulnerabilidade dos loops de eventos compartilhados em ataques secundários;
  - Processo espião monitora o padrão de uso do loop de outro processo enfileirando eventos e medindo o tempo que leva para ele ser despachado.
- Na Programação Orientada a Eventos (EDP - Event-driven programming)
  - Componentes centrais: Loops de Eventos.
  - *Loop Event,* consiste em uma fila FIFO
  - São utilizados pelos processos para armazenar e despachar mensagens recebidas de outros processos;
  - Eventos que chegam são colocados na fila e são despachados sequencialmente;

PROGRAMAÇÃO ORIENTADA A EVENTOS

- Ações de usuários (clicar, arrastar mouse, pressionar teclas, aumentar volume)
- Sinais de entrada/saída (Carregar arquivo, processos de rede, baixar arquivos)
- Mensagens de outros programas (Comunicação de processos por passagem de mensagem)

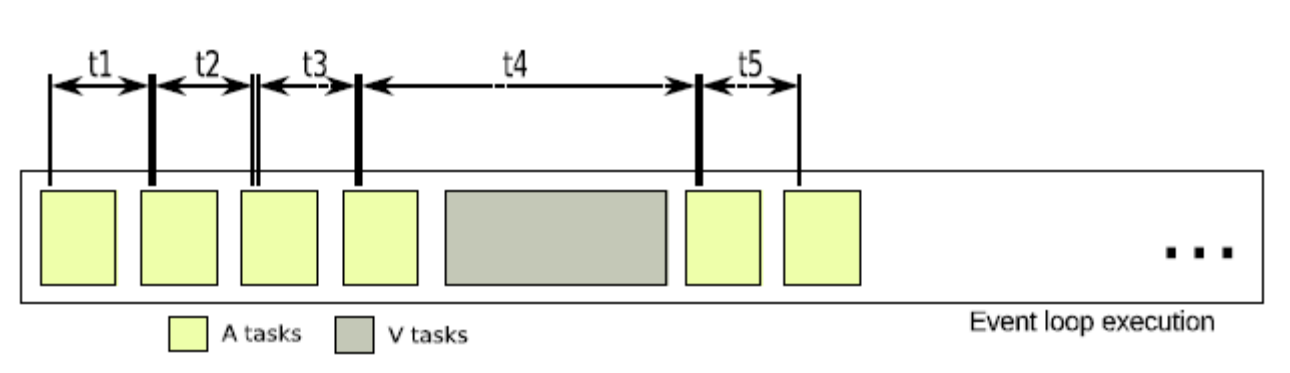
OBJETIVO DO ARTIGO

- Demonstrar que *loops* de eventos compartilhados são vulneráveis a ataques *side-channel.*
- Usando: um processo espião monitora os padrões de uso do *loop* pelos processos do navegador.

FILA DE UM LOOP DE EVENTO COMPARTILHADO

Ilustra o cenário para um loop que é compartilhado entre um atacante (A) e uma vítima (V)

1. Enfileira várias tarefas e registra o tempo em que cada uma delas é processada.
2. A diferença de tempo entre duas tarefas consecutivas revela se a tarefa V foi postada entre tarefas e quanto tempo elas levaram para executar



Loop de evento compartilhado

PROCESSOS ALVO

- 2 processos centrais de loops de eventos compartilhados no Google Chrome.
    1. **Processo Host: Loop de eventos é compartilhado entre todos os pedidos de recursos comuns (rede, interface de usuários).**
    2. **Processo de Renderização:** Loops podem ser compartilhados entre tarefas Javascript de diferentes abas e *iframes.*

Cada uma das threads mantém pelo menos um *loop* de eventos.

São realizadas 3 demonstrações:

1. Expõe como os atrasos de eventos durante a fase de carregamento podem ser utilizados para identificar uma página Web de maneira exclusiva.

2. Mostra como as ações do usuário em páginas de origem cruzada podem ser detectadas com base nos atrasos que eles introduzem no loop de eventos.

3. Demonstra que os loops de eventos compartilhados podem ser usados para transmitir informações entre páginas de origem cruzada.

POLÍTICA DE MESMA ORIGEM

Same-Origin Policy (SOP)

- Restringe os scripts de uma página da Web a acessar dados de outra página, se suas origens forem diferentes.

- Duas páginas têm a mesma origem se o protocolo, a porta e o host forem iguais.

- Trazendo desafios aos navegadores para uma comunicação flexível de origem cruzada.

COMPARTILHANDO NO PROCESSO DE RENDERIZAÇÃO

- Chrome suporta diferentes políticas que controlam como os aplicativos da web são mapeados e que influenciam se os loops são compartilhados ou não:
  - ***Política padrão (processo por site):*** Requer o uso de um processo de renderização dedicado para cada instância de um site. EX: https://docs.google.com e https://mail.google.com:8080 são do mesmo site, mas não da mesma origem, pois diferem no subdomínio e na porta.
  - ***Política de processo por site:*** Agrupa todas as instâncias de um site no mesmo processo de renderizado;
  - ***Política de processo por guia:*** Dedica um processo de renderizador a cada grupo de guias conectadas por script;

Em (64 bits) OSX e Linux, o limite para reutilizar renderizadores é calculado dividindo metade da RAM física entre os representantes, sob a suposição de que cada um consome 60 MB

COMPARTILHANDO NO PROCESSO HOST

- Renderizadores precisam se comunicar com o processo do *host* para solicitações de rede ou entrada do usuário;
- Mensagens correspondentes de todos os representantes passam pelo *loop* de eventos da *thread* de I/O do processo *host*.

Ilustra-se a comunicação utilizando 2 exemplos:

1. ***Fluxo da Interface do usuário:*** Como as ações do usuário fluem do host para o processo de renderização correspondente;
2. ***Pilha de Rede:*** Como as solicitações de rede fluem de um renderizador para o processo de host.

ESPIONAGEM DOS EVENTOS

- Violar o SOP (Política de mesma origem) e assim espionando os *event loops* da renderização de processos e do host do Chrome.

- Para cada processo uma potencial ameaça, que descreveremos em seguida.

CENÁRIO - *Renderer Process Event Loop*

1. ***Keylogger:*** A exibe um formulário de login para autenticar seus usuários por meio do OAuth de V. Como a operação não pede privilégios especiais e a senha nunca é enviada para A, a vítima pode confiar nela e preencher o formulário. Assim, a página de A monitora os intervalos de digitação, que pode ser usada para recuperar senhas de usuários.
2. ***Anúncio Malicioso:*** A é executado como um anúncio *iframed* em V. O SOP protege a privacidade e a integridade do V isolando logicamente ambos os ambientes de execução. No entanto, o *iframe* de A pode executar o Javascript no loop de eventos do V, permitindo que forneça informações sobre o comportamento do usuário em V.

TÉCNICAS DE MONITORAMENTO

- Para monitorar o loop de eventos do renderizador, é suficiente postar continuamente tarefas assíncronas e medir o intervalo de tempo entre os pares subsequentes de eventos.
- Medimos a resolução de monitoramento em termos do intervalo entre dois eventos de medição subsequentes em um loop vazio.

CENÁRIO - *Host Process Event Loop*

- Canal Secreto: Páginas de diferentes origens em execução em guias diferentes (desconectadas) podem usar o loop de eventos compartilhados para implementar um canal secreto, violando os mecanismos de isolamento do navegador. Esse canal pode ser usado para rastrear usuários em sessões ou para filtrar informações de páginas suspeitas sem um filtro de rede.
- Impressão Digital: Uma guia executando uma página desonesta pode identificar quais páginas estão sendo visitadas pelo usuário em outras guias, espionando o loop de eventos compartilhado. Isso é possível pelos blocos de I/O quando digitada a URL e pressionado *enter*.

TÉCNICAS DE MONITORAMENTO

- Solicitações de rede: Usar solicitações de rede para monitorar sistematicamente o loop de eventos do *thread* de I/O do processo do host.

- Trabalhadores Compartilhados: Baseia-se em *web workers*, que é um mecanismo para executar o *Javascript* no *backround*. E assim pode ser usado para espionar o encadeamento de I/O do processo *host*.

LoopScan Tool

- Foi desenvolvida uma ferramenta que permite explorar as características *side-channel* causadas pelos loops de eventos compartilhados.

LoopScanTool: https://github.com/cgvwzq/loopscan

ATAQUES

3 tipos de ataques serão estudados

1. Identificação de página
1. Observação das ações do usuário
1. Canal secreto

1. IDENTIFICAÇÃO DE PÁGINA

Identificar páginas carregadas em outras abas.

1. Selecionar amostras de sites. (500)
2. Coleta de dados. Visitar as páginas por 30x. Registrar os traços de eventos de delay no carregamento das páginas.
3. Cassificação: Duas aproximações

a) Histograma de eventos de *delay.*
b) Manter informações temporais sobre os eventos observados.

4. Técnicas de aceleração (Observação de tempo de execução)
5. Ajustes de parâmetros: (Para item 4). Duração de processo, intervalo de amostras, tipo de janela, tamanho de janelas, padrões de passos.)

2. AÇÕES DE USUÁRIO

Mostrar que é possível detectar ações do usuário executadas em uma aba de origem cruzada ou iframe, quando o processo do renderizador é compartilhado

1. Google OAuth Form: Medir intervalos de digitação de teclas em formulários de autenticação.

-> Pode revelar o tamanho de uma senha/nome usuario, mas não seu conteúdo.

-> 91.5% dos casos acertaram o tamanho de uma senha.

3. *Side-Channel*

Observar comportamento do computador

Loops de eventos compartilhados podem ser usados para criar canais ocultos, que não obedecem as políticas de segurança do navegador/SO.

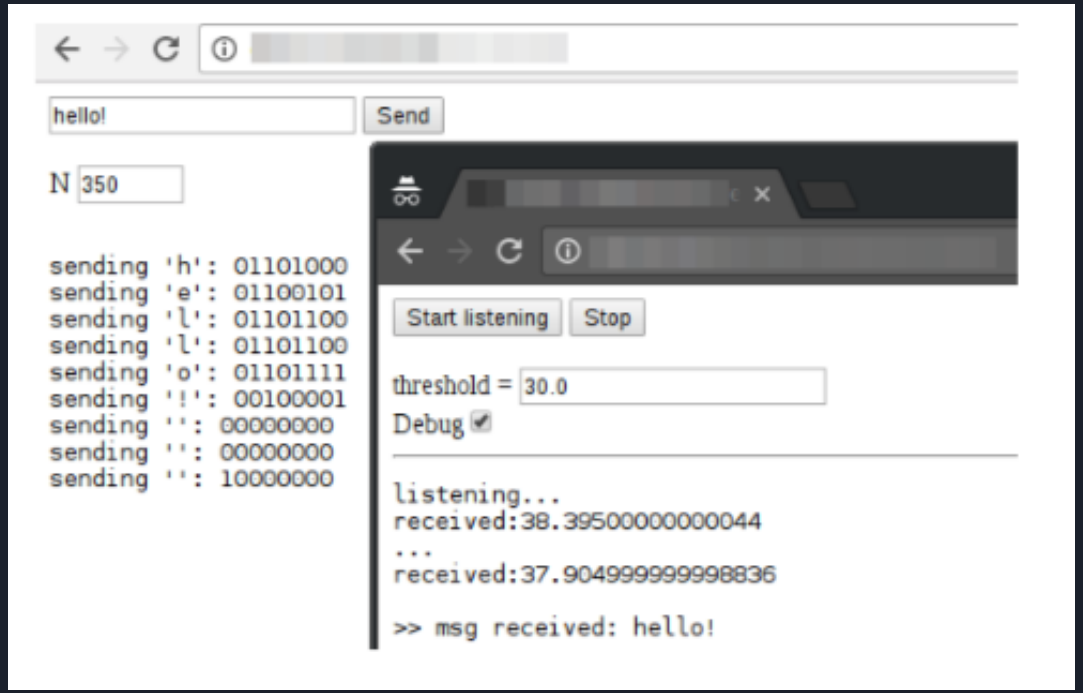
1. Processos de renderização

Implementamos um canal de comunicação para transmitir mensagens de uma página de remetente S para uma página de destinatário de origem cruzada R executando no mesmo processo de renderização.

Observamos que há vários canais ocultos alternativos para transmissão de informações entre páginas em execução no mesmo processador

3. *Side-Channel*

2. Processo de host: *Também implementamos um canal de comunicação para transmitir mensagens entre dois processos de renderizador cooperativo que compartilham o processo do host. A transmissão é unidirecional do remetente S para o receptor R.*



Canal secreto por meio do loop de eventos de E / S do processo de host do Chrome. Guias em diferentes processos de renderização (um deles navegando no modo de navegação anônima) se comunicam.

Por que Google Chrome?

1. Mais utilizado
2. Primeiro a implementar arquitetura multi-processos.

Em outros navegadores, é esperado um resultado similar em *side-channel*, pois também são orientados a eventos e possuem arquitetura similar.

CONSIDERAÇÕES FINAIS

- Evento de loops compartilhados no Chrome são vulneráveis a exploração.

- Os resultados da pesquisa foram enviados à equipe de segurança do Chromium, que decidiu não agir até o momento.

- O atacante pode ter diferentes propósitos, como a identificação de páginas da Web, a detecção do comportamento do usuário e a comunicação secreta.

REFERÊNCIAS

VILA, P., KOPF, B. **Loophole: Timing Attacks on Shared Event Loops in Chrome**. Technical University of Madrid (UPM)