Privilege separation Feb 22, 2021

Scribe: Stephan Kaminsky

Today's Topic is Design

- Announcements
  - First homework is on the website with some problems based on the problems given in the papers. Due a week from Wednesday.
- Paper review
  - Dummification idea
    - HTML, JS, etc is very complex to process. Renderer is just creating a simple bitmap. Kernels job is to just render the bitmap.
    - Very narrow interface for what is coming out of the renderer.
  - How is mouse input handled?
    - The renderer is given permission to receive UI events within certain bounds.
    - Renderer knows where links are so it knows how to map it back to a specific link.
  - Interesting topics
    - Focused on defending your machine from the web.
      - Two security principles: your local files, data, passwords, cookies, etc. Every website is potentially malicious.
      - Not trying to protect one website from another.
        - Malicious website cannot modify your files but could access other websites.
    - Alternative solution
      - Each website would have a separate kernel
      - Completely separate everything (separate copy of browser for each website)
        - Big overhead
        - Issues with compatibility
          - Subdomains might now work right
            - E.g. cookies might not be correctly shared
          - Google images would not work
            - Images would be from different domains so each image would require a new browser
          - Iframes
            - One is able to control another if nested (complicated to do in a secure way).
  - Reactions to the paper

        - We use IPC to talk outside the sandbox, why can't an attacker use a buffer overflow to attack that api?
        - The retrospective elements of what the benefit would have been if deployed 3 years ago.
        - Took the approach of mitigating/hardening of vulnerabilities instead of picking a single vulnerability and fixing it. They are only stopping only half of the issues.
            - Is fine because basically no performance or compatibility penalties in addition to patching half of the vulnerabilities.
    - Student Discussion
        - Both are an analysis of progress being made in security.
        - An evaluation of Google Chrome Extensions Security Architecture
            - Found many of the same vulnerabilities and mistakes being made in many extensions (in both big companies and small devs)
            - Key techniques to secure them
                - Privilege separation
                    - A security vulnerability in one partition should not affect another secure partition.
                    - Two types:

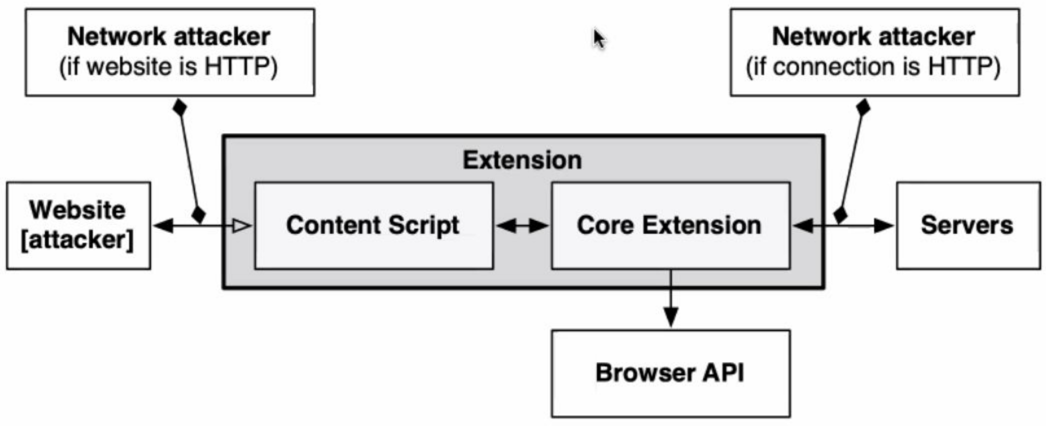

Figure 1: The architecture of a Google Chrome extension.

                        - Content scripts
                            - Injects js or CSS into page you are on
                        - Core Extension
                            - Interacts with chrome browser API
                - Isolated worlds
                - Permissions (Individual controls)
            - Talks about banning network requests made by extensions.
                - Would cause many extensions to not work

            - Value added does not equal the cost for adding it
        - Questions
            - Paper does not discuss malicious extensions
                - Steal cookies or keylogging
                - Great Extender was sold to a third party which then injected malicious code into it.
    - Some thoughts on security after ten years of qmail
        - Focused on emails
        - Main emphasis was to minimize bugs by decreasing code volume
        - Seems like pretty successful with very few security vulnerabilities.

Designing Systems

- Wireshark
    - Had a vulnerability where an attacker could send a special crafted packet which could exploit a vulnerability in it.
    - Solutions
        - Sandbox it
            - Still want to read packets on the network, perhaps it goes through some interface where packets can be filtered.
            - Inside sandbox will not affect processes outside of it.
            - Sandbox separate process per parser
        - Run it not as root, get the packets and process them not as root.
        - Does not need access to any other files but logs so it should not be given access to the general file system.
- SSH Server
    - You can log into another machine over ssh. Uses public key cryptography to authenticate. Connection signs with private key. It is critical that the private key stays private. How should we architect the server to protect the private key.
    - Solutions
        - Not allow root connections/hide private key behind root password.
        - Separate out the private key in a process which will do something to the data like sign it. So if the reset of the ssh code is corrupted, it cannot gain access to the private key even though it can still craft malicious replies.
            - Limitation of this is the attacker can still have access sign code.
            - What are the benefits?
                - They cannot sign anytime, only when your server is on.
- Secure Mail Client
    - Malicious attachment to gain control over your machine.
    - Open malicious email and the attacker gains control of your machine.
    - Attacker gains control over your mail client and then starts sending spam.
    - Solutions
        - Spam emails
            - In order to send an email, you sign it with a private key stored outside the mail client.

            - Attacker can only gain access to private key if they can gain root access or bypass separations
    - Malicious emails to send lots of emails
        - Renderers for all emails in own process
        - Have every time you want to make an outgoing network request, have to ask the user.
            - Downside, prevents emails on a timer.
        - Send button is handled by a trusted UI and is the only one who can issue network connections.
    - Malicious attachment
        - Trusted UI which walks through the filesystem/handles that.