UNVEILING VULNERABILITIES

SECURITY RISKS AND ATTACKS IN MODERN DESKTOP APPLICATIONS

Karthikeyan K

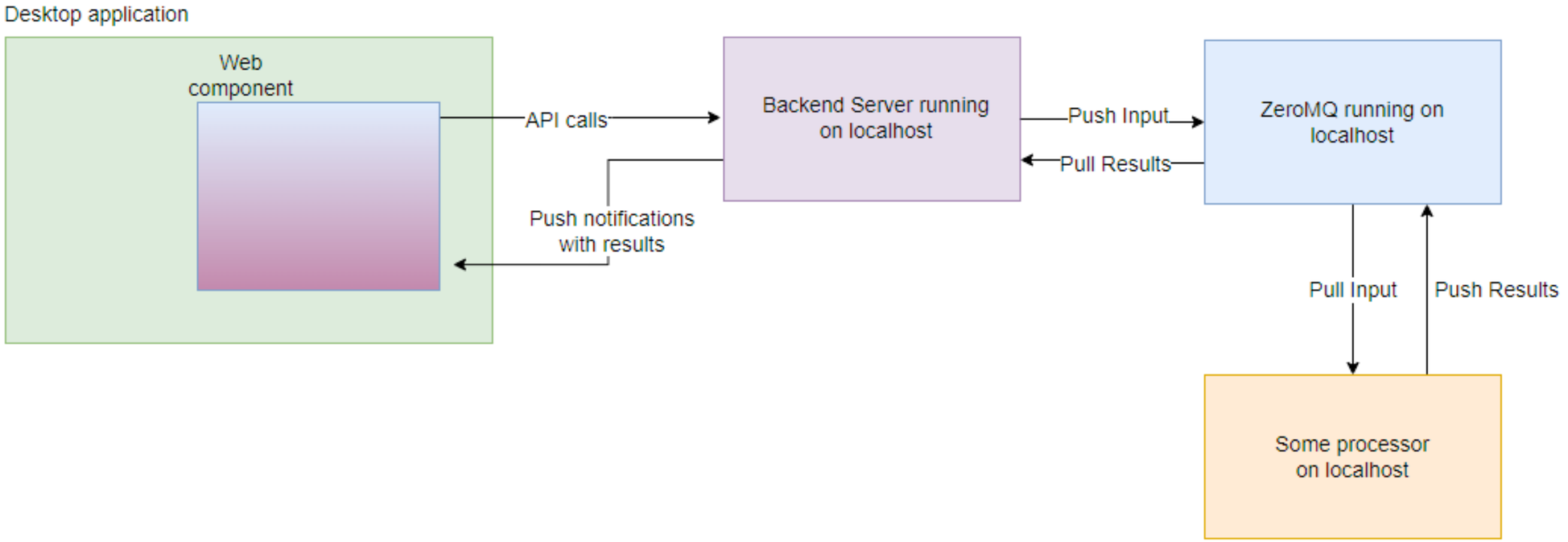
PRODUCT SECURITY@LOGITECH

Expertise areas : Pentesting,Application security, Cloud and cloud native,Devsecops,GRC and privacy

Certifications : CISSP,AWS Security,CEH,MCTS,MCP

OWASP and Defcon Chennai chapter leader, Public speaker

Thick client vs Thin client


Desktop application
Web component
API calls
Backend Server running on localhost
Push Input
Pull Results
ZeroMQ running on localhost
Push notifications with results
Pull Input
Push Results
Some processor on localhost

Tech stack

Front end

React js

Electron

Flutter

Back end

C++

Run as process

Communication

Websockets

gRPC

Windows name pipes | unix pipes

Storage

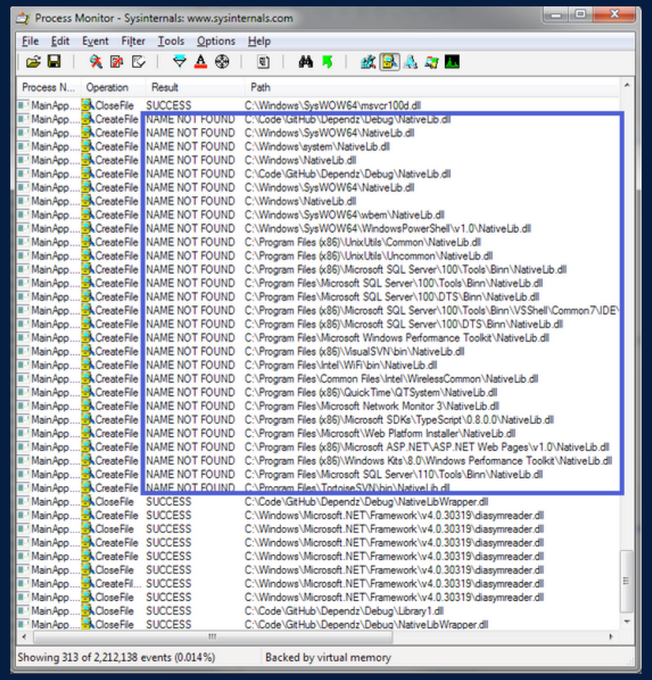
Local file system

S3 buckets

DLL Hijacking

Temp directory - Installer creates temporary executable files, then executes them during installation. These files are created with user writable permissions, meaning that a user, or a process running as a user can modify them, and execute instructions as root.

Unquoted service path - If a malicious individual has access to the file system, it is possible to elevate privileges to NT\System by placing a file such as "C:\Program.exe" and wait for/force a restart of the service or system.



Tools: Procmon

Symlink

The installer process with high privileges does not carefully check the junction and symlink,
and mistakenly takes the target pointed by the symlink as the creation object, which will trigger privilege escalation,
resulting in arbitrary directory creation and further causing a permanent system crash.

```
md C:\ProgramData\trap
mklink /J C:\ProgramData\App C:\ProgramData\trap
CreateSymlink.exe C:\ProgramData\trap\ups\220 C:\Windows\System32\cng.sys
```

Oauth

OAuth2.0 with PKCE has been implemented improperly which leads to csrf to login in desktop applications, this can be used for any desktop application that uses the endpoint,

State parameter is needed, this protects the app against CSRF where the same request can be relayed again to login server.

Scenerios:

The code-challenge is random but once a user initiates a login flow, it doesnt change even after a successful login, which means the old code challenge can be used again to get a valid login request for different accounts. This will log the person out of his actual account and connect his application to the attackers account who can now control it.

Problems we have noticed:

Lack of state param

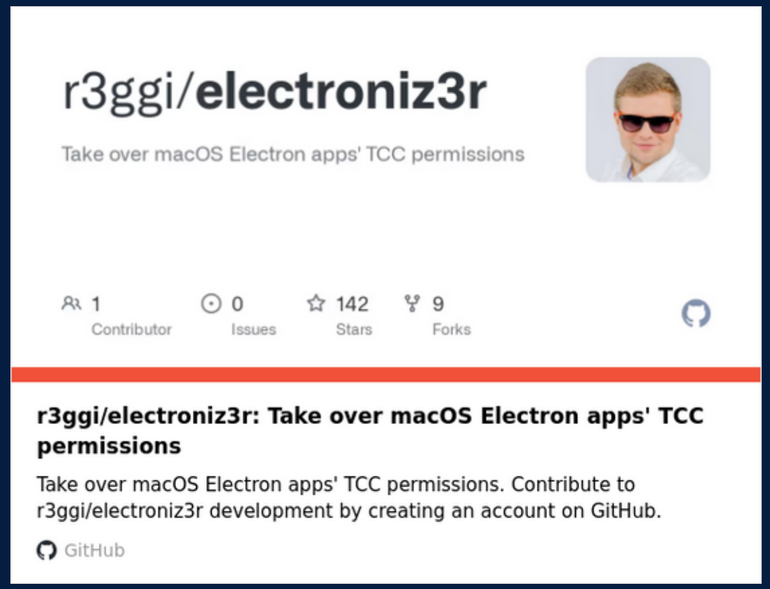
The code_challenge can be reused since the server side does not limit one session to one code_challenge. And the app uses code challenge as STATE because of its absence.

Electron

vulnerability allows attackers to compromise integrity. By exploiting this, unauthorized code can be injected into the product's processes, potentially leading to remote control and unauthorized access to sensitive user data.

ASAR - Atom shell archive format

% npx @electron/asar extract /Applications/test.app/Contents/MacOS/Contents/Resources/app.asar asar



Electron Fuses | Electron

Package time feature toggles

electronjs.org

Hardcoded Passwords/API keys

You can find juicy informations either in two or three places Like Memory, JS files and while reverse engineering the application

Thank's For Watching

Connect with Me.

+91 9789249246

https://www.linkedin.com/in/karthikeyan1337/