Lab 3 Threat Document

Dillon Staub, Will Hauber, John Miller, Matt Tucker

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Type | Element | Description | Impact |
| T1 | Spoofing | 2.1 Get address info | A hack app pretending to be Mapbox API and sends fake address data to “Get address info”. | The fake address data may contain harmful code and cause app crash. |
| T2 | Spoofing | 4.4 Create contact | A hack app pretending to “Fill in info in ContactManager” and sends fake location and contact data to “Create contact”. | The fake data may contain harmful code and cause app crash. |
| T3 | Spoofing | User | A hack app pretending to be Contact Manager and creates a fake contact (4.4) which is displayed to the User with either the correct or incorrect contact information as desired by the hacker’s programming. | The fake contact information may be a criminal’s desired meeting place with the User (criminal is a e.g. serial killer, con artist, etc.) instead of the location it should have been. |
| T4 | Tampering | 4.2 Open ContactManager | A snuck malware may modify features of “Open ContactManager”. | Something other than the intended application could be launched (spoof or otherwise). |
| T5 | Tampering | Data flow: from 2.1 Get address info to 2.2 Create marker | A snuck malware may intercept (hooking, etc.) the address info sent by “Get address info”, modify it, and send to “Create marker”. | Marker created at wrong location. |
| T6 | Tampering | Data flow: from 2.3 Apply style to marker to 2.4 Create popup with info | A snuck malware may modify information about which style is sent by 2.3 or alter the style itself. | The custom style may contain malicious code or use incompatible parameters which crash the application. |
| T7 | Repudiation | Data flow: Address location query (from 2.1 Get address info to Mapbox API) | The data that the user sends to Mapbox API could be modified by some attacker and they could deny this action. We do not have any logs that say what data the user actually sent so we have no way of proving there was an attack. | The app will receive incorrect information from the Mapbox API and the marker location may be wrong or may not be displayed at all. |
| T8 | Repudiation | Data flow: Location contact info (from 4.1 to 4.2 Open ContactManager) | The contact information could be modified by some attacker and they could deny this action. We do not have any logs that say what the original contact information was and therefore have no way of proving there was an attack. | The contact that is created will not have the correct information or perhaps no contact will be created at all. |
| T9 | Repudiation | User Settings | The user settings for privacy could be modified and we would have no way of proving an attack was made on these settings. | The user would expect their contact to have a private address and the address would not be private. |
| T10 | Information Disclosure | Data flow: Address location query (from 2.1 Get address info to Mapbox API) | A hack app pretending to be Mapbox API intercepts and takes queried address data. | The fake app could share information or post it online and would have information about the user’s recent address queries. |
| T11 | Information Disclosure | Data flow: Location contact info (from 4.1 to 4.2 Open ContactManager) | A hack app pretending to be ContactManager intercepts and takes contact information. | The fake app has access to the contact data and could share this contact info online. |
| T12 | Information Disclosure | Data flow: Address location query (from 2.1 Get address info to Mapbox API) | A hack app pretending to be Mapbox API intercepts and takes user’s current location. | The fake app could share information or post it online and would have information about the user’s possible location. |
| T13 | Denial of Service | Data flow: Address location query (from 2.1 Get address info to Mapbox API) | User address data could be modified in order to not allow the Mapbox API to return a valid address location. | The user would input a valid address but the data would be corrupted such that the Mapbox API does not return valid address information and no marker is displayed. |
| T14 | Denial of Service | 4.2 Open ContactManager | A hack app could consume all CPU time not allowing ContactManager to open when it is supposed to. | The user would expect ContactManager to open in order to create a contact but the app will never open. |
| T15 | Denial of Service | Data flow: from 2.3 Apply style to marker to 2.4 Create popup with info | A snuck malware swap out the Style sent from 2.3 to 2.4 with an invalid/corrupt one designed to cause feature malfunction. | When the corrupt style is loaded it would either prevent the marker from displaying correctly (not rendered) or crash the app, degrading the quality of service provided by the app |
| T16 | Elevation of Privilege | 5. Change privacy setting | A snuck malware hijack “Change privacy setting” process. | The hacker can change privacy settings without the user’s permission, allowing the hacker to freely see information the user had restricted access to |
| T17 | Elevation of Privilege | 4.1 Launch Mapbox contact page | A snuck malware could hijack 4.1 in order to launch a hack app that effects the Android system. | The hack app take over Android and perform actions sent to it by the hacker over the internet |
| T18 | Elevation of Privilege | 4.3 Create contact | A snuck malware changes how 4.3 displays the contacts to the user, sending hypnotic messages to the user by quickly flashing the screen on and off. | Through hypnosis, the hacker has elevated privilege to control the user’s mind directly, essentially obtaining all privileges held by the user. |