black hat ASIA 2020

OCTOBER 1-2, 2020

ARSENAL

Android Application Vulnerability Hunting System

By Todd Han, Junzhi Liu, Lilang Wu and Lance Jiang

Information Classification: General

Who we are?

Lilang Wu

- Security expert in Sangfor Technologies Inc.
- iOS, MacOS, and Android kernel vulnerability discovery and malware hunting.
- Speak at many conferences, including Black Hat USA 2018/2019, BlackHat EU 2018, HITB, CodeBlue and VirusBulletin etc..
- @Lilang_Wu, 574407955@qq.com

Todd Han

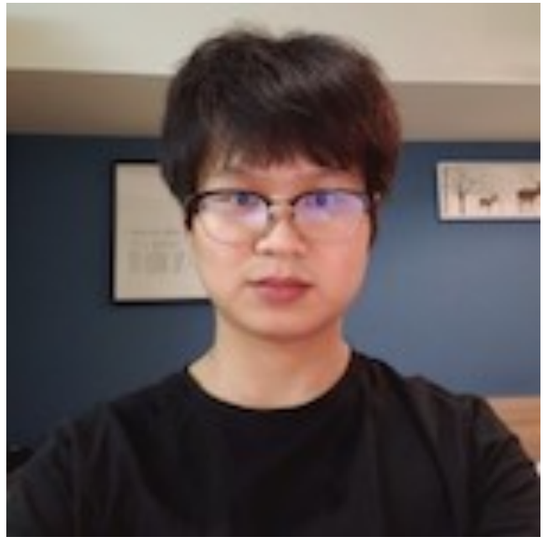
- Security Development Engineer in Sangfor Technologies Inc.
- Linux and Android kernel vulnerability discovery and malware hunting.
- Ever been spoken in conferences including CodeBlue 2018, HITB AMS 2019.
- @exia_han, exiahan@exiahan.com

Information Classification: General

Who we are?

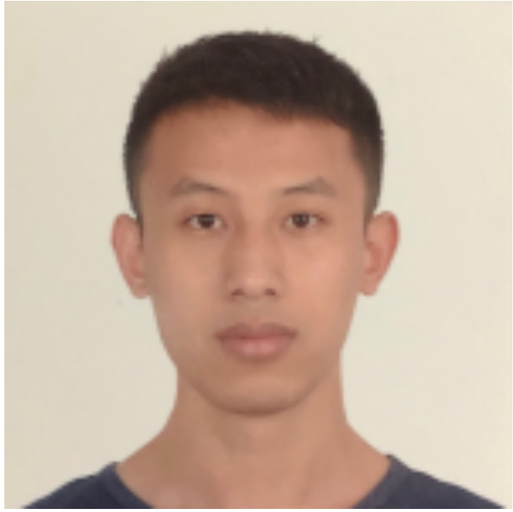
Junzhi Lu

- Security Researcher @ TrendMicro
- CFT Player
- HITB 2019 Speaker
- Hunting for XNU kernel and Browser vulnerabilities
- @pwn0rz



Lance Jiang

- Senior engineer from TopSec Inc.
- Focuses on mobile security research, Android kernel/apk vulnerabilities hunting
- HITB 2019 Speaker
- @Binas, https://github.com/Diamiracle



Information Classification: General

Agenda

- SAST Overview
- Vulnerabilities Study and Detection Patterns
- Meta Information Collection
- Detection Strategy Implementation base on Static Taint-Analysis
  - Common Checkers Implementation
  - Value-Sensitive Flow Analysis Implementation
  - Specific Vulnerabilities Detection Implementation
- Case Study

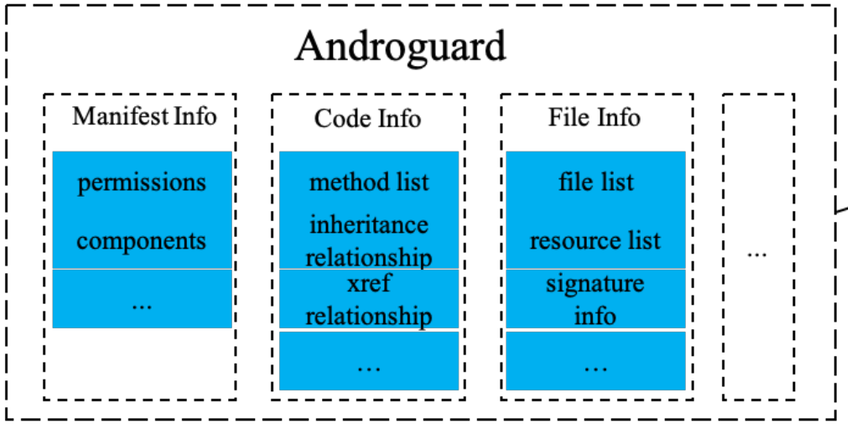
Information Classification: General

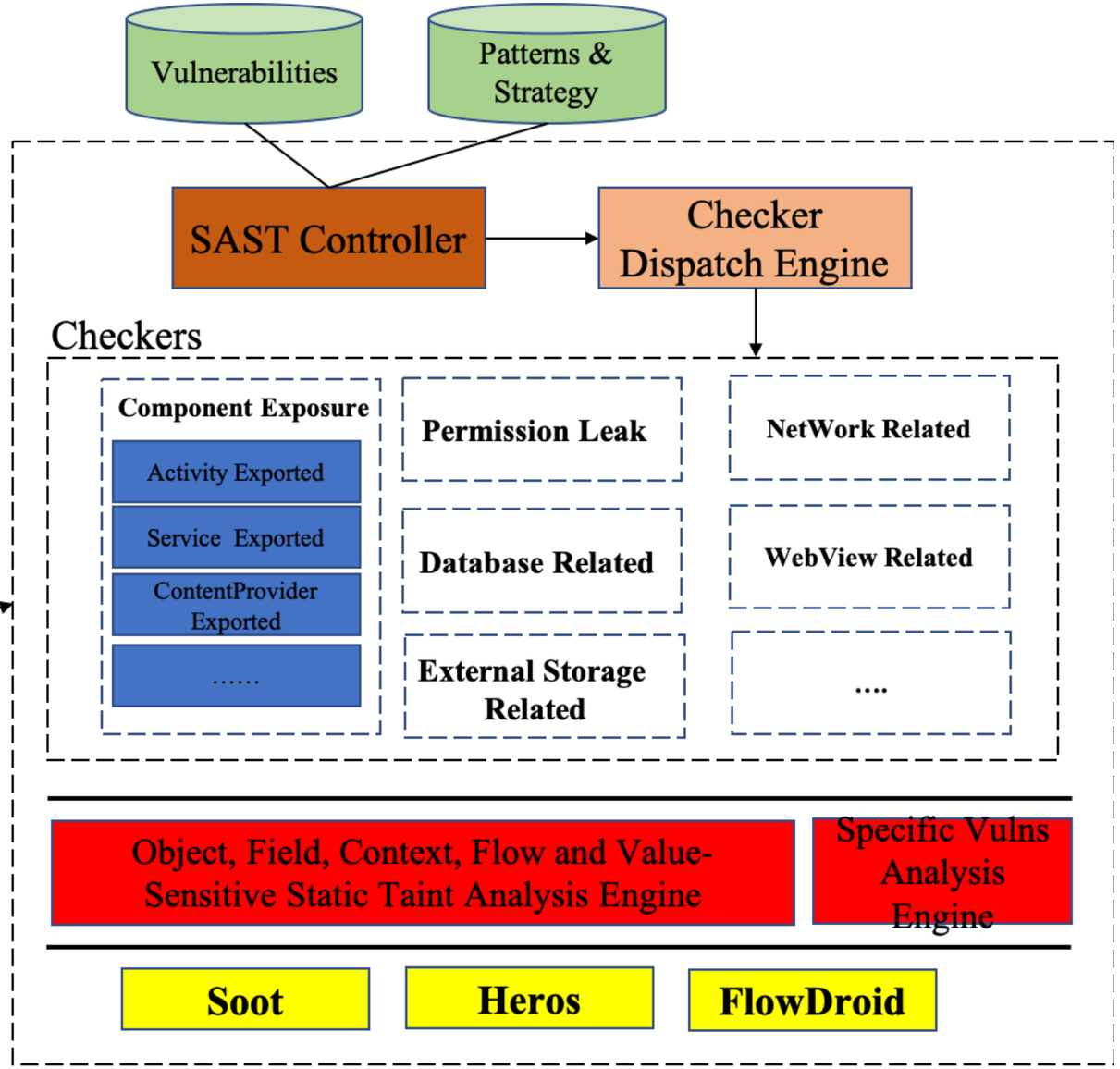
Agenda

- **SAST Overview**
- Vulnerabilities Study and Detection Patterns
- Meta Information Collection
- Detection Strategy Implementation base on Static Taint-Analysis
  - Common Checkers Implementation
  - Value-Sensitive Flow Analysis Implementation
  - Specific Vulnerabilities Detection Implementation
- Case Study

Information Classification: General

SAST Overview





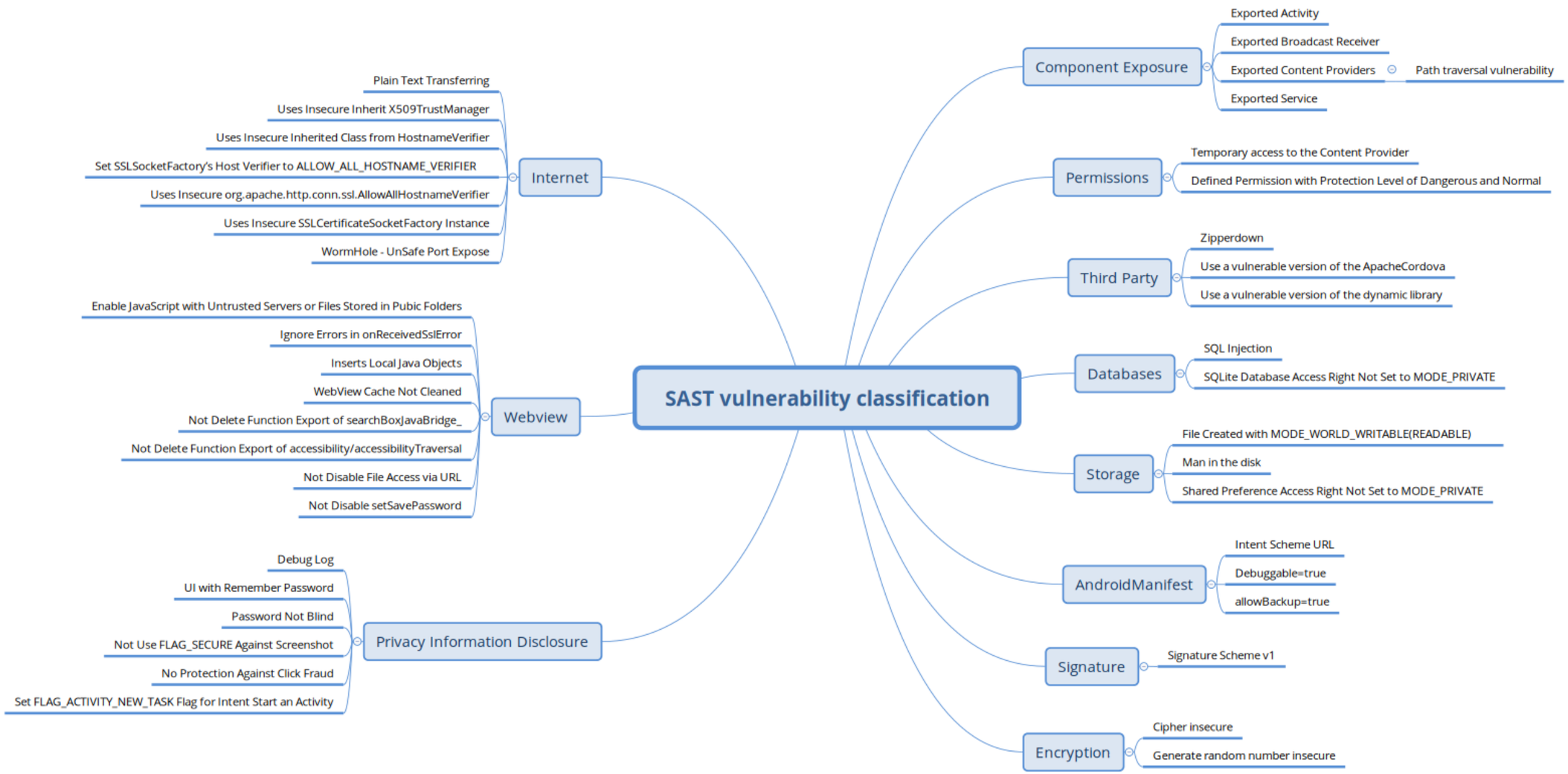
Information Classification: General

Agenda

- SAST Overview
- **Vulnerabilities Study and Detection Patterns**
- Meta Information Collection
- Detection Strategy Implementation base on Static Taint-Analysis
  - Common Checkers Implementation
  - Value-Sensitive Flow Analysis Implementation
  - Specific Vulnerabilities Detection Implementation
- Case Study

Information Classification: General

Vulnerabilities Classifies



Component Exposure Class

- Four component types maybe expose to risk if exported attribute is true
  - Common risk: allow others to access its private resources
  - For Content Provider:
    - Path Traversal
      - allow attackers to traversal private even system resources
    - SQL Injection
      - allow attackers to operate private SQLite database

Information Classification: General

Detect Patterns

```
<android.net.Uri: android.net.Uri parse(java.lang.String)> -> _SOURCE_
...
...
...
...
```

ContentProvider Exported

```
vulList ← [ ]
content_provider_list ← getContentProviderListFromManifest()
for cp in content_provider_list:
  if cp.isExported():
    source ← getSourcePoints("ContentProvider")
    sink ← getSinkPoints("ContentProvider")
    taint_value ← "../../../../etc/hosts"
    result_taint_value ← do_flow_analysis(cp.fullClassName, source, sink, taint_value)
    if result_taint_value == taint_value:
      vulList.add(cp)
return vulList
```

Source Sink Related Func

```
Func getSourcePoints(str Type) {
  file_path ← ROOT_DIR + type + _source.xml
  source_list ← parseXMLFile(file_path)
  return source_list
}

Func getSinkPoints(str Type) {
  file_path ← ROOT_DIR + type + _sink.xml
  sink_list ← parseXMLFile(file_path)
  return sink_list
}
```

```
<android.os.ParcelFileDescriptor: android.os.ParcelFileDescriptor open(java.io.File,int)> -> _SINK_
...
...
...
...
```

Information Classification: General

Detect Paterns

- Enhanced IFDS
- Init taint_value at source point
- Transform at every sensitive taint point we cares

do_flow_analysis

```
Func do_flow_analysis(class_name, source, sink, taint_value) {
  if current_component == class_name:
    sink_taint_value ← IFDS_alias_analysis(source, sink, taint_value)
  return sink_taint_value
}

Func IFDS_alias_analysis(source, sink, taint_value) {
  [Nornal_IFDS_Work_Flow]
  ...
  switch invoked_function_name {
    case replace:
      args ← getInvocationArgs(current_expr)
      new_taint_value ← taint_value.replace(args[0], args[1])
    case parseURI:
      ...
    case substring:
      ...
    ...
  }
  ...
  [Normal_IFDS_Work_Flow]
}
```

Information Classification: General

Permissions Class

- App contains some dangerous permissions or user-defined permissions which protectionLevel is dangerous
  - Common risk: Allow other apps access some sensitive data.
  - For Content Provider
    - Temporary Access
      - Temporary access permission like FLAG_GRANT_READ_URI_PERMISSION may cause sensitive information leaked to attackers

Information Classification: General

Detect Patterns

	Temporary access to the Content Provider
1	vulList ← [ ]
2	permissionLst ← read AndroidManifest.xml
3	for p in permissionList:
4	if p is grantUriPermission:
5	vulList.add(p)
6	codeRef ← xref(addFlags) + xref(setFlags)
7	for ref in codeRef:
8	arg ← getArg(ref)
9	if arg is FLAG_GRANT_READ/WRITE_URI_PERMISSION:
10	vulList.add(p)
11	return vulList

Database Class

- Using database with insecure permission config no string filter
  - Common Risk: Data leaking or data corrupt
    - SQL Injection
      - Query processing function leak of string filtering action may cause SQL Injection
    - Wrong Permission
      - Database created without MODE_PRIVATE will give other app change to access database

Database Class

SQL Injection

```
vulList ← [ ]
sensitive_sql_functions ← [execSQL, rawQuery, buildQueryString, buildQuery, etc]
sensitive_sql_invocation_list ← [ ]
SQL_TAINT_VALUE ← "'administrator'--' AND password = ''"

for q in sensitive_sql_functions:
  tmp_xref_list ← xrefUntilTop(q)
  for i in tmp_xref_list:
    correspondClass ← getCorrespondingComponent(i)
    if correspondClass:
      sensitive_sql_invocation_list.add(correspondClass)
for i in sensitive_sql_invocation_list:
  source ← getSourcePoints("SQL")
  sink ← getSinkPoints("SQL")
  taint_value ← SQL_TAINT_VALUE
  result_taint_value ← do_flow_analysis(i.fullClassName, source, sink, taint_value)
  if result_taint_value == taint_value:
    vulList.add(i)
return vulList
```

```
// Function to get root node of an invocation chain
Func xrefUntilTop(ref) {
  ...
  [Xref Until Reach The Root Function of Invocation Chain]
  ...
  return topRef
}
// Function to get the class if the caller belongs to a
// Components[Activity, ContentProvider, Service, Receiver]
Func getCorrespondingComponent(invocation) {
  ...
  [Check Whether Invocation belongs to a component]
  ...
  [Return None if not, or return corresponding class]
}
```

Storage Class

- File in wrong place or with wrong permissions may lead risks
  - Common Risk: Causing unexpected file changing or reading by attackers
  - Common Vulnerabilities
    - Wrong Permission like MODE_WORLD_WRITABLE
      - Anyone can access file created with permission mode_world_writable
    - Man in the disk
      - File which lied in external storage are always facing tamper risks

Information Classification: General

Storage Class

File Created Unsafe

```
vulList ← [ ]
xrefCode ← xref(openFileOutput)
for ref in xrefCode:
    arg ← getArg(ref)
    if MODE_WORLD_WRITABLE/READABLE in arg:
        vulList.ad(ref)
        return vulList
```

Man in the disk

```
vulList ← [ ]
sensitiveFunctions ← [File, createFile, etc]
sensitiveStrings ← ['sccard0', 'emulated/0', 'storage/sdcard1', etc]
sensitivePathFunctions ← ['getExternalStoragePublicDirectory', 'getExtenalStorageDirectory',
'getExternalFilesDir']
strRef ← [ ]
for i in sensitiveFunctions:
    codeRefs ← getCodeRef(i)
    for ref in CodeRefs:
        arg ← getArg(ref)
        if arg in sensitiveStrings:
            vulList.add(ref)
        else
            if arg is return from sensitivePathFunctions:
                vulList.add(ref)
return vulList
```

Information Classification: General

Encryption Class

- Weak Cipher/Hasher and wrong random generator usage may lead to risks
  - Common Risks: Give attackers chances to decrpyt or guess the plain text or the next random value
  - For cipher/hahser
    - Weak cipher/hasher algorithms which are easy to decryption or guess used in apps
  - For random generator
    - Using random generator with weak seeds like some guessable constants

Encryption Class

	hardcodedKeyExtraction
1.	hardcodedKeyList = {}
2.	**Foreach** item **in** stringsList:
3.	**if** item **matches in** regexPatternSet{RSA , API_KEY,TOKEN ,…}
4.	add item to hardcodedKeyList
5.	**Foreach** {func,argIndex, keyPropagateRule} **in** cipherFuncChecks:
6.	constants = extractConstantParams(func, argIndex, keyPropagateRule)
7.	append constants to hardcodedKeyList
8.	return hardcodedKeyList

```
_regex = {
    'google_api'      : r'AIza[0-9A-Za-z-_]{35}',
    'firebase'   : r'AAAA[A-Za-z0-9_-]{7}:[A-Za-z0-9_-]{140}',
    'google_captcha' : r'6L[0-9A-Za-z-_]{38}|^6[0-9a-zA-Z_-]{39}$',
    'google_oauth'    : r'ya29\.[0-9A-Za-z\-_]+',
    'amazon_aws_access_key_id' : r'AKIA[0-9A-Z]{16}',
    'amazon_mws_auth_toke' : r'amzn\\.mws\\.[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}',
    'amazon_aws_url' : r's3\.amazonaws.com[/]+|[a-zA-Z0-9_-]*\.s3\.amazonaws.com',
    'amazon_aws_url2' : r"(" \
           r"[a-zA-Z0-9-\.\_]+\.s3\.amazonaws\.com" \
           r"|s3://[a-zA-Z0-9-\.\_]+" \
           r"|s3-[a-zA-Z0-9-\.\_\/]+" \
           r"|s3.amazonaws.com/[a-zA-Z0-9-\.\_]+" \
           r"|s3.console.aws.amazon.com/s3/buckets/[a-zA-Z0-9-\.\_]+)",
    'facebook_access_token' : r'EAACEdEose0cBA[0-9A-Za-z]+',
    'authorization_basic' : r'basic [a-zA-Z0-9=:_\+\/-]{5,100}',
```

	extractConstantParams
1.	**ForEach** edge **in** InverseICFG:
2.	**If** edge.isAssignment lhs = rhs:
3.	**If** lhs.alias in TrackPoint and rhs is constant:
4.	add rhs to FoundList, kill lhs in TrackPoint
5.	**ElseIf** lhs,alias in TrackPoint and rhs is value:
6.	gen rhs and kill lhs in TrackPoint
7.	**If** edge.isCallStmt:
8.	**If** local_var.alias in TrackPoint:
9.	apply function back propagate rule from local_var to param
10.	If edge.isReturnStmt:
11.	If ret_var.alias in TrackPoint
12.	back propagate ret value to local variable

Information Classification: General

Networking Communication Class

- Communicating with wrong configed https or http may lead to risks
  - Common Risks: Data tansfered may be monitored by attackers
  - For HTTPS
    - Wrong HostName Verifier Usage
      - This will trust all https sites including sites which have wrong certs
    - Use getInsecure
      - Using of getInsecure will lose the advantages https provides

Networking Communication Class

	AllowAllHostnameVerifier Wrong Usage Detection
1.	vulList ← [ ]
2.	unsafeFunctions ← [AllowAllHostnameVerifier, etc]
3.	for i in unsafeFunctions:
4.	codeRefs ← xref(i)
5	for ref in codeRefs:
6	vulList.add(ref)
7	return vulList

	getInsecure Wrong Usage Detection
1.	vulList ← [ ]
2.	codeRefs ← xref(getInsecure)
3.	for ref in codeRefs:
4.	vulList.add(ref)
5.	return vulList

Networking Communication Class

InsecureHTTPSVerifer

```
vulnList = {}
constants = extractConstantParams(setHostNameVerifier,1,insecureHTTPVerifierPropagateRule)
Foreach c in constant:
    flag = true
    Foreach stmt in c.getMethod('verify').Stmts():
        if stmt is conditionalStmt or stmt contains callStmt:
            flag = false
    if flag:
        add c to vulnList
return vulnList
```

extractConstantParams

```
ForEach edge in InverseICFG:
    If edge.isAssignment lhs = rhs:
        If lhs.alias in TrackPoint and rhs is constant:
            add rhs to FoundList, kill lhs in TrackPoint
        ElseIf lhs,alias in TrackPoint and rhs is value:
            gen rhs and kill lhs in TrackPoint
    If edge.isCallStmt:
        If local_var.alias in TrackPoint:
            apply function back propagate rule from local_var to param
    If edge.isReturnStmt:
        If ret_var.alias in TrackPoint
            back propagate ret value to local variable
```

Information Classification: General

WebView Class

- Enabling insecure components and wrong config
  - Common Risks: Wrong config of WebView and insecure components enabling can cause secure issues.
  - JS Related
    - addJavaScriptInterface
      - Add javaScriptInterface to webview give attackers chance to invoke local java methods
  - SSL Related
    - Unsafe SslErrorHandler
      - Wrong implementation of SslErrorHandler may cause user visit untrusted sites

Information Classification: General

WebView Class

	addJavascriptInterface Wrong Usage Detection
1.	vulList ← [ ]
2.	targetAPI ← parsing AndroidManifest.xml
3.	if targetAPI >= 17:
4.	return vulList
5.	codeRefs ← xref(addJavascriptInterface)
6.	for ref in codeRefs:
7.	vulList.add(ref)
8.	return vulList

	SslErrorHandler Wrong Usage Detection
1.	vulList ← [ ]
2.	codeRefs ← xref(SslErrorHandler.process)
3.	for ref in codeRefs:
4.	caller ← getCaller(ref)
5.	if caller == onReceivedSslError:
6.	vulList.add(ref)
7.	return vulList

Privacy Leakage Class

- Unsafe config for components and debug code remains in release lead to risks
  - Common Risks:
  - For Activity:
    - Screenshot allowed
      - Allow screenshot on activities which contain sensitive data may cause data leaking
  - For Log:
    - Debug log remains
      - Log infos in release version may expose sensitive data to attackers

Information Classification: General

Privacy Leakage Class

	Screenshot Risk Detection
1.	vulList ← [ ]
2.	Password_EditText_Related_Activity_List ← analyzing layout.xml files
3.	for ac in Password_EditText_Related_Activity_List
4.	func ← getOnCreate(ac)
5.	arg ← getArgOf(setFlag/addFalg)
6.	if arg & FLAG_SECURE is 0:
7.	vulList.add(ac)
8.	return vuLList

	Debug Log Risk Detection
1.	vulList ← [ ]
2.	logCodeRefs ← xref(android.util.log)
3.	printCodeRefs ← xref(printStream)
4.	codeRefs ← logCodeRefs + printCodeRefs
5.	for ref in codeRefs:
6.	vulList.add(ref)
7.	return vulList

Misc Class

- Some misc part which will also cause secure isues
  - For ApkFile:
    - Janus
      - Apk with only jar sign is under attack of Janus
    - Old third party libraries
      - Apk contains third part libraries which in old version may be under attacks of known vulnerabilities

Information Classification: General

Misc Class

	Signature Scheme v1
1	sigScheme ← getSigScheme(apk)
2	if sigScheme is jarSign:
3	return True

	Use a vulnerable version of the dynamic library
1	vulList ← [ ]
2	knownVuls ← [COLLECTED KNOWN VULS MAP]
3	soFiles ← getLibries(apk)
4	for so in soFiles:
5	if knownVuls[so.version] not Empty:
6	vulList.add(so)
7	return vulList

Agenda

- SAST Overview
- Vulnerabilities Study and Detection Patterns
- **Meta Information Collection**
- Detection Strategy Implementation base on Static Taint-Analysis
  - Common Checkers Implementation
  - Value-Sensitive Flow Analysis Implementation
  - Specific Vulnerabilities Detection Implementation
- Case Study

Information Classification: General

Meta Information List

Meta Info	Related Vulnerabilities	Related Interfaces in AndroGuard
XMLs(Include AndroidManifest and Resources)	Components Exposure Privacy Leaking Incorrect Permissions	APIs in androarsc_main
Smali Code References	Components Exposure Incorrect Permissions, Unsafe Database Config Insecure Storage Operation Insecure Encryption Solution, Insecure Networking Communication Unsafe Webview Config Privacy Info Leaking	Xref related APIs, Class/method analysis related APIs, Instructions related APIs
Package Info(Certs, etc)	MISC Janus	Signature related APIs
.so files in /lib/	MISC Third-party vuln	-

Information Classification: General

Best Practice

- Add API Wrappers for easy rule writing(1/2)

```
def getAllResString(a)
def isApiUsedWithString(d, x, s)
def getSystemField(x)
def isApiDeclared(d, x, className, methodName, descriptor)
def isApiUsed(d, x, className, methodName, descriptor)
def isApiUsedWithMehtodDescriptorReturn(d, x, className, methodName,
def isApiUsedWithParameterExact(d, x, className, methodName, descrip
def isApiUsedWithParameterFlag(d, x, className, methodName, descript
def isApiUsedWithParameterString(d, x, className, methodName, descri
def isNotApiUsedWithParameterExact(d, x, className, methodName, desc
def isNotApiUsedWithParameterFlag(d, x, className, methodName, descr
def isNotApiUsedWithParameterString(d, x, className, methodName, des
def isApiUsedAfterApi(d, x , className1, methodName1, descriptor1, c
def isInheritedFrom(ds, subClassName, superClassName)
def getApiParameter(d, x, className, methodName, descriptor, paramNo
def findInheritedClassesFrom(ds, superClassName)
def true1false0empty2(booleanStr)
def formatClassName(className)
def convertClassName(classname)
def formatClassMethodName(a, className, methodName=None)
def isStringExists(x, manifest, allresString, isStartsWithNotContain
def isHttpGetPost(httpStr)
```

```
def isApiUsed(d, x, className, methodName, descriptor):
    descriptorRegEx = __transform2RegEx(descriptor)
    usedApiList = []
    methods_generator = x.find_methods(className, methodName, descriptorRegEx)
    classFilters = [
        "Landroid/support/.*",
        "Landroidx/.*",
        "Lcom/google/.*",
        "Lcom/facebook/.*",
        "Lcom/twitter/.*",
        "Lcom/amazon/.*",
        "Lcom/amazonaws/.*",
        "Lorg/apache/.*"
    ]
    for method_analysis in methods_generator:
        for xref_from_class, xref_from_method, caller_offset in method_analysis.get_xref_from():
            if xref_from_method is not None:
                class_name = xref_from_class.orig_class.get_name()
                inWhiteList = False
                for classFilter in classFilters:
                    if re.match(classFilter, class_name) is not None:
                        inWhiteList = True
                        break
                if not inWhiteList:
                    usedApiList.append((xref_from_method.get_class_name(), xref_from_method.get_name()));
    return usedApiList;
```

Best Practice

- Add API Wrappers for easy rule writing(2/2)

```
23  def typeMap(javaType)
24  def getClass(ds, className)
25  def getComponentAttr(a, componentName, attrName)
26  def isAttrValInComponent(a, componentName, attrName, attrV
27  def isStringUsed(x, s)
28  def isMethodUsed(x, className, methodName, descriptor)
29  def testField(d, x)
30  def findAllWebViewDef(a, d, x)
31  def FoundPWEditor(a)
32  def get_activities(a)
33  def GetSubStrInfo(ThisTString, StartFlag, EndFlag, StartPo
34  def GetSubStr(ThisTString, StartFlag, EndFlag, StartPos=0)
35  def isHttpSend(httpStr)
36  def isApiUsedWithParaOutside(d, x, className, methodName,
37  def isStringExistsInDex(x, isStartsWithNotContains, target
38  def isStringExistsInAxml(manifest, targetStr)
39  def isApiUsedOutside(d, x, className, methodName, descript
40  def isStringExistsInDexShadow(x, isStartsWithNotContains,
41  def findVersionString(x, targetStr)
42  def removeThirdPart(class_list)
43  def getAllTopComponents(a)
44  def getTopComponents(x, known_components, target_method_an
```

```
383  def getApiParameter(d, x, className, methodName, descriptor, paramNo):
384      usedApiList = []
385      classFilters = [
386          "Lcom/amazonaws/.*",
387          "Landroid/support/.*",
388          "Lcom/google/.*",
389          "Lcom/facebook/.*",
390          "Lcom/twitter/.*",
391          "Lcom/amazon/.*",
392          "Lorg/apache/.*",
393          "Landroidx/.*",
394      ]
395      descriptorRegEx = __transform2RegEx(descriptor);
396      #paths = x.find_methods(className, methodName, descriptorRegEx);
397      methods_generator = x.find_methods(className, methodName, descriptorRegEx);
398      # if len(paths) < 1:
399      #     return usedApiList;
400
401      # pathIdxSet = set();
402      for method_analysis in methods_generator:
403          # if method_analysis.is_external():
404          #     continue
405          # if path.get_src_idx() in pathIdxSet:
406          #     continue;
407          # pathIdxSet.add(path.get_src_idx());
408          # method = method_analysis.get_method()
409          # instructions = method.get_instructions();
410          # instructionList = list(instructions);
411          for xref_from_class, xref_from_method, caller_offset in method_analysis.get_xref_from():
412              if xref_from_method is not None:
413                  instIdx, vRegNum = __findMethodInstructionIdxParamVRegNum(xref_from_method, className, methodName, descriptorRegEx, paramNo);
414                  if vRegNum < 0:
```

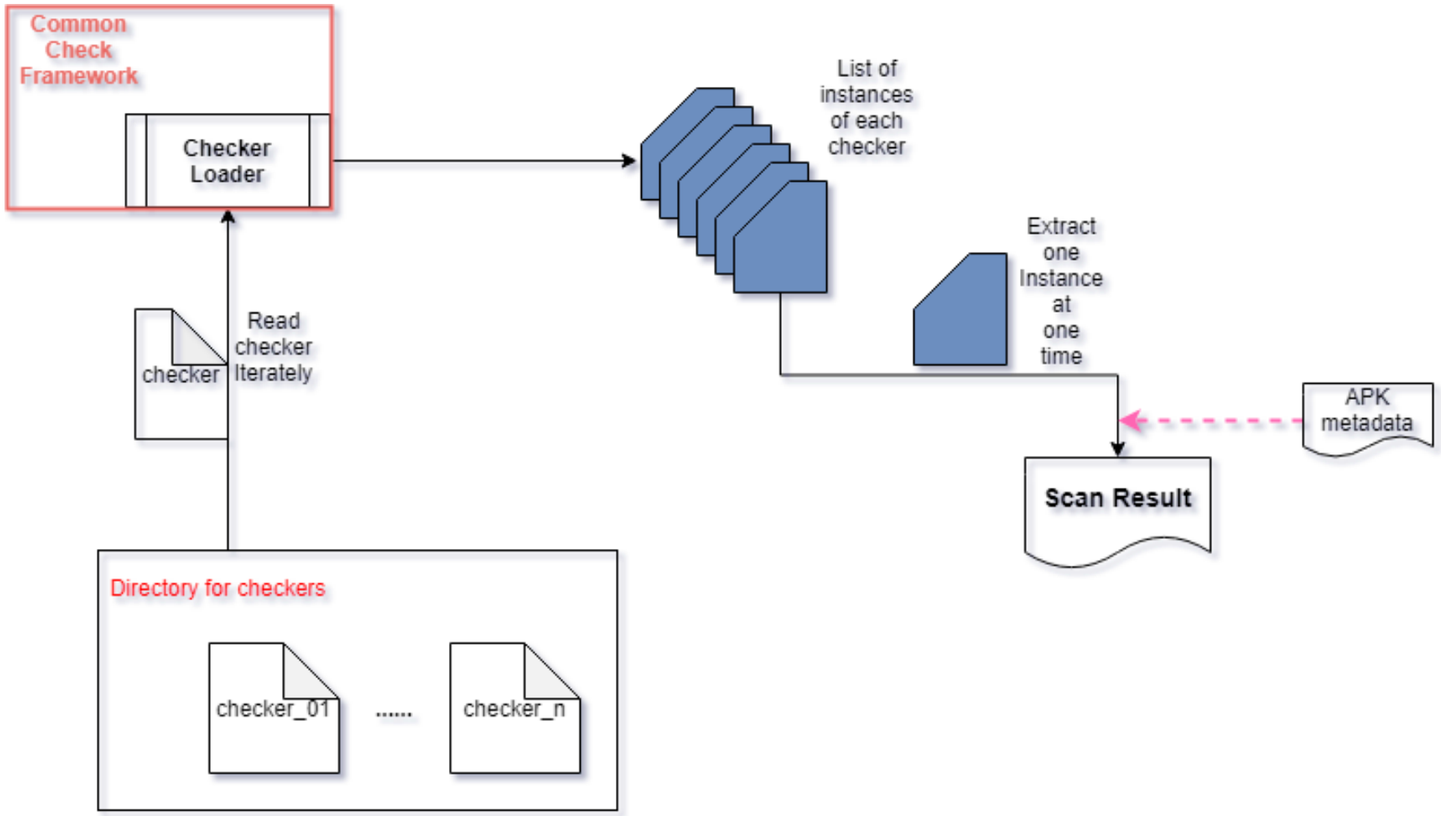
Agenda

- SAST Overview
- Vulnerabilities Study and Detection Patterns
- Meta Information Collection
- **Detection Strategy Implementation base on Static Taint-Analysis**
  - **Common Checkers Implementation**
  - Value-Sensitive Flow Analysis Implementation
  - Specific Vulnerabilities Detection Implementation
- Case Study

Information Classification: General

Common Checkers Architecture

- Working Flow
  - Traversing folder
  - Load each checker into list
  - Do scan with metadata
  - Flow analysis if necessary
  - Deliver result



Structure of Checkers

- Each checker is a plugin
  - Unique TYPE Define
    - PRIMARY_TYPE
    - SECONDARY_TYPE
  - A class Inherit a common class called "Checkers" with below 2 fields
    - A Filed called SCAN_PRIORITY
    - A common function called "scan(self)"

Information Classification: General

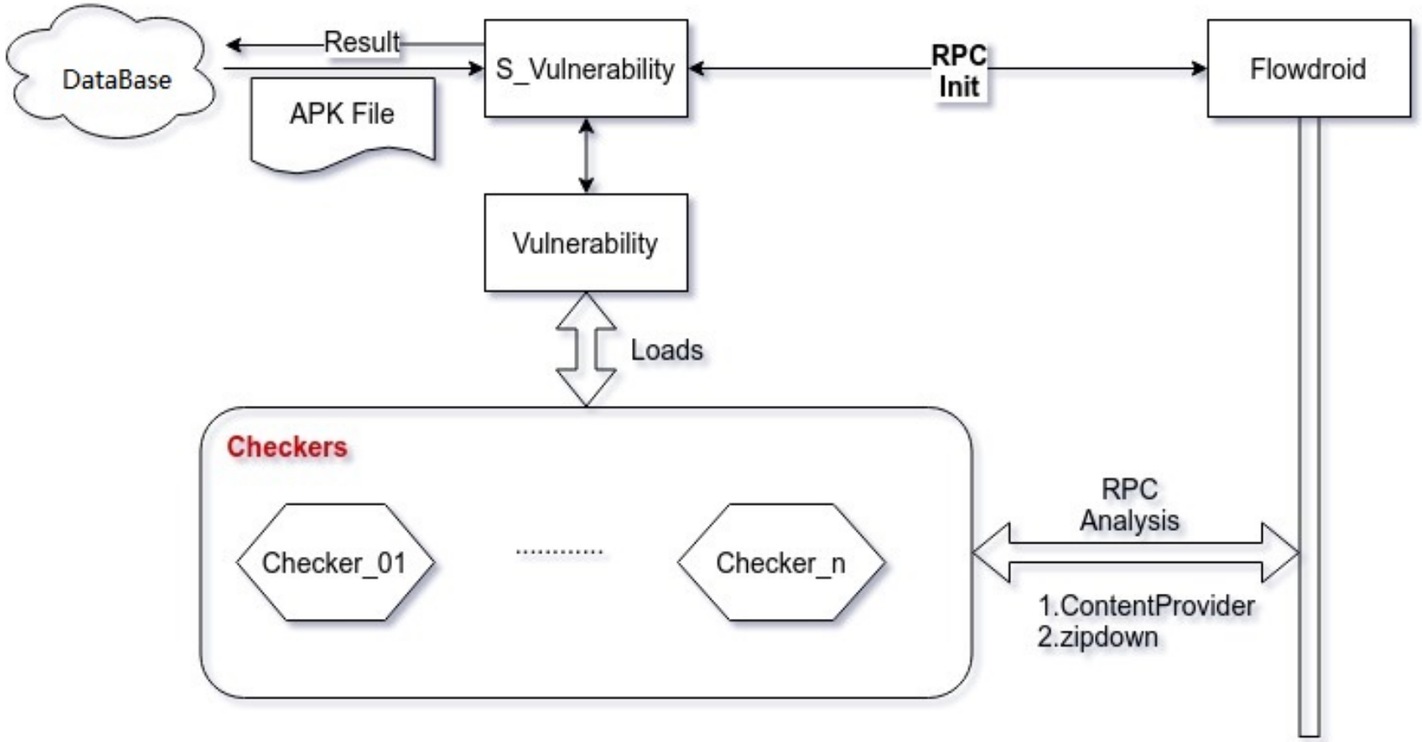
Agenda

- SAST Overview
- Vulnerabilities Study and Detection Patterns
- Meta Information Collection
- **Detection Strategy Implementation base on Static Taint-Analysis**
  - Common Checkers Implementation
  - **Value-Sensitive Flow Analysis Implementation**
  - Specific Vulnerabilities Detection Implementation
- Case Study

Information Classification: General

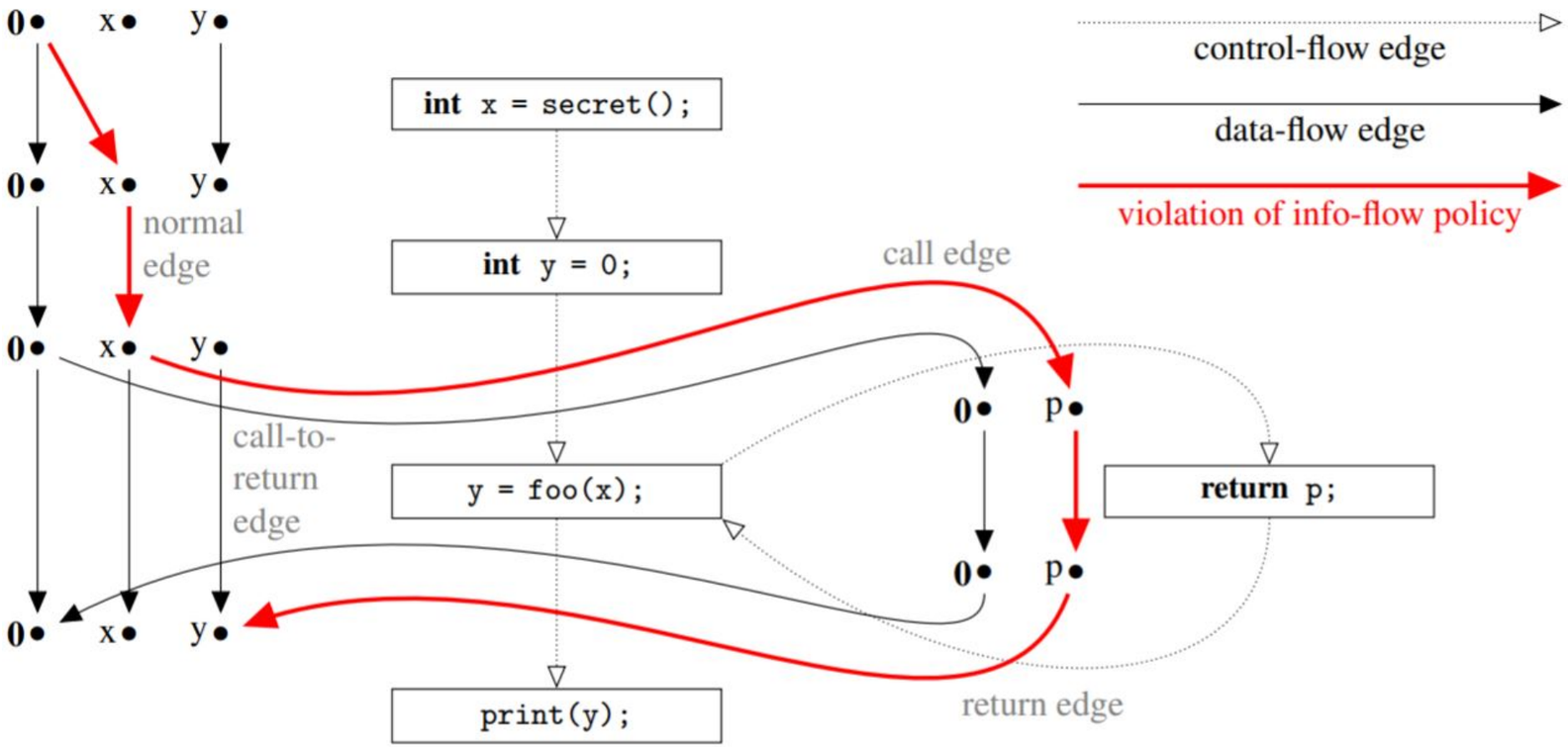
Flow Analysis Architecture

- Seperated with common check
- Only invoke if needed



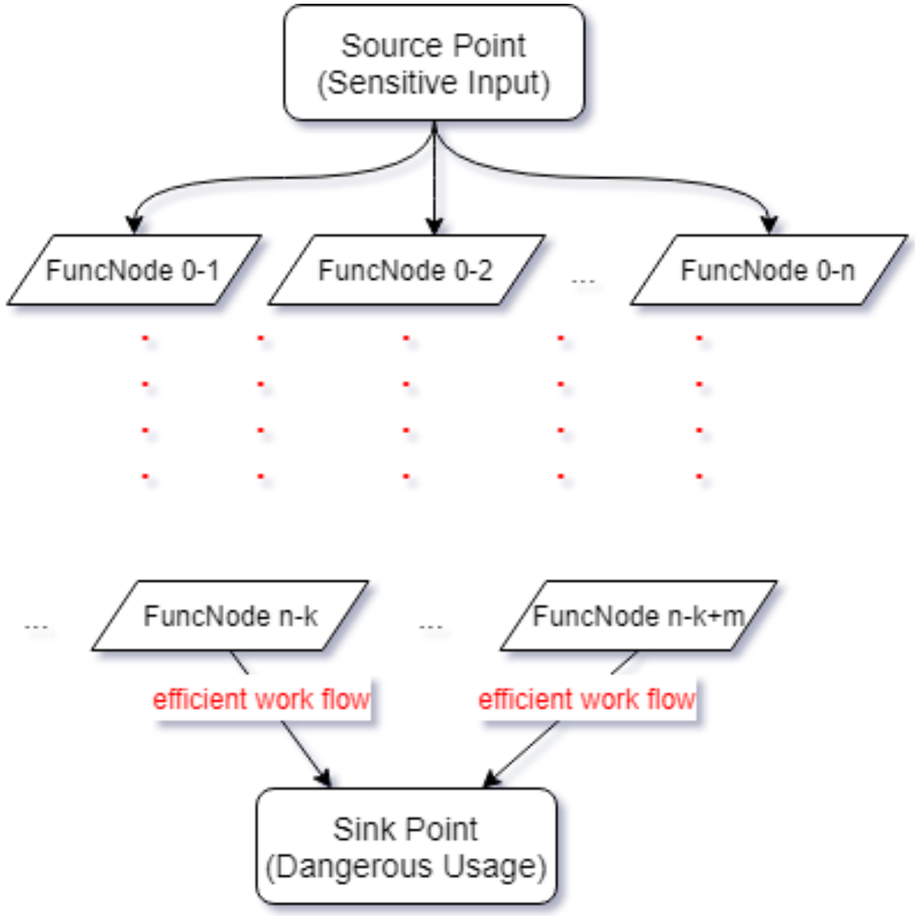
Flow Analysis Internal

- Based on Flowdroid (IFDS)
  - Tracing the value from source to sink (Alias Analysis) -- *Pic from flowdroid paper*



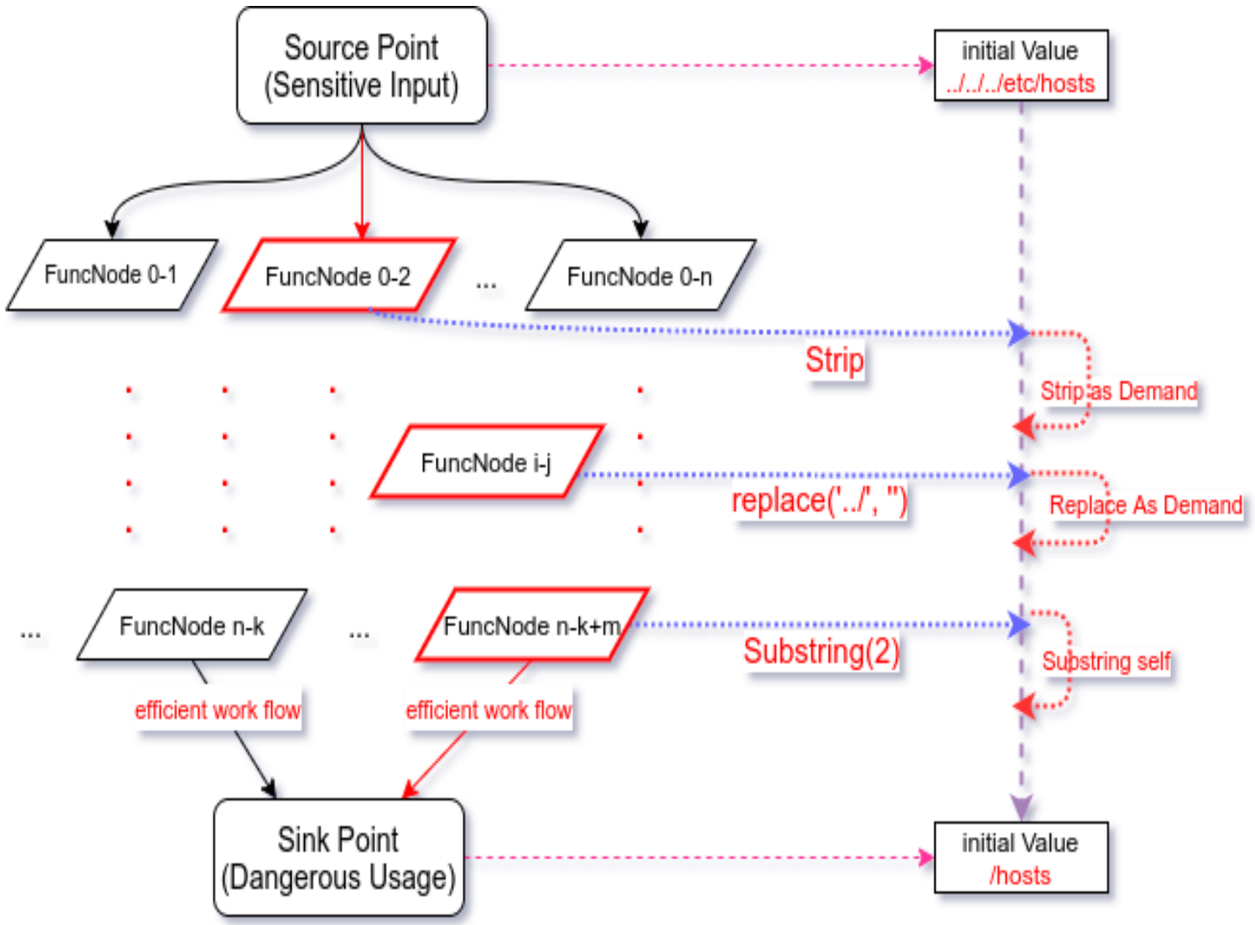
Flow Analysis Internal

- Source to Sink
  - Source: Where sensitive value retrieved
  - Sink: Where sensitive value processed
  - Ability of value sensitive



Flow Analysis Internal

- Source to Sink With Value Sensitive
  - Source: Where sensitive value retrieved
  - Sink: Where sensitive value processed
  - Value
    - Changes if necessary from source to sink



Best Practice

- What we do?
  - 1. we add a field to set the value and keep tracking the value
    - Where the TaintValue is just a wrapper of string but provides some useful extensions

```
public class AccessPath implements Cloneable {

...
    private TaintValue taintValue;
...
}
```

Best Practice

- What we do?
  - 2. We modified code in SetupApplication.java to only process EntryPoint we cared.
  - 3. In “scanMethodForSourcesSinks”, when the forwardprolem slover init the seed, we init our TaintValue corresponding to current scenario, aka what kind of vulnerability we are try to dig.

```
PatchingChain<Unit> units = m.getActiveBody().getUnits();
for (Unit u : units) {
    Stmt s = (Stmt) u;
    if (sourcesSinks.getSourceInfo(s, manager) != null) {
        forwardProblem.addInitialSeeds(u,
            Collections.singleton(forwardProblem.zeroValue()));
        TaintValue oldTaintValue = u.getDefBoxes().get(0).getTaintValue();
        TaintValue newTaintValue = null;
        if (oldTaintValue != null) {
            newTaintValue = TaintFactory.generateTaintValue(
                TGlobal.getTaintType(), oldTaintValue.getValue());
        }
        else {
            newTaintValue = TaintFactory.generateTaintValue(
                TGlobal.getTaintType(), TGlobal.getValueToTaint());
        }
        u.getDefBoxes().get(0).setTaintValue(newTaintValue);

        if (getConfig().getLogSourcesAndSinks())
```

Best Practice

- What we do?
  - 4. Then we add code in EasyTaintWrapper, just insert a function to trace the process the TaintValue we add in Class AccessPath.

```
this.killList =
    SootMethodRepresentationParser.v().parseClassNames(killList, true);
    logger.info("Loaded wrapper entries for {} classes and {} exclusions.",
        classList.size(), excludeList.size());

    this.easyWrapperRule = new EasyTaintWrapperRule(classList, includeList);
```

Best Practice

- What we do?
  - 5. Then it's the key point, we do what the code really want to do on the value that we cares when it flows. Like transform the value via below mehtod.
    - String: substring, replaceAll, replace.
    - ZipEntry: getName
    - URI: getPath
  - Finally we return the changed value in the sink point to check whether it still dangerous or already be filtered to be safe

```
public boolean getPath(String path,
    AccessPath taintedPath, Stmt stmt) throws Exception {
    if (path == null) {
        return true;
    }
    try {
        URI uri = new URI(path);
        String newPath = uri.getPath();
        if (newPath.startsWith("/")) {
            newPath = newPath.substring(1);
        }
        TaintValue taintValue =
            TaintFactory.generateTaintValue(TGlobal.getTaintType(),
                newPath);
        taintedPath.setTaintValue(taintValue);
        return true;
    } catch (Exception e) {
        logger.info("Error when process URI " + path);
    }
    return true;
}
```

Best Practice – Function List

- Here are the functions we implementation for support value-sensitive taint analysis

```
# Packages to include in the analysis
^android.
^java.
^org.apache.http.
^org.joda.time.

<android.net.Uri: android.net.Uri getQueryParameter(java.lang.String)>
<android.net.Uri: android.net.Uri parse(java.lang.String)>
<android.net.Uri: java.lang.String getScheme()>
<android.net.Uri: java.lang.String getPath()>

<java.lang.StringBuilder: java.lang.StringBuilder append(boolean)>
<java.lang.StringBuilder: java.lang.StringBuilder append(char)>
<java.lang.StringBuilder: java.lang.StringBuilder append(char[])>
<java.lang.StringBuilder: java.lang.StringBuilder append(char[],int,int)>
<java.lang.StringBuilder: java.lang.StringBuilder append(double)>
<java.lang.StringBuilder: java.lang.StringBuilder append(float)>
<java.lang.StringBuilder: java.lang.StringBuilder append(int)>
<java.lang.StringBuilder: java.lang.StringBuilder append(long)>
<java.lang.StringBuilder: java.lang.StringBuilder append(java.lang.Object)>
<java.lang.StringBuilder: java.lang.StringBuilder append(java.lang.String)>
<java.lang.StringBuilder: java.lang.StringBuilder append(java.lang.CharSequence)>
<java.lang.StringBuilder: java.lang.StringBuilder append(java.lang.CharSequence,int,int)>
```

Information Classification: General

Agenda

- SAST Overview
- Vulnerabilities Study and Detection Patterns
- Meta Information Collection
- **Detection Strategy Implementation base on Static Taint-Analysis**
  - Common Checkers Implementation
  - Value-Sensitive Flow Analysis Implementation
  - **Specific Vulnerabilities Detection Implementation**
- Case Study

Information Classification: General

Constant Param Extraction

- Special optimization for special case: SINK point is clear, but where is source?
- A backward IFDS analysis on inverse ICFG
- Start from the function call statement that we are insterested in
- End at the assignment statement.
- Find where the params is defined & whether it is a constant.
- Can be used to find vulnerable function call
- Can be used to extract some hardcoded key / token / secret

Information Classification: General

Constant Param Extraction

```
//EASY MODE
byte[] keyRaw = {0x00, 0x00, ...};
byte[] ivRaw = {0x00, 0x00, ...};

SecretKeySpec key = new SecretKeySpec(keyRaw, "AES");
IvParameterSpec iv new IvParameterSpec(ivRaw);

Cipher cipher= Cipher.getInstance("AES/CBC/PKCS7Padding");
cipher.init(Cipher.ENCRYPT_MODE, key, iv);
```

Global analysis? Yes!

Constant Param Extraction

```
/*
Medium Mode
*/

void decrypt(...){
    byte[] keyRaw = {0x00, 0x00, ...};
    byte[] ivRaw = {0x00, 0x00, ...};

    decryptWithAESCBCMode(keyRaw,ivRaw,...);
}

void decryptWithAESCBCMode(byte[] keyRaw,byte[] ivRaw,...){
    SecretKeySpec keySpec = new SecretKeySpec(keyRaw, "AES");
    IvParameterSpec ivSepc new IvParameterSpec(ivRaw);

    Cipher cipher= Cipher.getInstance("AES/CBC/PKCS7Padding");
    cipher.init(Cipher.ENCRYPT_MODE, keySpec, ivSepc);
}
```

We need analysis on iCFG!!!

Constant Param Extraction

Track Forward: We have to hold a lot data fact(Constant Definitions)

Track Back: Less data fact(Track Poir

	Constant Param Extraction on inverse iCFG
1	ForEach edge in InverseICFG
2	If edge.isAssignment lhs = rhs:
3	If lhs.alias in TrackPoint and rhs is constant:
4	add rhs to FoundList, kill lhs in TrackPoint
5	Else if lhs.alias in TrackPoint and rhs in value:
6	gen rhs and kill lhs in TrackPoint
7	If edge.isCallStmt:
8	If local_var.alias in TrackPoint:
8	Propergate local variables back to arguments
10	If edge.isReturnStmt:
11	If ret_value.alias in TrackPoint:
12	Propergate ret value to function local variable list

Information Classification: General

Constant Param Extraction : Demo

```
/*
Medium Mode
*/

void decrypt(...){
    byte[] keyRaw = {0x00, 0x00, ...};
    byte[] ivRaw = {0x00, 0x00, ...};

    decryptWithAESCBCMode(keyRaw,ivRaw,...);
}

void decryptWithAESCBCMode(byte[] keyRaw,byte[] ivRaw,...){
    SecretKeySpec keySpec = new SecretKeySpec(keyRaw, "AES");
    IvParameterSpec ivSepc new IvParameterSpec(ivRaw);

    Cipher cipher= Cipher.getInstance("AES/CBC/PKCS7Padding");
    cipher.init(Cipher.ENCRYPT_MODE, keySpec, ivSepc);
}
```

Assume that we only care about param **keySpec** for **javax.crypto.Cipher.init**

Constant Param Extraction : Demo

```
/*
Medium Mode
*/

void decrypt(...){
    byte[] keyRaw = {0x00, 0x00, ...};
    byte[] ivRaw = {0x00, 0x00, ...};

    decryptWithAESCBCMode(keyRaw,ivRaw,...);
}

void decryptWithAESCBCMode(byte[] keyRaw,byte[] ivRaw,...){
    SecretKeySpec keySpec = new SecretKeySpec(keyRaw, "AES");
    IvParameterSpec ivSepc new IvParameterSpec(ivRaw);

    Cipher cipher= Cipher.getInstance("AES/CBC/PKCS7Padding");
    cipher.init(Cipher.ENCRYPT_MODE, keySpec, ivSepc);
}
```

Call Edge: Gen (keySpec)

Constant Param Extraction : Demo

```
/*
Medium Mode
*/

void decrypt(...){
    byte[] keyRaw = {0x00, 0x00, ...};
    byte[] ivRaw = {0x00, 0x00, ...};

    decryptWithAESCBCMode(keyRaw,ivRaw,...);
}

void decryptWithAESCBCMode(byte[] keyRaw,byte[] ivRaw,...){
    SecretKeySpec keySpec = new SecretKeySpec(keyRaw, "AES");
    IvParameterSpec ivSepc new IvParameterSpec(ivRaw);

    Cipher cipher= Cipher.getInstance("AES/CBC/PKCS7Padding");
    cipher.init(Cipher.ENCRYPT_MODE, keySpec, ivSepc);
}
```

Assign Stmt + Call Stmt: Do_Nothing()
Call Edge: Gen (keySpec)

Constant Param Extraction : Demo

```
/*
Medium Mode
*/

void decrypt(...){
    byte[] keyRaw = {0x00, 0x00, ...};
    byte[] ivRaw = {0x00, 0x00, ...};

    decryptWithAESCBCMode(keyRaw,ivRaw,...);
}

void decryptWithAESCBCMode(byte[] keyRaw,byte[] ivRaw,...){
    SecretKeySpec keySpec = new SecretKeySpec(keyRaw, "AES");
    IvParameterSpec ivSepc new IvParameterSpec(ivRaw);

    Cipher cipher= Cipher.getInstance("AES/CBC/PKCS7Padding");
    cipher.init(Cipher.ENCRYPT_MODE, keySpec, ivSepc);
}
```

Assign Stmt + Call Stmt: Do_Nothing()

Assign Stmt + Call Stmt: Do_Nothing()
Call Edge: Gen(keySpec)

Constant Param Extraction : Demo

```
/*
Medium Mode
*/

void decrypt(...){
    byte[] keyRaw = {0x00, 0x00, ...};
    byte[] ivRaw = {0x00, 0x00, ...};

    decryptWithAESCBCMode(keyRaw,ivRaw,...);
}

void decryptWithAESCBCMode(byte[] keyRaw,byte[] ivRaw,...){
    SecretKeySpec keySpec = new SecretKeySpec(keyRaw, "AES");
    IvParameterSpec ivSepc new IvParameterSpec(ivRaw);

    Cipher cipher= Cipher.getInstance("AES/CBC/PKCS7Padding");
    cipher.init(Cipher.ENCRYPT_MODE, keySpec, ivSepc);
}
```

Assign Stmt + Call Stmt: Gen (keyRaw), Kill (keySpec)
Assign Stmt + Call Stmt: Do_Nothing()

Assign Stmt + Call Stmt: Do_Nothing()
Call Edge: Gen(keySpec)

Constant Param Extraction : Demo

```
/*
Medium Mode
*/

void decrypt(...){
    byte[] keyRaw = {0x00, 0x00, ...};
    byte[] ivRaw = {0x00, 0x00, ...};

    decryptWithAESCBCMode(keyRaw,ivRaw,...);
}

void decryptWithAESCBCMode(byte[] keyRaw,byte[] ivRaw,...){
    SecretKeySpec keySpec = new SecretKeySpec(keyRaw, "AES");
    IvParameterSpec ivSepc new IvParameterSpec(ivRaw);

    Cipher cipher= Cipher.getInstance("AES/CBC/PKCS7Padding");
    cipher.init(Cipher.ENCRYPT_MODE, keySpec, ivSepc);
}
```

Call Stmt: Gen(keyRaw_local), Kill(keyRaw_param)

Assign Stmt + Call Stmt: Gen (keyRaw), Kill (keySpec)
Assign Stmt + Call Stmt: Do_Nothing()

Assign Stmt + Call Stmt: Do_Nothing()
Call Edge: Gen(keySpec)

Constant Param Extraction : Demo

```
/*
Medium Mode
*/

void decrypt(...){
    byte[] keyRaw = {0x00, 0x00, ...};
    byte[] ivRaw = {0x00, 0x00, ...};

    decryptWithAESCBCMode(keyRaw,ivRaw,...);
}

void decryptWithAESCBCMode(byte[] keyRaw,byte[] ivRaw,...){
    SecretKeySpec keySpec = new SecretKeySpec(keyRaw, "AES");
    IvParameterSpec ivSepc new IvParameterSpec(ivRaw);

    Cipher cipher= Cipher.getInstance("AES/CBC/PKCS7Padding");
    cipher.init(Cipher.ENCRYPT_MODE, keySpec, ivSepc);
}
```

Assign Stmt: Do_Nothing()

Call Stmt: Gen(keyRaw_local), Kill(keyRaw_param)

Assign Stmt + Call Stmt: Gen (keyRaw), Kill (keySpec)
Assign Stmt + Call Stmt: Do_Nothing()

Assign Stmt + Call Stmt: Do_Nothing()
Call Edge: Gen(keySpec)

Constant Param Extraction : Demo

```
/*
Medium Mode
*/

void decrypt(...){
    byte[] keyRaw = {0x00, 0x00, ...};
    byte[] ivRaw = {0x00, 0x00, ...};

    decryptWithAESCBCMode(keyRaw,ivRaw,...);
}

void decryptWithAESCBCMode(byte[] keyRaw,byte[] ivRaw,...){
    SecretKeySpec keySpec = new SecretKeySpec(keyRaw, "AES");
    IvParameterSpec ivSepc new IvParameterSpec(ivRaw);

    Cipher cipher= Cipher.getInstance("AES/CBC/PKCS7Padding");
    cipher.init(Cipher.ENCRYPT_MODE, keySpec, ivSepc);
}
```

Assign Stmt: Gen(keyRaw) : We found it!
Assign Stmt: Do_Nothing()

Call Stmt: Gen(keyRaw_local), Kill(keyRaw_param)

Assign Stmt + Call Stmt: Gen (keyRaw), Kill (keySpec)
Assign Stmt + Call Stmt: Do_Nothing()

Assign Stmt + Call Stmt: Do_Nothing()
Call Edge: Gen(keySpec)

Constant Param Extraction : Practice 1

```
URL url = new URL(inputedURL);

if (url.getProtocol().toLowerCase().equals("https")) {
    trustAllHosts();
    HttpsURLConnection https = (HttpsURLConnection) url.openConnection();
    https.setHostnameVerifier(DO_NOT_VERIFY);
    urlConnection = https;
} else {
    urlConnection = (HttpURLConnection) url.openConnection();
}
```

Insecure HTTPS options, from pivaa project

Constant Param Extraction : Practice 2

```java
/**
 * Encrypt DATA
 * @param value
 * @return
 */
public static String encryptAES_ECB_PKCS5Padding(String value) {
    try {
        byte[] key = {
                1, 2, 3, 4, 5, 6, 7, 8, 8, 7, 6, 5, 4, 3, 2, 1
        };
        SecretKeySpec skeySpec = new SecretKeySpec(key, "AES");

        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec);

        byte[] encrypted = cipher.doFinal(value.getBytes());

        Base64 b64 = new Base64();
        System.out.println("encrypted string: " + new String(b64.encodeBase64(encrypted)));

        return new String(b64.encodeBase64(encrypted));
    } catch (Exception ex) {
        ex.printStackTrace();
    }
```

Hardcoded AES keys that can not be identified by Regex, from pivaa project

Conclusion

- We introduce our common checkers workflow, and details our value-sensitive and constant propagate algorithm base on static taint analysis engine. Base on these engine, we can cover most vulnerabilities occurred in android APKs.

Information Classification: General

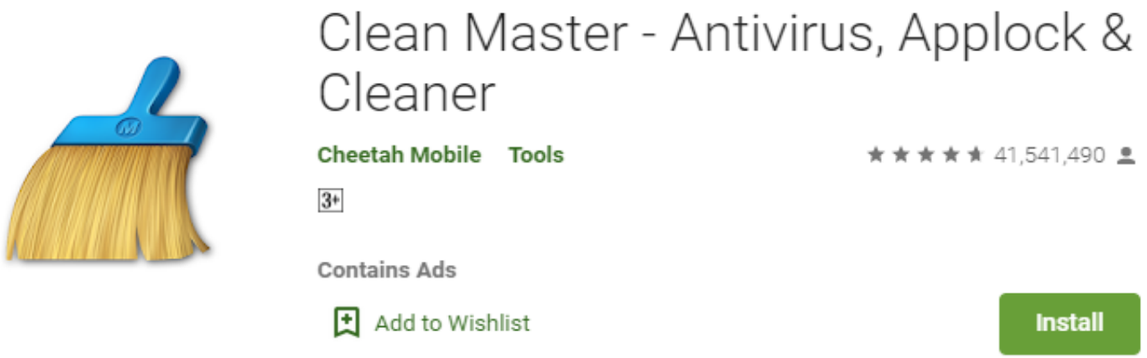
Agenda

- SAST Overview
- Vulnerabilities Study and Detection Patterns
- Meta Information Collection
- Detection Strategy Implementation base on Static Taint-Analysis
  - Common Checkers Implementation
  - Value-Sensitive Flow Analysis Implementation
  - Specific Vulnerabilities Detection Implementation
- **Case Study**

Information Classification: General

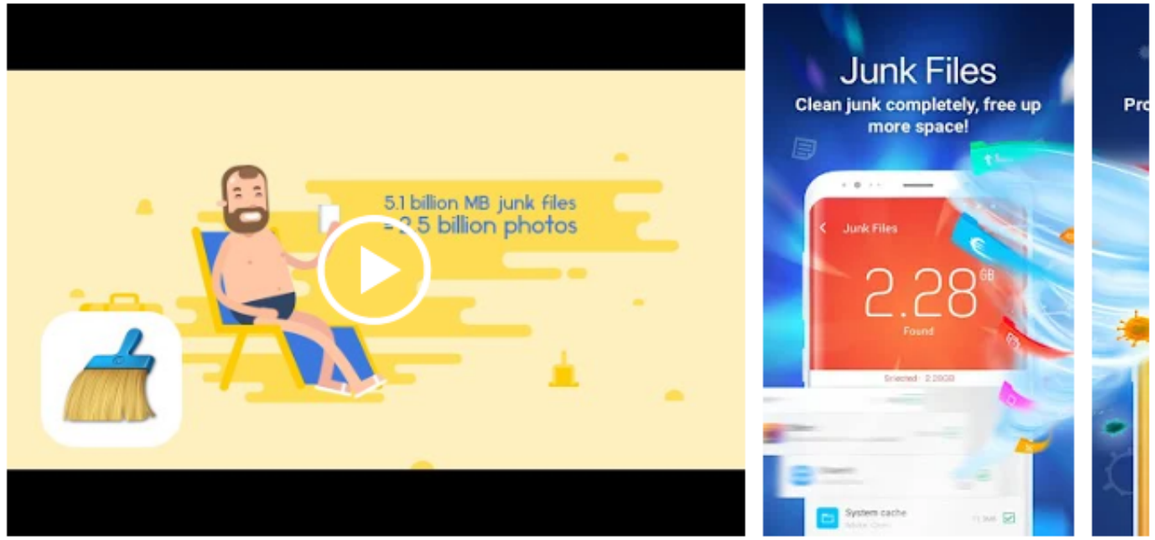
Case Study (1/4) - ContentProvider path traversal vulnerability

- google play download: 1,000,000,000+
- affected version: 7.3.1



ADDITIONAL INFORMATION

Updated	Size	Installs
October 22, 2019	24M	1,000,000,000+
Current Version	**Requires Android**	**Content Rating**
7.3.1	Varies with device	Rated for 3+ Learn More
Permissions	**Report**	**Offered By**
View details	Flag as inappropriate	Cheetah Mobile



Case Study (1/4) - Details

- “com.cleanmaster.notification.NotificationFileProvider”:exported = ”true”

```
</provider>
<provider android:authorities="com.cleanmaster.provider.notification" android:exported="true" android:name="com.cleanmaster.notification.NotificationFileProvider" />
<activity android:configChanges="keyboardHidden|orientation|screenSize" android:launchMode="singleTask" android:name="com.cleanmaster.vip.VipActivity"
```

- openAssetFile function
  was rewritten without
  any "../" filter

Case Study (1/4) - Details

- “com.cleanmaster.notificati

```
</provider>
<provider android:authorities="com.cleanmaster.provider.notif
<activity android:configChanges="keyboardHidden|orientation|s
```

- openAssetFile function was rewritten without any "../" filter

```
private static ParcelFileDescriptor n(Uri arg2) {
    File v0 = new File(arg2.getPath());
    ParcelFileDescriptor v1 = null;
    if(!v0.exists()) {
        return v1;
    }

    int v2 = 268435456;
    try {
        return ParcelFileDescriptor.open(v0, v2);
    }
    catch(FileNotFoundException v2_1) {
        v2_1.printStackTrace();
        return v1;
    }
}

public boolean onCreate() {
    return 0;
}

public AssetFileDescriptor openAssetFile(Uri arg7, String arg8) throws FileNotFoundException {
    ParcelFileDescriptor v1 = NotificationFileProvider.n(arg7);
    if(v1 == null) {
        return null;
    }

    return new AssetFileDescriptor(v1, 0, -1);
}

public AssetFileDescriptor openAssetFile(Uri arg7, String arg8, CancellationSignal arg9) throws FileNotFoundException {
    ParcelFileDescriptor v1 = NotificationFileProvider.n(arg7);
    if(v1 == null) {
        return null;
    }

    return new AssetFileDescriptor(v1, 0, -1);
}
```

Case Study (1/4) - Details

- “com.cleanmaster.notificati

```
</provider>
<provider android:authorities="com.cleanmaster.provider.notif
<activity android:configChanges="keyboardHidden|orientation|s
```

- openAssetFile function was rewritten without any "../" filter

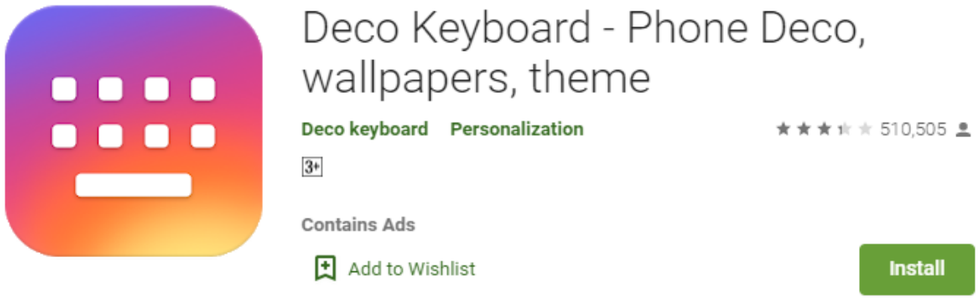
- POC:

Content://com.cleanmaster.p

```
private static ParcelFileDescriptor n(Uri arg2) {
    File v0 = new File(arg2.getPath());
    ParcelFileDescriptor v1 = null;
    if(!v0.exists()) {
        return v1;
    }

    int v2 = 268435456;
    try {
        return ParcelFileDescriptor.open(v0, v2);
    }
    catch(FileNotFoundException v2_1) {
        v2_1.printStackTrace();
        return v1;
    }
}

public boolean onCreate() {
    return 0;
}

public AssetFileDescriptor openAssetFile(Uri arg7, String arg8) throws FileNotFoundException {
    ParcelFileDescriptor v1 = NotificationFileProvider.n(arg7);
    if(v1 == null) {
        return null;
    }

    return new AssetFileDescriptor(v1, 0, -1);
}

public AssetFileDescriptor openAssetFile(Uri arg7, String arg8, CancellationSignal arg9) throws FileNotFoundException {
    ParcelFileDescriptor v1 = NotificationFileProvider.n(arg7);
    if(v1 == null) {
        return null;
    }

    return new AssetFileDescriptor(v1, 0, -1);
}
```

Case Study (2/4) - ContentProvider path traversal vulnerability

- google play download: 10,000,000+
- affected version: 4.2.2



Updated	Size	Installs
June 26, 2019	9.8M	10,000,000+
Current Version	**Requires Android**	**Content Rating**
4.2.2	4.1 and up	Rated for 3+ Learn More
Interactive Elements	**Permissions**	**Report**
Users Interact, Digital	View details	Flag as inappropriate

Case Study (2/4) - Details

- com.designkeyboard.keyboard.keyboard.provider.InputContentProvider

```
<provider android:authorities="com.brainpub.phonedecor.provider.keyboard" android:exported="true" android:grantUriPermissions="true" android:name="com.designkeyboard.keyboard.keyboard.provider.InputContentProvi
    <meta-data android:name="android.support.FILE_PROVIDER_PATHS" android:resource="@xml/filkbd_provider" />
    <grant-uri-permission android:pathPattern=".*" />
</provider>
```

```java
@Nullable public ParcelFileDescriptor openFile(@NonNull Uri arg2, @NonNull String arg3) throws FileNotFoundException {
    return this.openFileEx(arg2, arg3);
}
```

---a)

```java
@Nullable private ParcelFileDescriptor openFileEx(@NonNull Uri arg5, @NonNull String arg6) throws FileNotFoundException {
    Log.e("InputContentProvider", "openFile :" + arg5.toString());
    if(!"r".equalsIgnoreCase(arg6)) {
        throw new SecurityException("Read only Content provider");
    }

    File v0 = this.getFileFromUri(arg5);
    if(v0 != null && (v0.exists())) {
        Log.e("InputContentProvider", "File Path :" + v0.getAbsolutePath());
        return ParcelFileDescriptor.open(v0, 268435456);
    }

    throw new FileNotFoundException("File not Found");
}
```

---b)

```java
@Nullable private File getFileFromUri(Uri arg5) {
    File v0 = null;
    try {
        String v1_1 = InputContentProvider.regurateUri(arg5).getQueryParameter("url");
        if(TextUtils.isEmpty(((CharSequence)v1_1))) {
            return v0;
        }

        Uri v2 = Uri.parse(v1_1);
        if(!"file".equals(v2.getScheme())) {
            return v0;
        }

        v0 = new File(v2.getPath());
    }
    catch(Exception v1) {
        v1.printStackTrace();
    }

    return v0;
}
```

---c)

---f)

```java
private static Uri regurateUri(Uri arg2) throws Exception {
    while(true) {
        String v0 = arg2.getQueryParameter("url");
        if(v0 != null && (v0.startsWith("content"))) {
            arg2 = Uri.parse(v0);
            continue;
        }

        return arg2;
    }

    return arg2;
}
```

---d)

---e)

- PoC:
  - Content://com.brainpub.phonedecor.provider.keyboard/?url=file://../../../../system/etc/hosts

Information Classification: General

Case Study (3/4) - Zipperdown Vulnerability

- google play download: 1,000,000+
- affected version is 2.0.128.1

ADDITIONAL INFORMATION

Updated	**Size**	**Installs**
September 25, 2019	23M	1,000,000+
Current Version	**Requires Android**	**Content Rating**
2.0.128.1	4.2 and up	Rated for 3+ Learn More
Permissions	**Report**	**Offered By**
View details	Flag as inappropriate	Appz Inventors

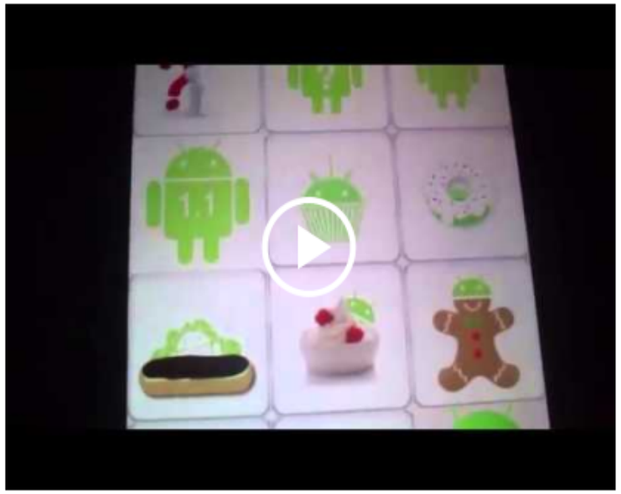
Update for Android (info) - Software Update Info

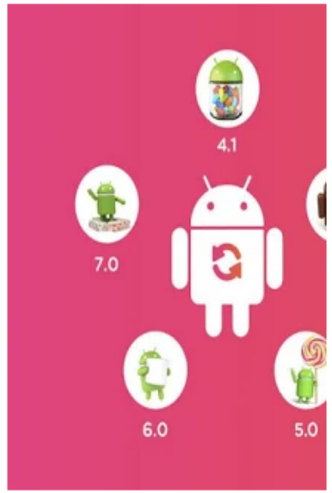
Appz Inventors Books & Reference 11,212

3+

Contains Ads

Add to Wishlist Install





Case Study (3/4) - Details

- Function within the class of com.startapp.android.publish.adsCommon.Utils.f
- two parameters: arg8 is the context, arg9 is the zip file path
- Using ZipInputStream open the zip file in place a)
- Got one file entry in place b)
- Using getName function to get file name in place c)

```
protected static void a(Context arg8, String arg9) {
    ZipEntry v1_5;
    ZipEntry v0_2;
    ZipInputStream v1_3;
    FileOutputStream v1_4;
    Throwable v0_1;
    ZipInputStream v2;
    FileOutputStream v0 = null;
    byte[] v3 = new byte[1024];
    try {
        v2 = new ZipInputStream(new FileInputStream(arg9));
    }
    catch(Throwable v1) {
        v2 = ((ZipInputStream)v0);
        FileOutputStream v7 = v0;
        v0_1 = v1;
        v1_4 = v7;
        goto label_62;
    }
    catch(IOException v1_1) {
        v1_4 = v0;
        v2 = ((ZipInputStream)v0);
        goto label_52;
    }
    catch(FileNotFoundException v1_2) {
        v1_3 = ((ZipInputStream)v0);
        goto label_34;
    }

    try {
        v1_4 = v0;
        v0_2 = v2.getNextEntry();
        goto label_11;
    }
    catch(FileNotFoundException v1_2) {
    }
    catch(Throwable v1) {
        Throwable v7_1 = v1;
        v1_5 = v0_2;
        v0_1 = v7_1;
        goto label_62;
        while(true) {
        label_11:
            if(v0_2 == null) {
                goto label_44;
            }

            try {
                v0 = new FileOutputStream(arg8.getFilesDir().getPath() + "/" + v0_2.getName());
            }
            catch(Throwable v0_1) {
                goto label_62;
            }
```

a)

b)

c)

Case Study (3/4) - Details

- POC:
  - If v0_2.getName() is the following string, it will result in RCE:

    ../../../../../../data/data/<other package's dir>/<executable file>

Case Study (3/4) - Details

- POC:
  - If v0_2.getName() is the following string, it will result in RCE:

    ../../../../../../data/data/<other package's dir>/<executable file>

Case Study (4/4) – Intent URL Scheme Class

- google play download: 5,000,00(
- affected version:7.1.25

SATCH VIEWER

KDDI株式会社 Entertainment ★★★☆☆ 1,167

3+

Contains Ads

Add to Wishlist

ADDITIONAL INFORMATION

Updated	Size	Installs
October 9, 2019	153M	5,000,000+
Current Version	**Requires Android**	**Content Rating**
7.1.25	4.1 and up	Rated for 3+ Learn More
Permissions	**Report**	**Offered By**
View details	Flag as inappropriate	KDDI株式会社

Case Study (4/4) - Details

- POC

Case Study (4/4) - Details

- POC

```
this.b = this.getIntent().getSerializableExtra("EXTRA_BARCODE_DATA");
View v3 = this.findViewById(id.text_barcode_read_data);
((TextView)v3).setText(this.b.originText);
```

——————a)

——————b)

```
v3 = {BarcodeData@7162} "barcodeType=1, dataType=2, originText=file://../../../../../system/etc/hosts, analyzedText=null, pickupInfo=null"
```

——————c)

```
((TextView)v3).setOnClickListener(new View$OnClickListener() {
    public void onClick(View arg3) {
        if(BarcodeResultActivity.a(this.a).barcodeType == 0) {
            BarcodeResultActivity.a(this.a, "https://mono-talk.com/goods/page?code=" + BarcodeResultActivity.a(this.a).pickupInfo + "&app=satch");
        }
        else {
            Intent v3_1 = new Intent(this.a.getApplicationContext(), BarcodeOriginTextActivity.class);
            v3_1.putExtra("EXTRA_ORIGIN_TEXT", BarcodeResultActivity.a(this.a).originText);
            this.a.startActivity(v3_1);
        }
    }
});
```

——————d)

```
protected void onCreate(Bundle arg9) {
    super.onCreate(arg9);
    this.setContentView(layout.activity_barcode_origin_text);
    String v4 = a.a(this.getIntent().getStringExtra("EXTRA_ORIGIN_TEXT")).replaceAll("((https?|ftp|gopher|telnet|file):((//)|(\\\\))+[\\w\\d:#@%/;$()~_?\\+-=\\\\\\.&]*)", "<a href=\"$1\">$1</a>");
    View v2 = this.findViewById(id.web);
    ((WebView)v2).getSettings().setJavaScriptEnabled(false);
    ((WebView)v2).loadDataWithBaseURL("", v4, "text/html", "UTF-8", "");
    this.findViewById(id.adView).a(new com.google.android.gms.ads.c$a().a());
}
```

——————f)

```
v4 = "<a href="file://../../../../../system/etc/hosts">file://../../../../../system/etc/hosts</a>"
```

——————g)

Conclusions

- In this section, we introduce some representative cases found by SAST tools. These apps either have huge installations or developed by famous companies. All these security issues had been informed the vendors through ZDI or email.

[internal case]Kohl's ContentProvider Path Traversal Vulnerability — Nov. 12, 2019, 7:44 p....

[internal case]Clean Master ContentProvider Path Traversal Vulnerab

[internal case]ESC Pocket Guidelines ContentProvider Path Traversal

[internal case]Deco Keyboard ContentProvider Path Traversal Vulnera

Vulnerabilities found in KDDI SATCH VIEWER application

LW **Lilang Wu (RD-CN)** Thursday, Octob

To: Shiga Tomoyuki (RD-JP); Kobayashi Keiko (TS-JP); Yoshino Ryohei (TS-JP); +4 more

SATCH VIEWER Two... 364.7 KB

Download All Preview All

Hi Shiga,

We have found two bugs in KDDI VIEWER application, the details can be get from attachment file. Please help us

Questions?

Information Classification: General