
black hat
ASIA 2018
MARCH 20-23, 2018
MARINA BAY SANDS / SINGAPORE
VSPMiner: Detecting Security Hazards
in SEAndroid Vendor Customizations via Large-Scale Supervised Machine Learning
Xiangyu Liu, Yi Zhang, Yang Song
Alibaba Security
#BHASIA / @BlackHatEvents

Whoami

- Xiangyu Liu
- Security Engineer @Alibaba
- CUHK PhD (2016)
- Academic: IEEE S&P, ACM CCS
- Industry: DEF CON
- Interests: Intrusion Detection, Mobile security

- Co-author: Yi Zhang, Yang Song @Alibaba

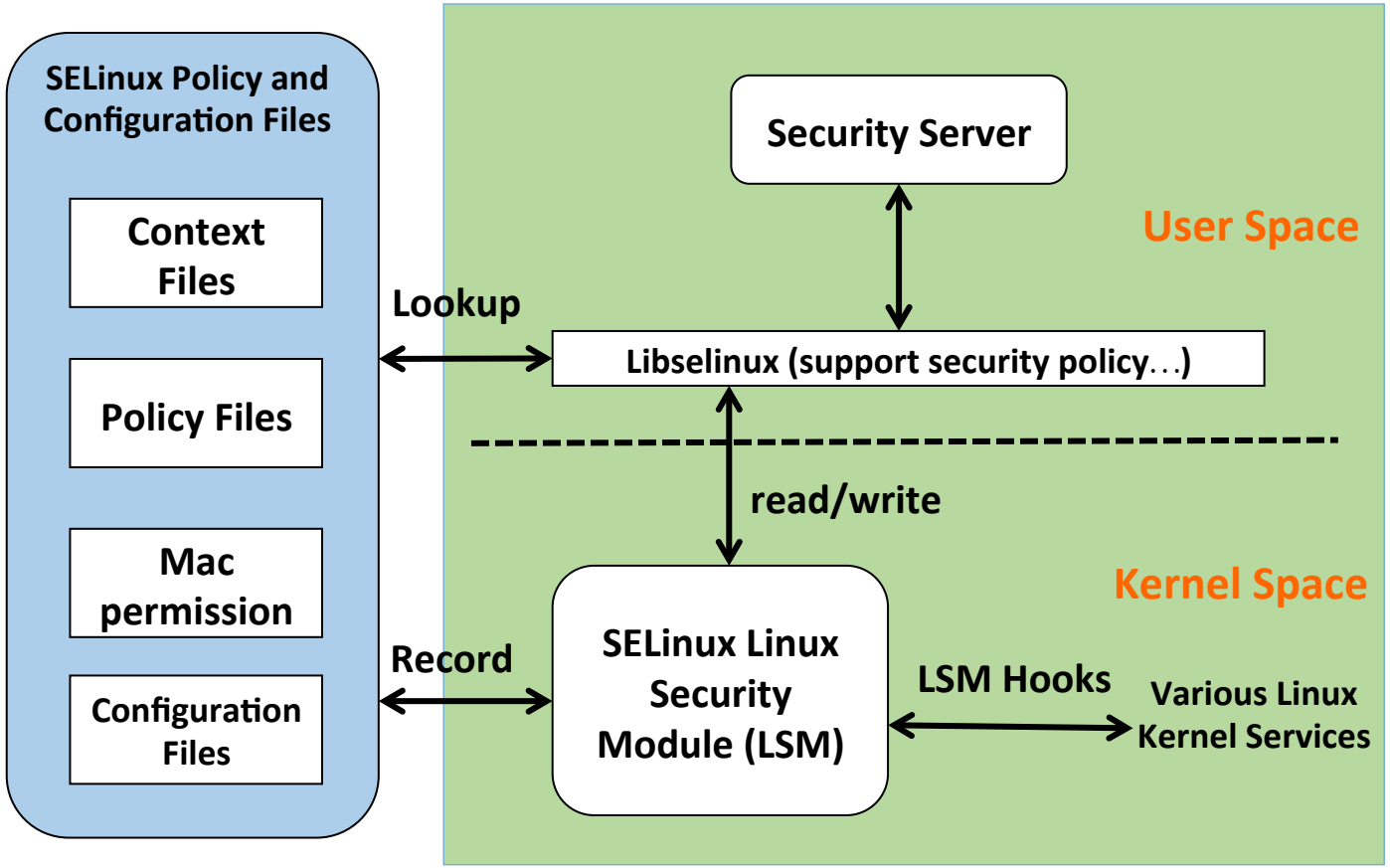
Agenda

- Background
- VSPMiner
- Evaluation
- Summary

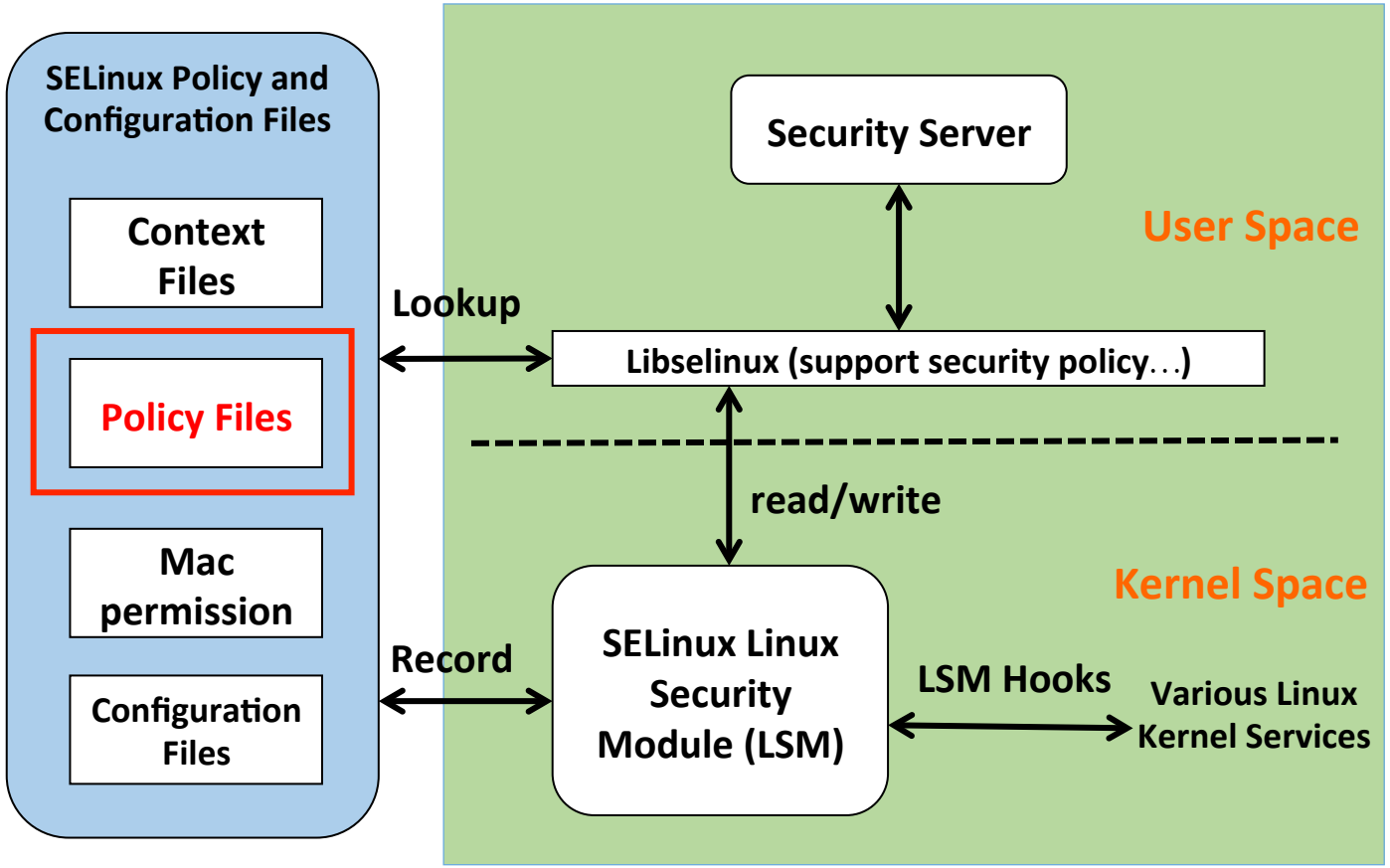
Background-SEAndroid

- Android uses SELinux to enforce mandatory access control (MAC) over all processes.
  - After Android 4.4
- Privilege escalation becomes much more difficult

Background-SEAndroid Framework



Background-SEAndroid Framework



Background-SEAndroid Policy

- The effectiveness of SEAndroid depends on the employed policies.

- *allow/neverallow subject object:object_class permission*
  - *sbj, obj, obj_class, perm (for short)*
  - Allow rules define benign operations
    - E.g., allow appdomain app_data_file:file {read write execute}
  - Neverallow rules define privilege escalation (compile time)
    - E.g., neverallow untrusted_app init:file {read}

- Security labels <=> Concrete subjects/objects
  - system_file <=> /system(/.*)
  - system_data_file <=> /data(/.*)

Vendors don’t know how to write policies

@pof “Defeat SEAndroid” at Defcon 2013

Background-Refine Policy

- Using audit logs
  - 6-tuple access patterns
  - <concrete_sbj, sbj, concrete_obj, obj, obj_class, perm>
- Policy engineers parse the logs to refine policy
- Log access events not matched with allow rules

Background-Challenges

- Millions of audit logs
- Expert experience
  - Allow benign accesses
  - Prevent malicious accesses
- Unknown new malicious access patterns

Background-Vendor Customizations

- Vendor customizations
  - Add apps, device drivers and other new features
- Small time window
  - Manufactures always have only about 6 months (or less) to customize the official version

ZTE LeEco Xiaomi oppo htc SAMSUNG HUAWEI vivo

Background-Related Work

- EASEAndroid @Usenix Security'16
  - Samsung devices
  - Using audit logs
  - Known access patterns, i.e., 17 malicious access patterns
  - Semi-supervised machine learning

- Learning unknown based on semantic correlations
  - Nearest-Neighbor (NN) Classifier
  - Pattern-to-Rule Distance Measurer
  - Co-Occurrence Learner

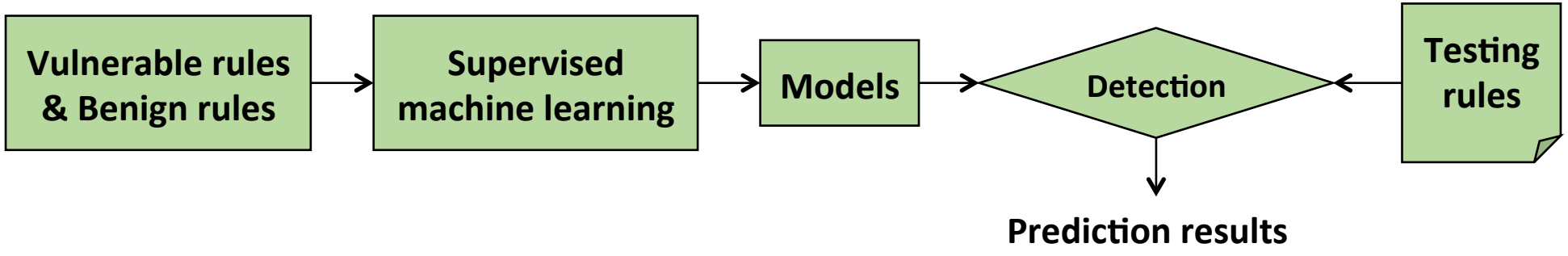
VSPMiner

- **V**ulnerable **S**EAndroid **P**olicy **Miner**

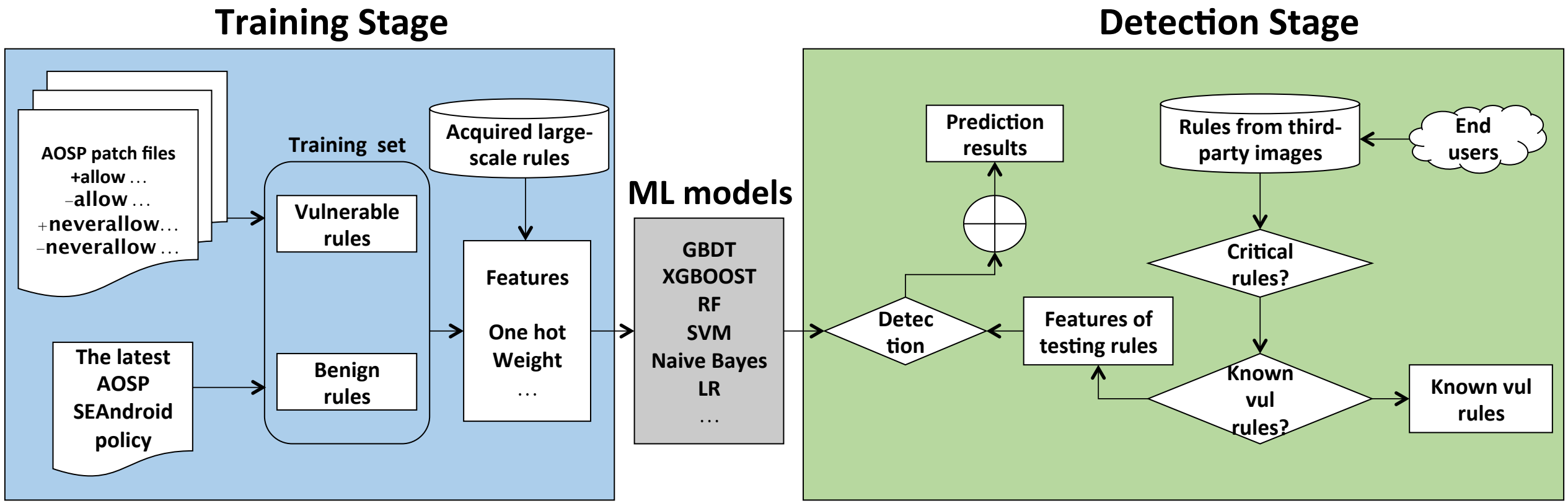
- Features
  - All 3rd party vendors
  - Thousands of “vulnerable” rules acquired from SEAndroid patch files
  - Features are extracted from rules
  - Supervised machine learning

VSPMiner

- Key idea
  - Assume the latest AOSP policy is trusted
  - The rules deleted from SEAndroid Patch files are defined as vulnerable
  - Focus on critical rules

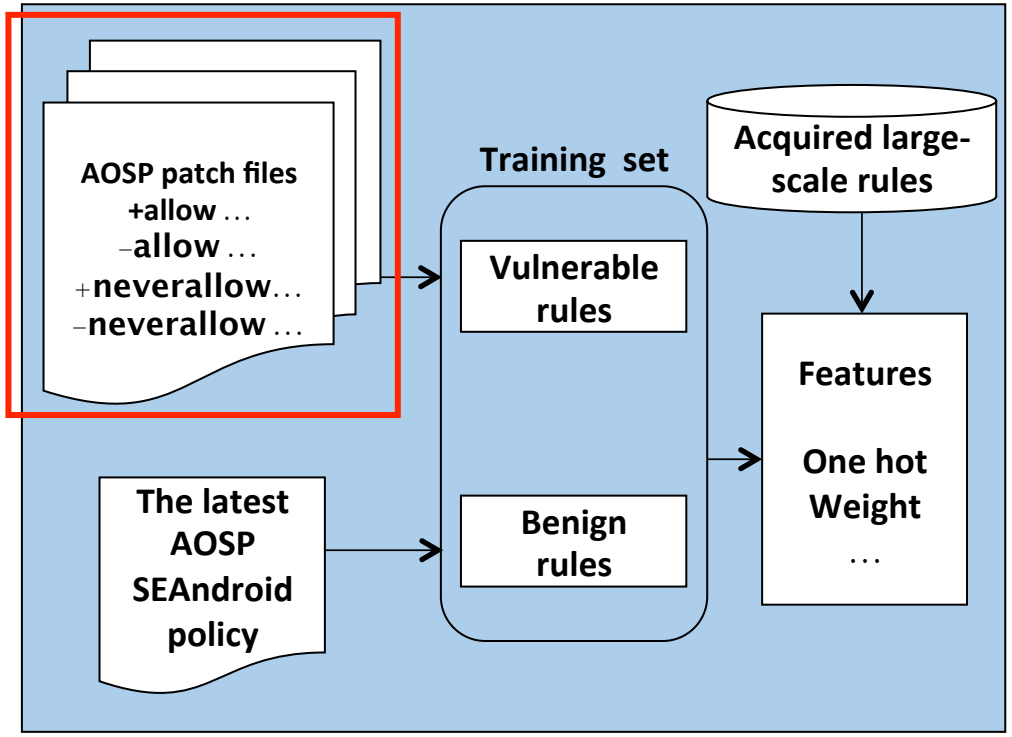


VSPMiner Architecture



VSPMiner Architecture

Training Stage

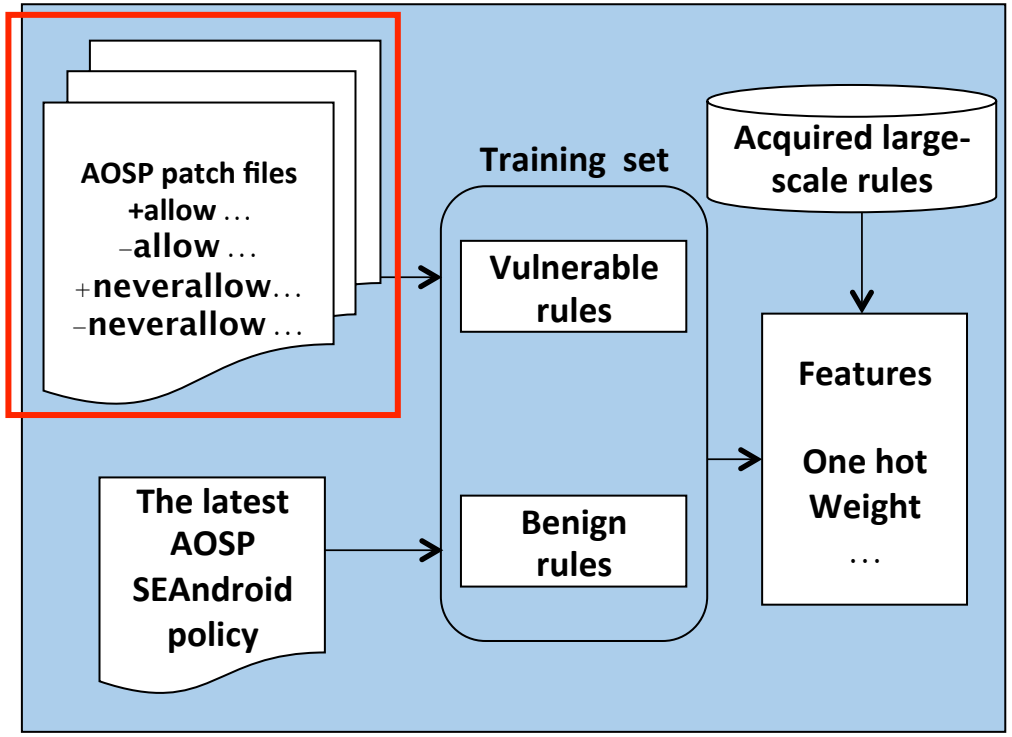


Analyzing SEAndroid Patch Files

- SEAndroid Evolution
  - https://android.googlesource.com/platform/system/sepolicy
- Obtain commit IDs from the log file
  - 9584 commit ids.
  - Aug 1 13:27:32 2017
- *diff* each commit ID with its parent
  - +allow/-allow rules
  - +neverallow/-neverallow rules.

VSPMiner Architecture

Training Stage

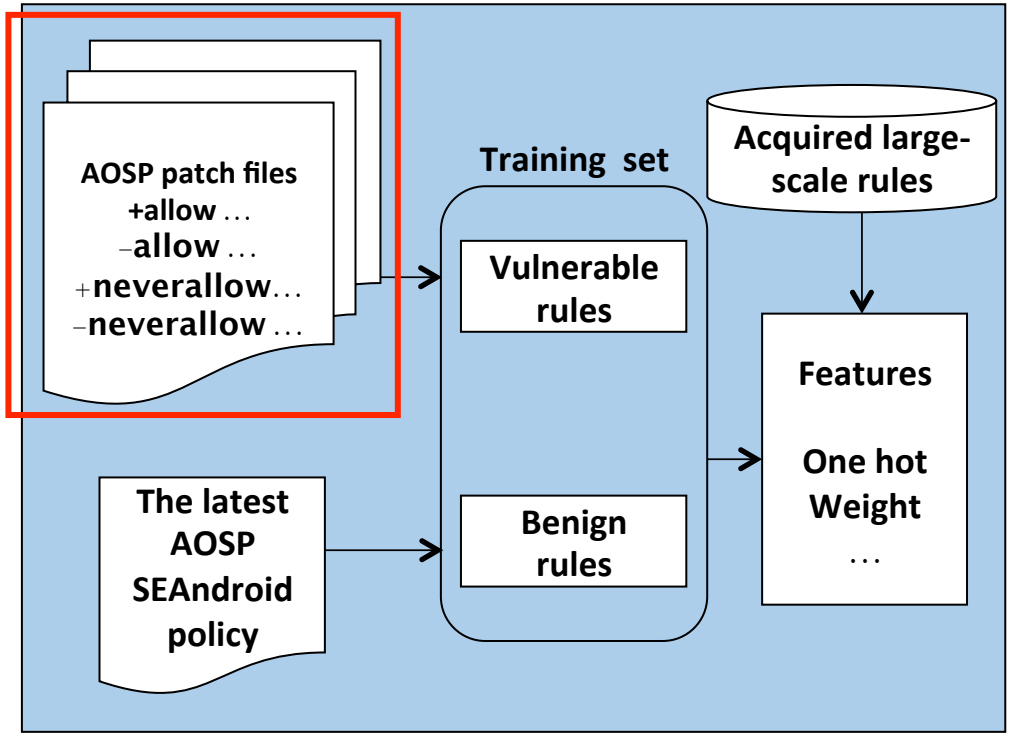


Differential Analysis

- Specific commit id + allow rules
- Split
  - {domain -init} → domain, –init
  - ~{relabelto getattr} → ~relabelto, ~getattr
- Deleted allow rules S^{id}_{-ar}
- Added allow rules S^{id}_{+ar}
- Differential analysis $P^{id}_{ar} = S^{id}_{-ar} - S^{id}_{+ar}$
  - E.g., init kernel:security load_policy

VSPMiner Architecture

Training Stage



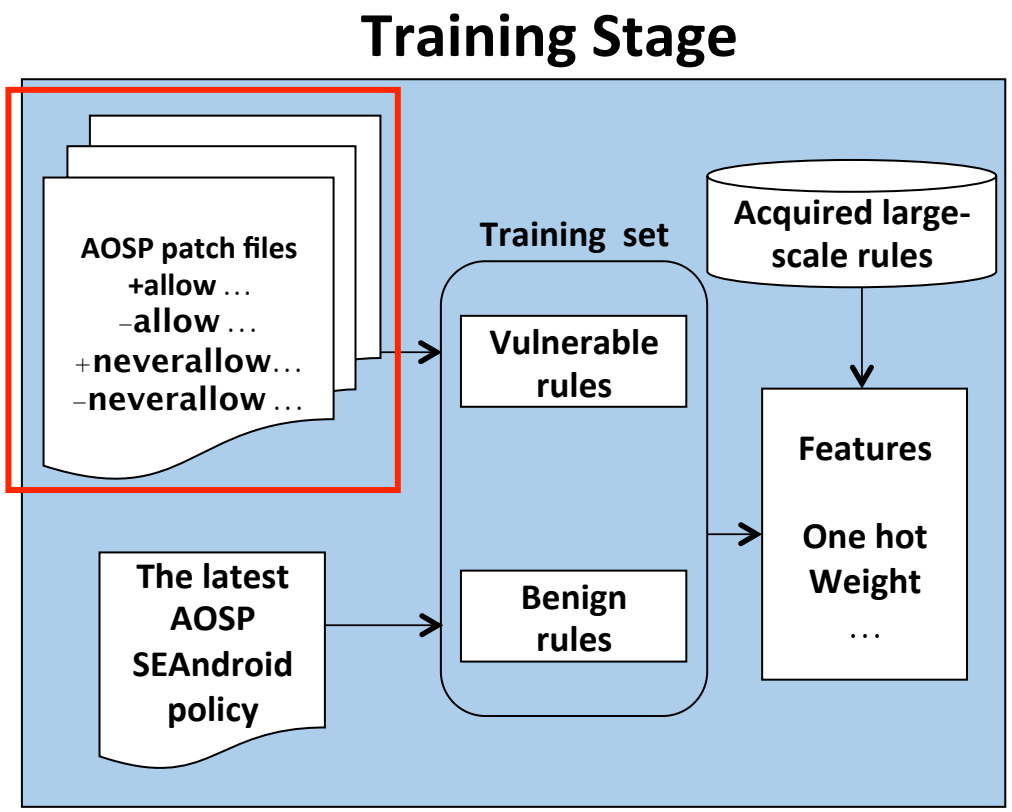
Differential Analysis

- *neverallow* rules
- Deleted *neverallow* rules S^{id}_{-nar}
- Added *neverallow* rules S^{id}_{+nar}
- Differential analysis

$$P^{id}_{nar} = S^{id}_{+nar} - S^{id}_{-nar}$$

  - E.g., untrusted_app init:file read

VSPMiner Architecture



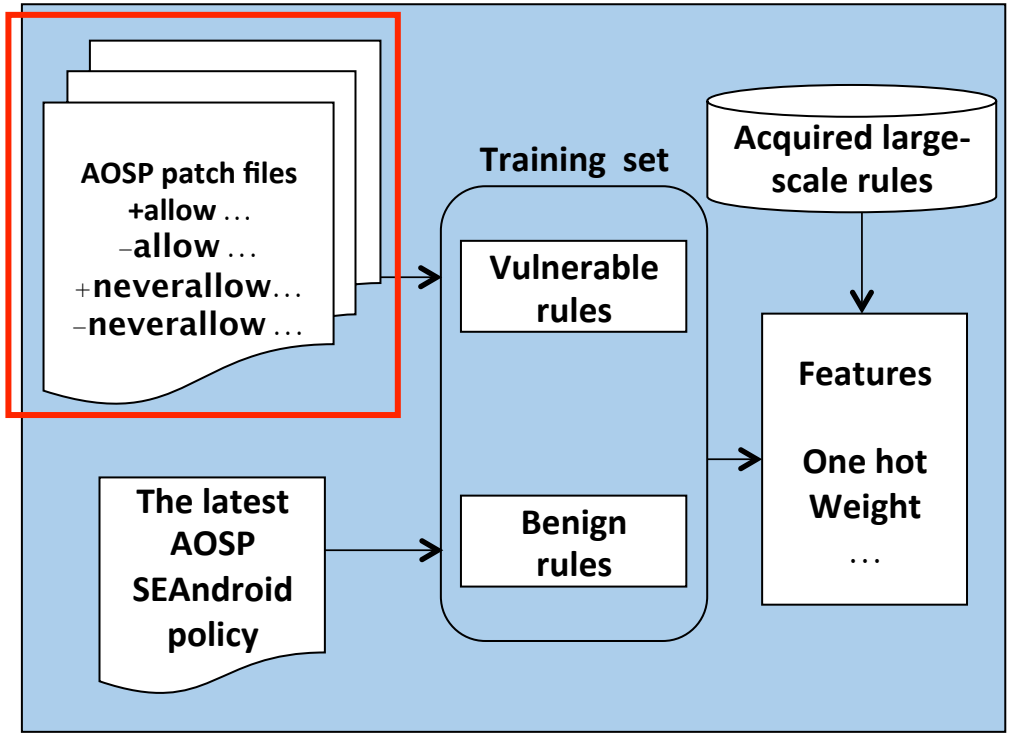
Combining Differential Results

- Combine the results of all commit ids

$$P_r^{'} = \bigcup_{id \in ID} P_{ar}^{id} \cup \bigcup_{id \in ID} P_{nar}^{id}$$

VSPMiner Architecture

Training Stage

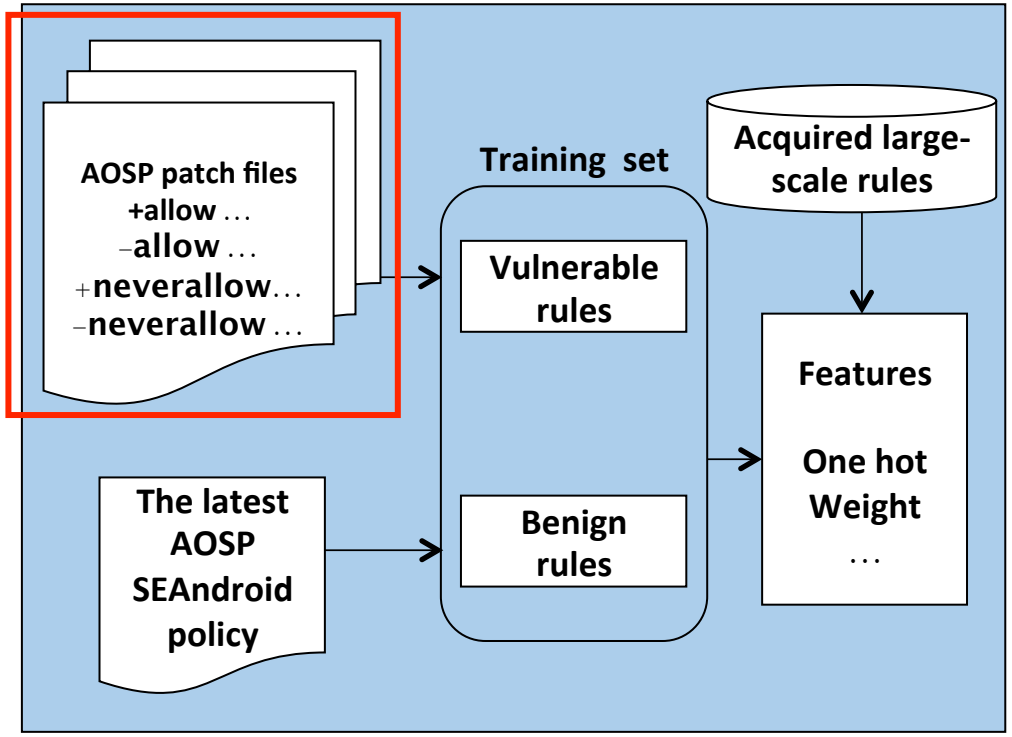


Special Symbols

- Special rules
  - Contains special symbols: '*' , '-' and '~'
- '*' match any item in the same field
- '-' is except operations
  - Performed on the subject or object
- '~' also refers to except operations
  - Used for the permission

VSPMiner Architecture

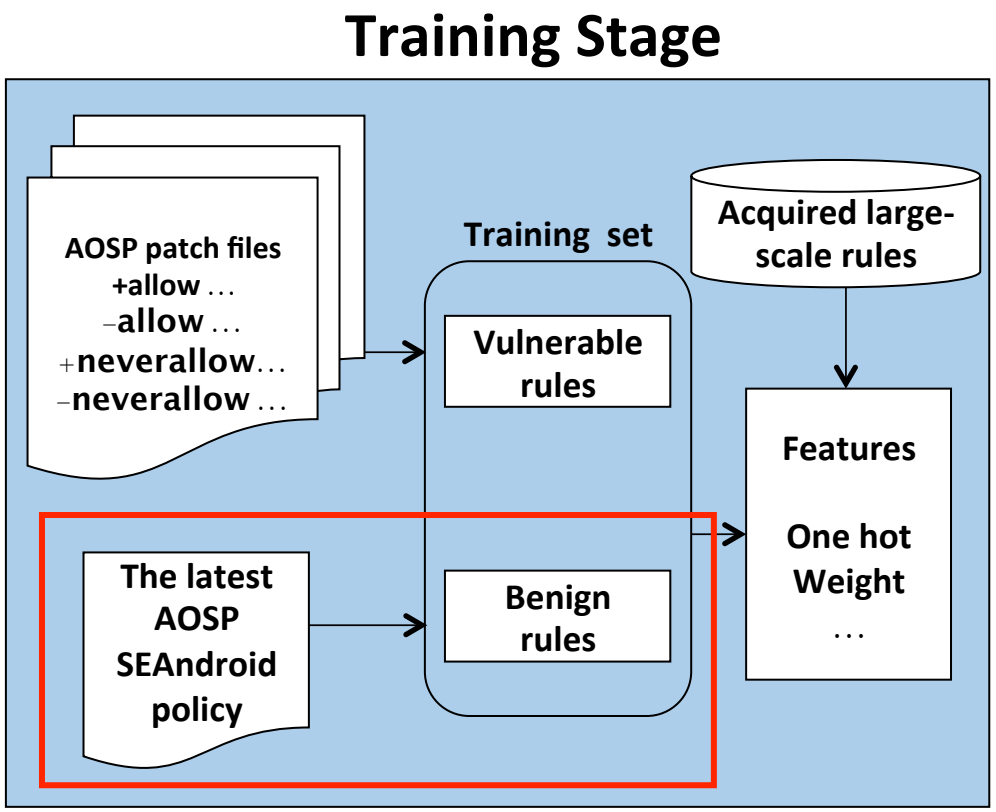
Training Stage



Handle Special Symbols

- Define critical field set: subject, object, object class and permission
  - That has appeared in common rules of P'_r
  - S_{sbj}, S_{obj}, S_{obj_class}, S_{perm}
- Replacing
  - '*' -> all the corresponding critical fields
  - '-' -> all the other critical subjects/objects
  - '~' -> all the other permissions
- P_r denotes the known vulnerable rules

VSPMiner Architecture

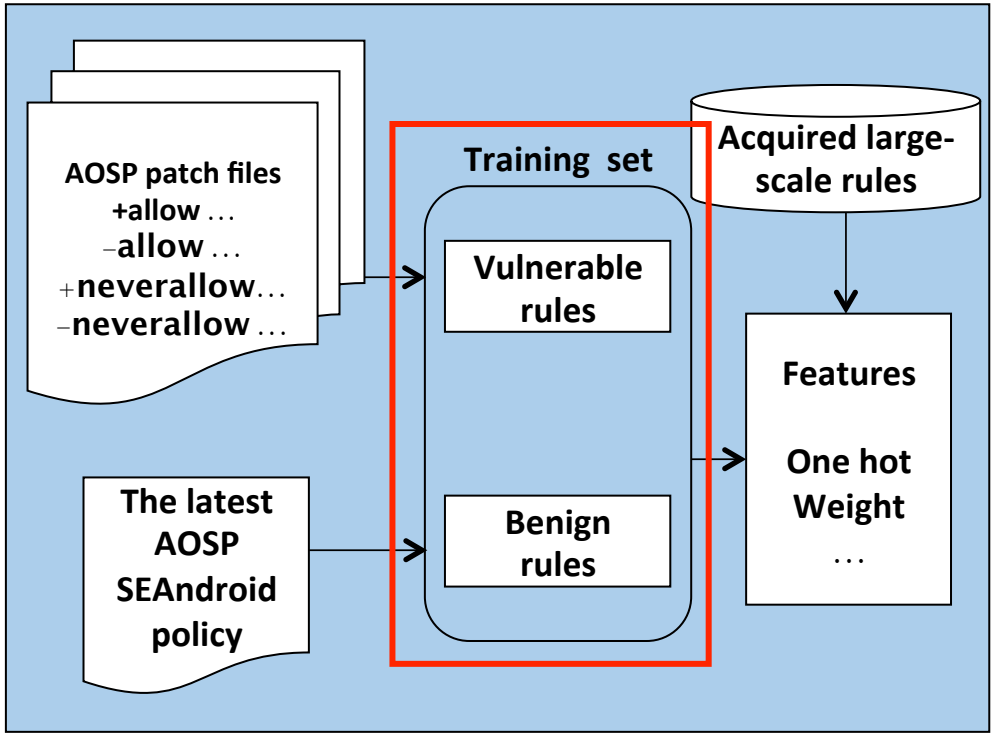


Obtain Benign Rules

- Acquire the latest AOSP SEAndroid
- Only consider the rules that consists of elements in the critical field set
  - R_{Benign}

VSPMiner Architecture

Training Stage



Training Set

- Eliminate the inference
- Positive samples

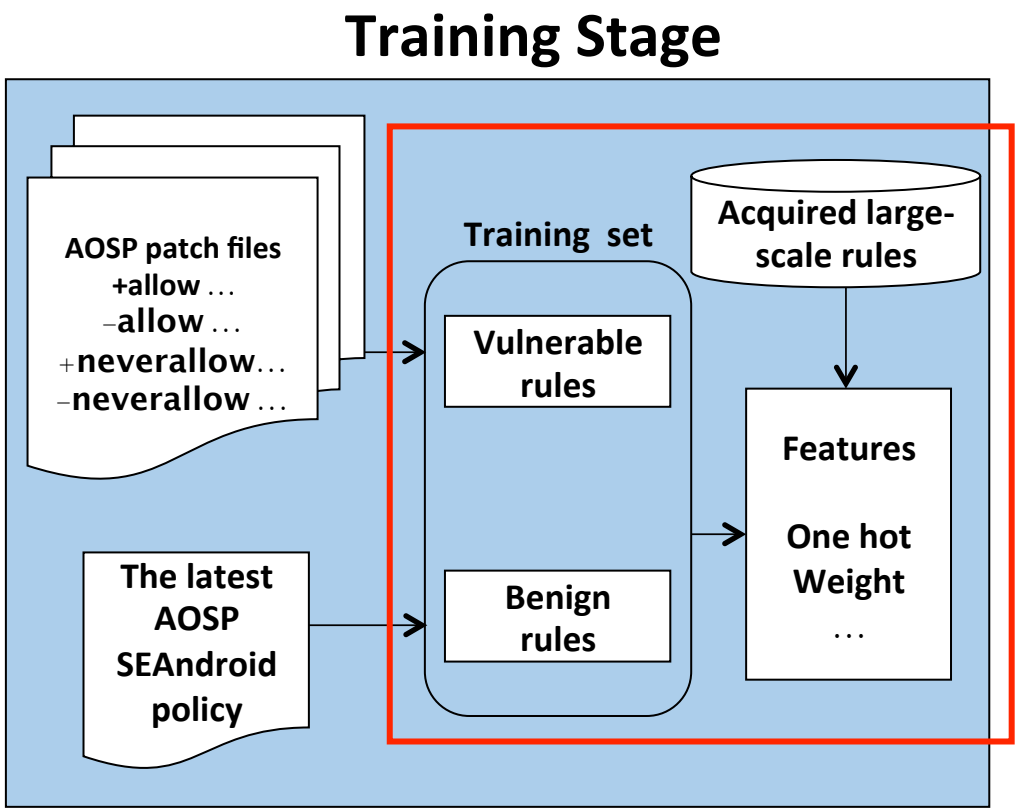
$$P_r - P_r \cap R_{Benign}$$

  - 25467 vulnerable rules
- Negative samples

$$R_{Benign} - P_r \cap R_{Benign}$$

  - 24146 benign rules

VSPMiner Architecture

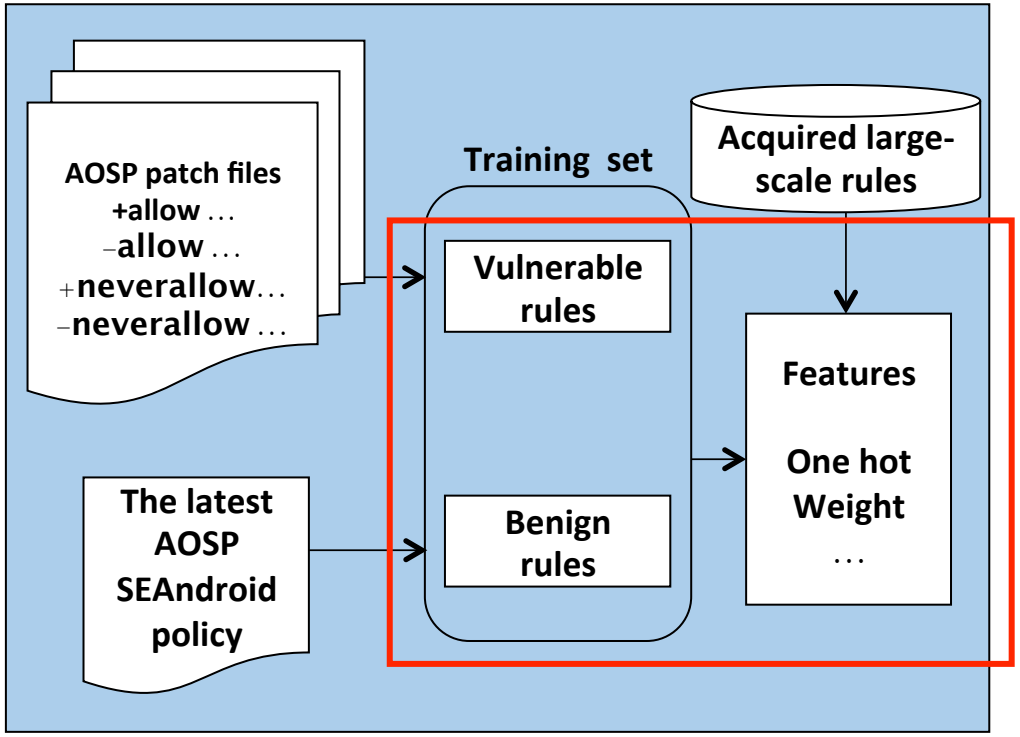


Feature Extraction

- One hot encoding for each field
  - Not enough
  - Lack in-depth understanding
- Weight information (hard to determine)
  - From vulnerable rules
  - From large-scale rules

VSPMiner Architecture

Training Stage

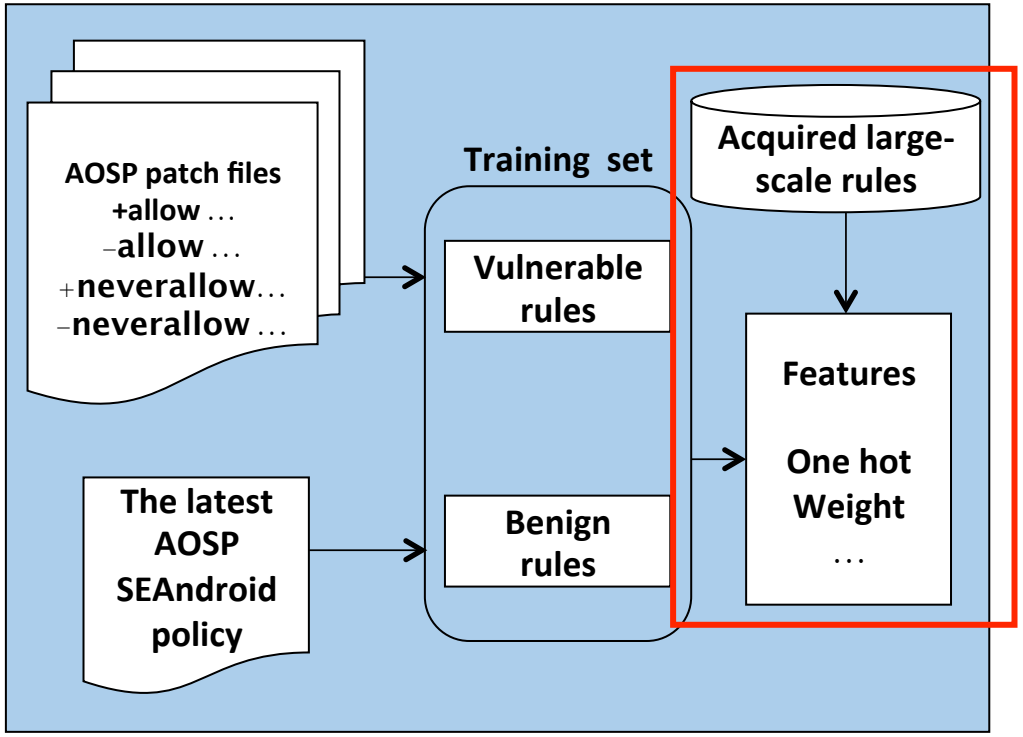


Feature Extraction

- From vulnerable rules
- Frequency information
  - Count of each field
    - *sbj_cnt, obj_cnt, obj_class_cnt, perm_cnt*
  - Count of combinations
    - *obj_perm_cnt*
  - Normalization
    - *sbj_weight, obj_weight, obj_class_weight, perm_weight, obj_perm_weight*

VSPMiner Architecture

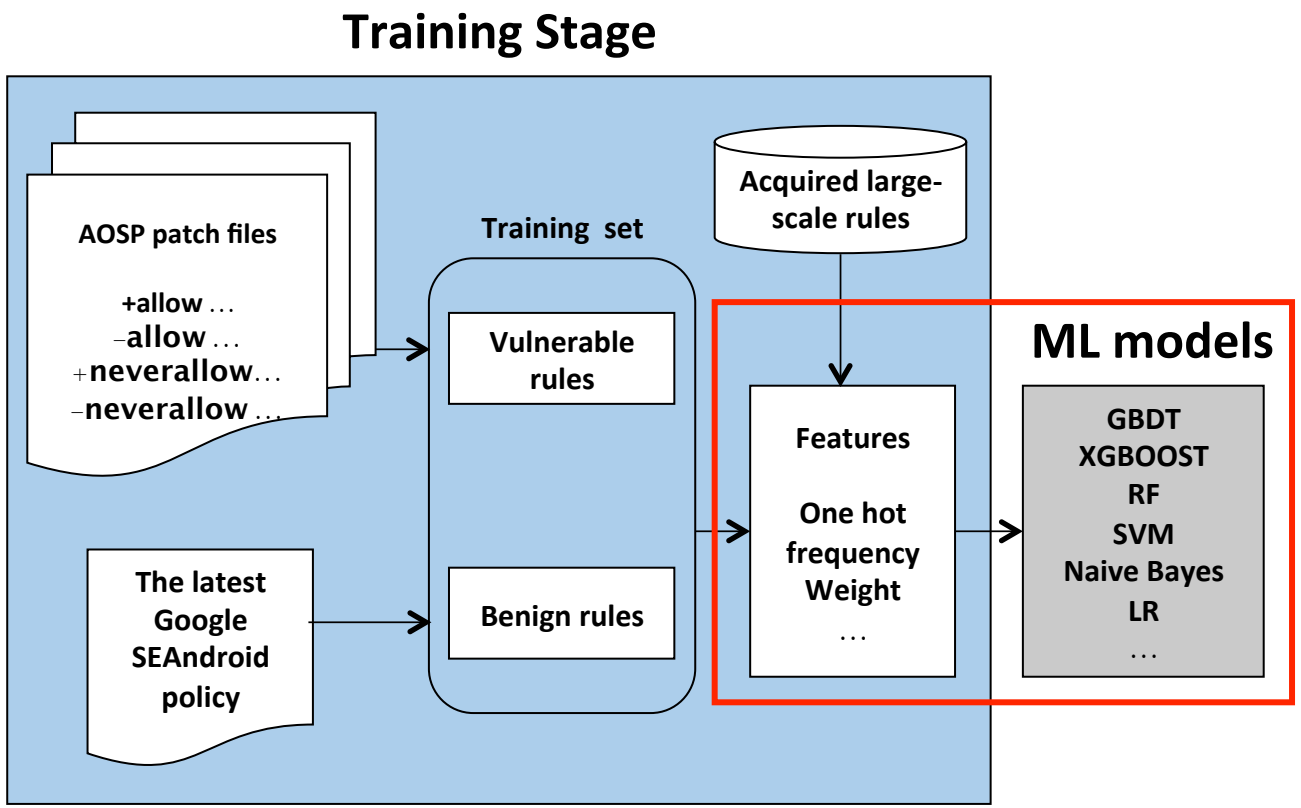
Training Stage



Feature Extraction

- From large-scale rules
- Frequency information
  - Count of each field in all distinct rules
    - *sbj_cnt, obj_cnt, obj_class_cnt, perm_cnt*
  - Count of each rule in all images
    - *rule_cnt*
  - Nomalization
    - *sbj_freq, obj_freq, obj_class_freq, perm_freq, rule_freq*

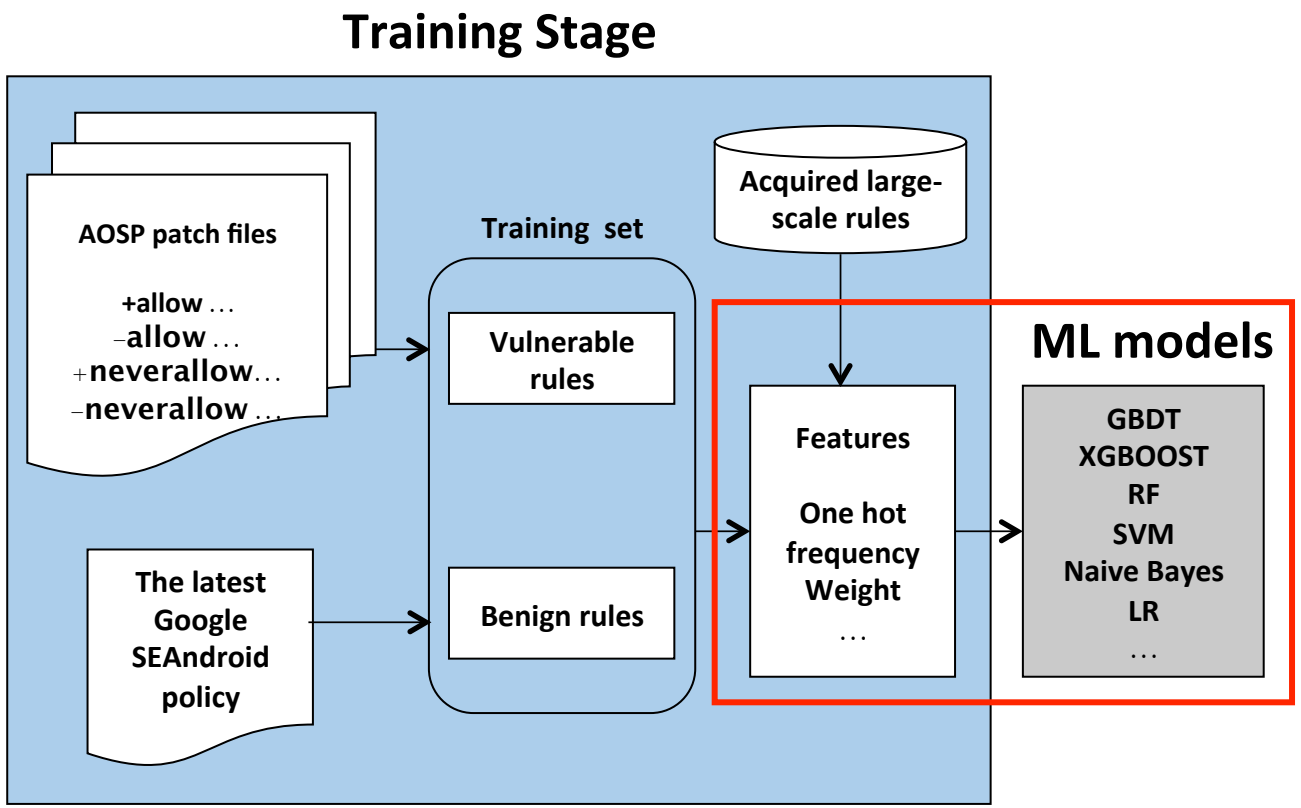
VSPMiner Architecture



Model Training

- Proposed Algorithm
  - Random Forest
  - GBDT
  - XGBOOST
  - SVM
  - Naive Bayes
  - Logistic Regression
  - KNN
- Null values
  - *obj_perm_weight*, *rule_freq*
  - Replaced by mean value

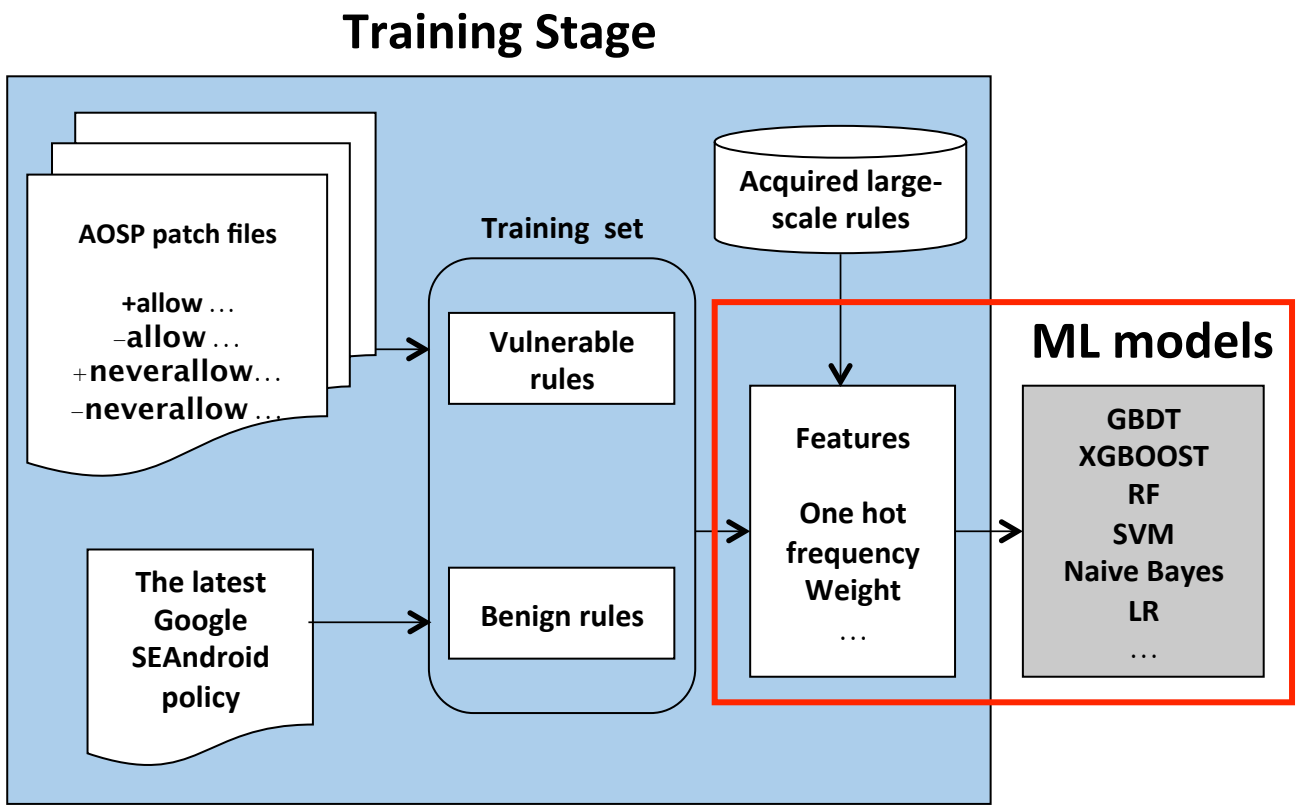
VSPMiner Architecture



Cross-Validation

- Training set : Validation set = 8 : 2
- Key criteria
  - Recall
  - F1-score
  - AUC
  - KS

VSPMiner Architecture

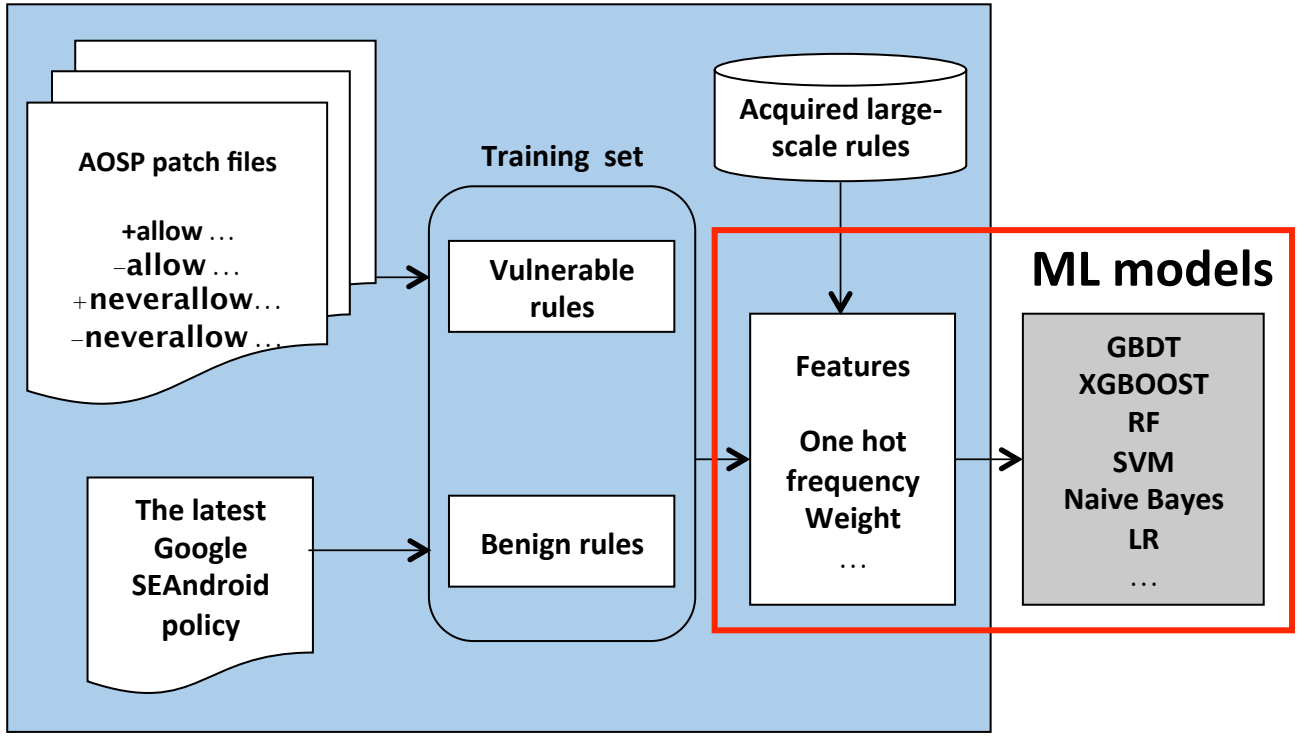


Cross-Validation

Classifier	Recall	F1-score	AUC	KS
GDBT	0.9927	0.9908	0.9994	0.9811
XGBOOST	0.9953	0.9939	0.9996	0.9876
Random Forest	0.9917	0.9908	0.9995	0.9812
KNN	0.9386	0.9545	0.9905	0.9096
SVM	0.8826	0.9098	0.971	0.819
Naivebayes	0.7535	0.8535	0.9089	0.7421
Logistic Regression	0.8885	0.9112	0.9717	0.8233

VSPMiner Architecture

Training Stage

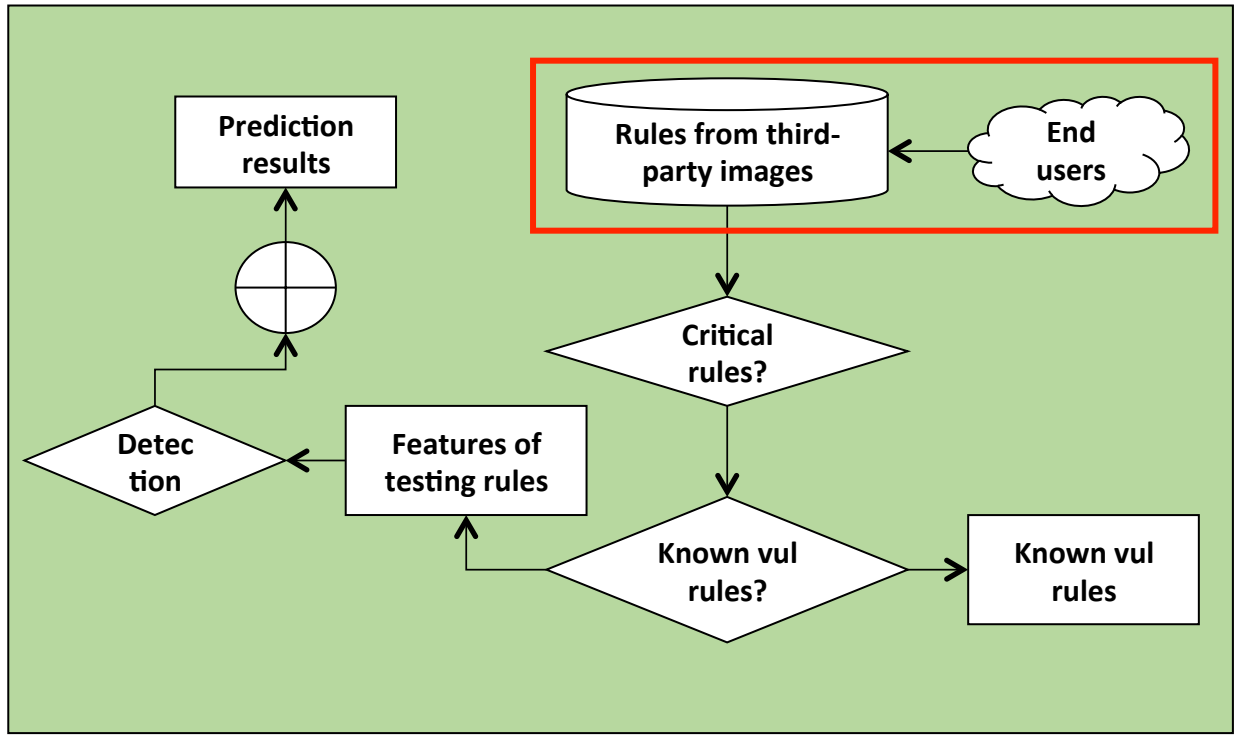


Cross-Validation

Classifier	Recall	F1-score	AUC	KS
GDBT	0.9927	0.9908	0.9994	0.9811
XGBOOST	0.9953	0.9939	0.9996	0.9876
Random Forest	0.9917	0.9908	0.9995	0.9812
KNN	0.9386	0.9545	0.9905	0.9096
SVM	0.8826	0.9098	0.971	0.819
Naivebayes	0.7535	0.8535	0.9089	0.7421
Logistic Regression	0.8885	0.9112	0.9717	0.8233

VSPMiner Architecture

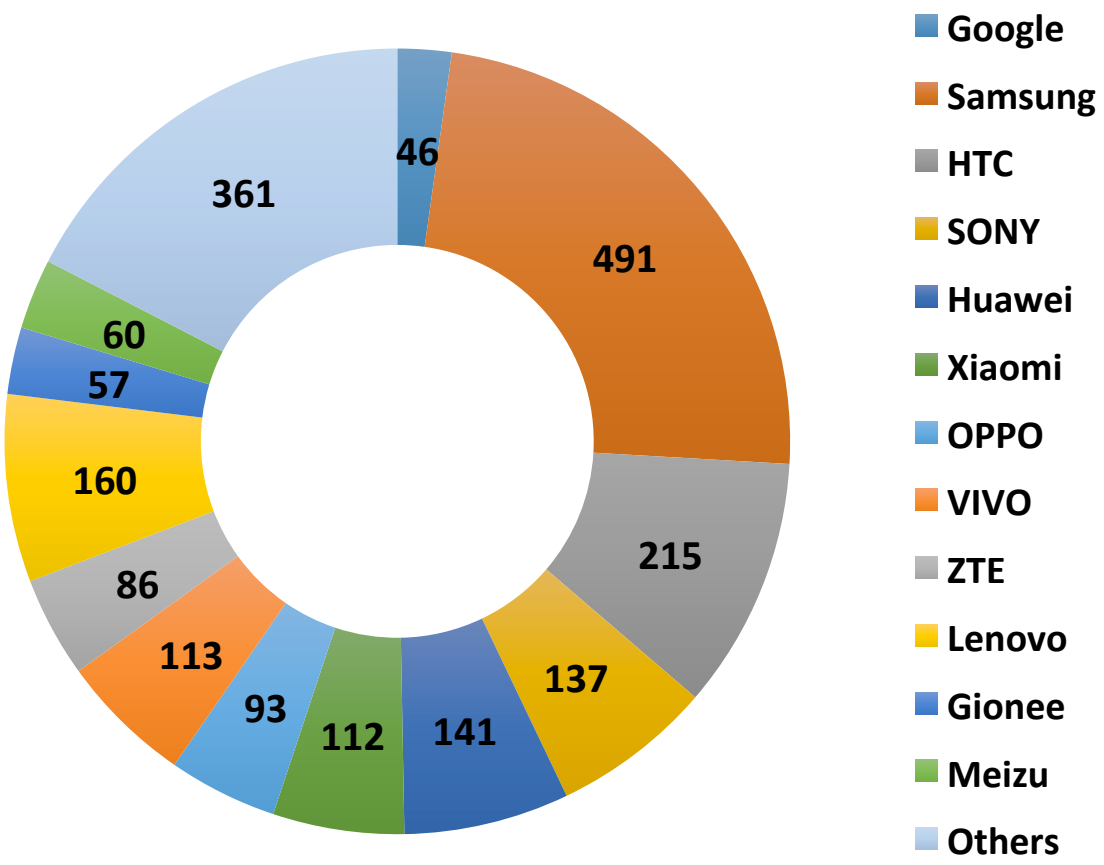
Detection Stage



Large-scale Rules Acquisition

- Qian Dun(钱盾) @Alibaba Security
- Anonymous + user agreement
- Information:
  - <Brand, model, version, sbj, obj, obj_class, perm>
  - Filecontext
    - Object <=> specific content

VSPMiner Architecture

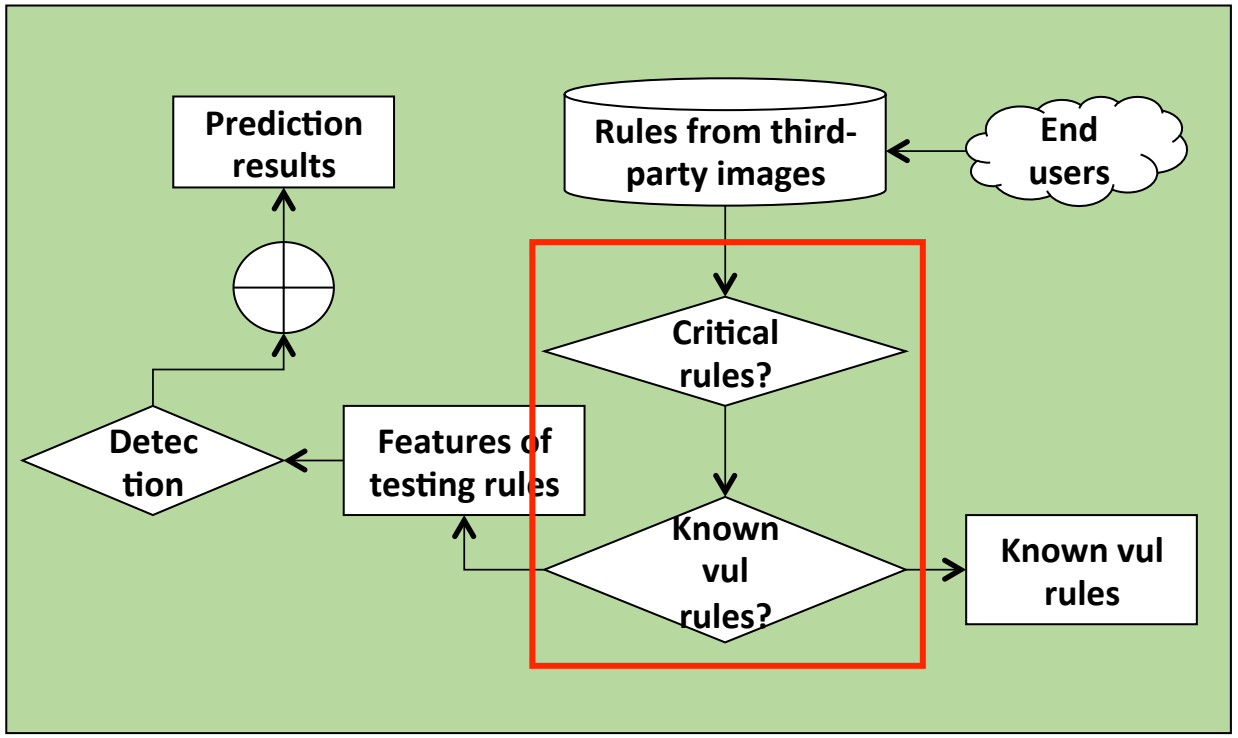


Data Source

- The others contains: Coolpad, LG, DOOV, Smatisan, Meitu, OnePlus, TCL, Leeco, Nubia, Qiku
- 22 brand, 2072 different images
- 4870838 distinct rules

VSPMiner Architecture

Detection Stage

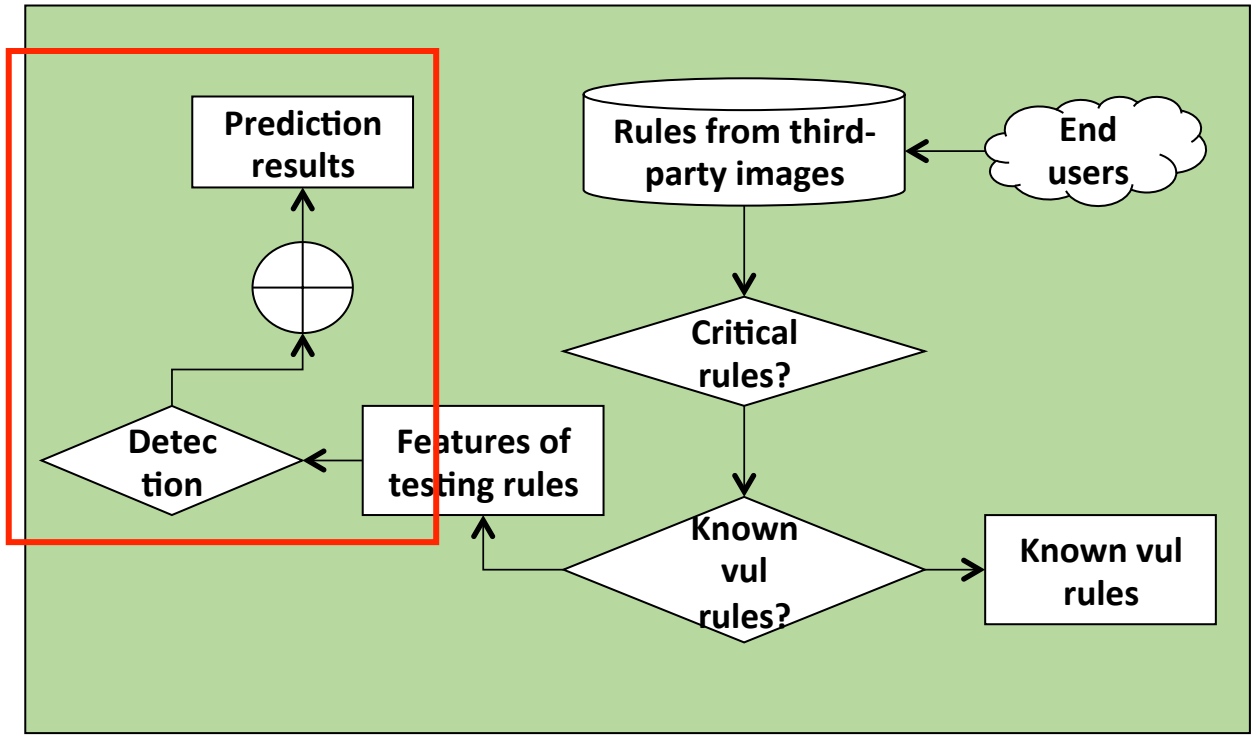


Data Filtering

- Selecting critical rules
  - Removing Google
  - Using the critical field information
- Filter out known vulnerable rules
  - Using data in training set
- 233235 testing rules
- The rules that belong to known vulnerable rules will be investigated later.

VSPMiner Architecture

Detection Stage

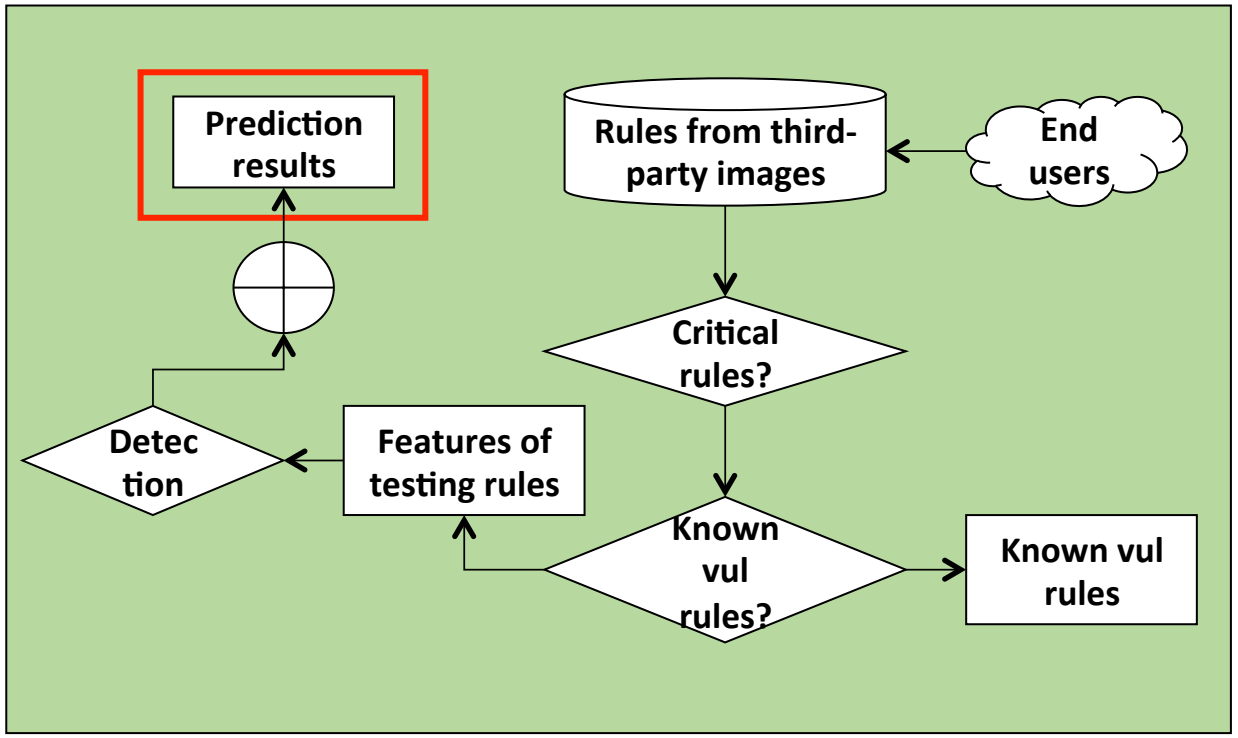


Prediction

- Models are used separately
  - GBDT, XGBOOST, RF
- Conservative approach
  - The rules are predicted as vulnerable in all the three models
- 132702 rules are predicted as vulnerable
  - 2832 problem access patterns (object, permission) are first revealed.

Evaluation

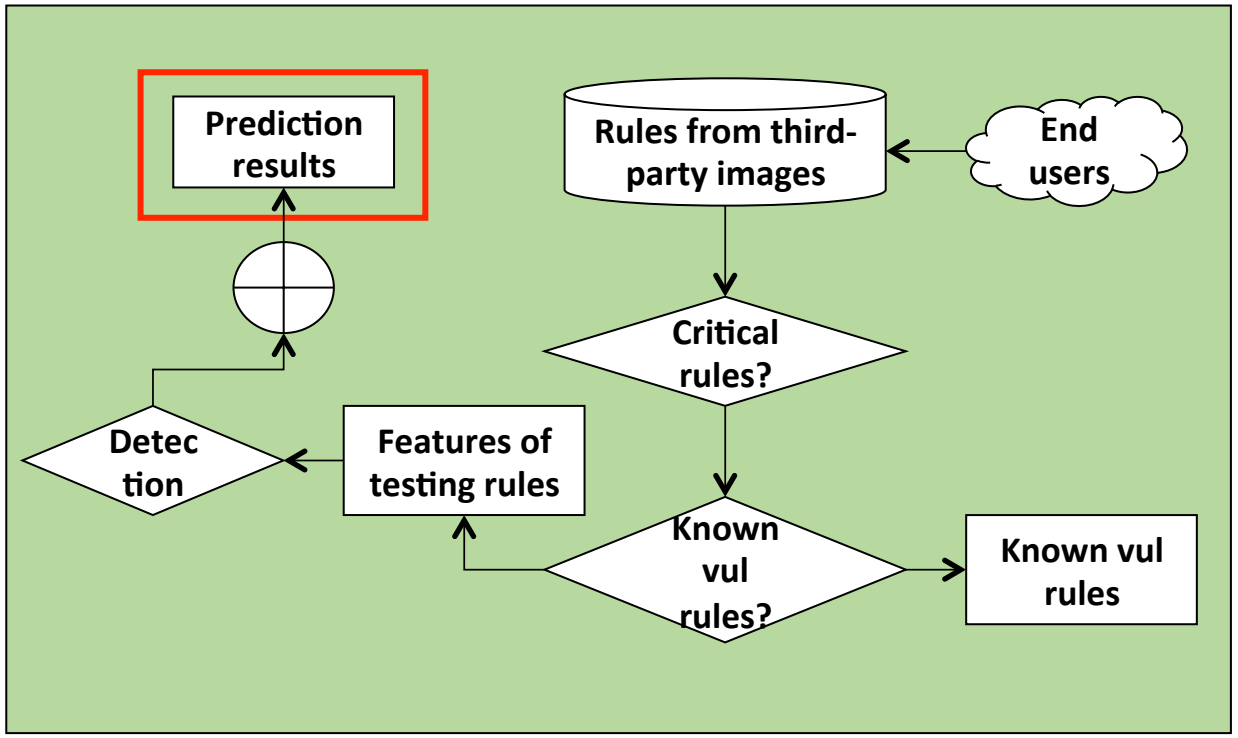
Detection Stage



Brand	# of vul rules	# of testing rules	Percent
COOLPAD	2919	10983	0.27
DOOV	1858	5785	0.32
GIONEE	3001	9472	0.32
HTC	13294	20523	0.65
HUAWEI	3223	11729	0.27
LEECO	8734	12607	0.69
LENOVO	3369	9196	0.37
LGE	2111	6297	0.34
MEITU	4032	12726	0.32
MEIZU	3855	9349	0.41
NUBIA	8924	12854	0.69
ONEPLUS	7640	10242	0.75
OPPO	11172	18093	0.62
QIKU	581	2009	0.29
SAMSUNG	123729	210249	0.59
SMARTISAN	277	717	0.39
SONY	11073	17312	0.64
TCL	2367	6334	0.37
VIVO	2483	6949	0.36
XIAOMI	10491	16420	0.64
ZTE	2758	7311	0.38

Evaluation

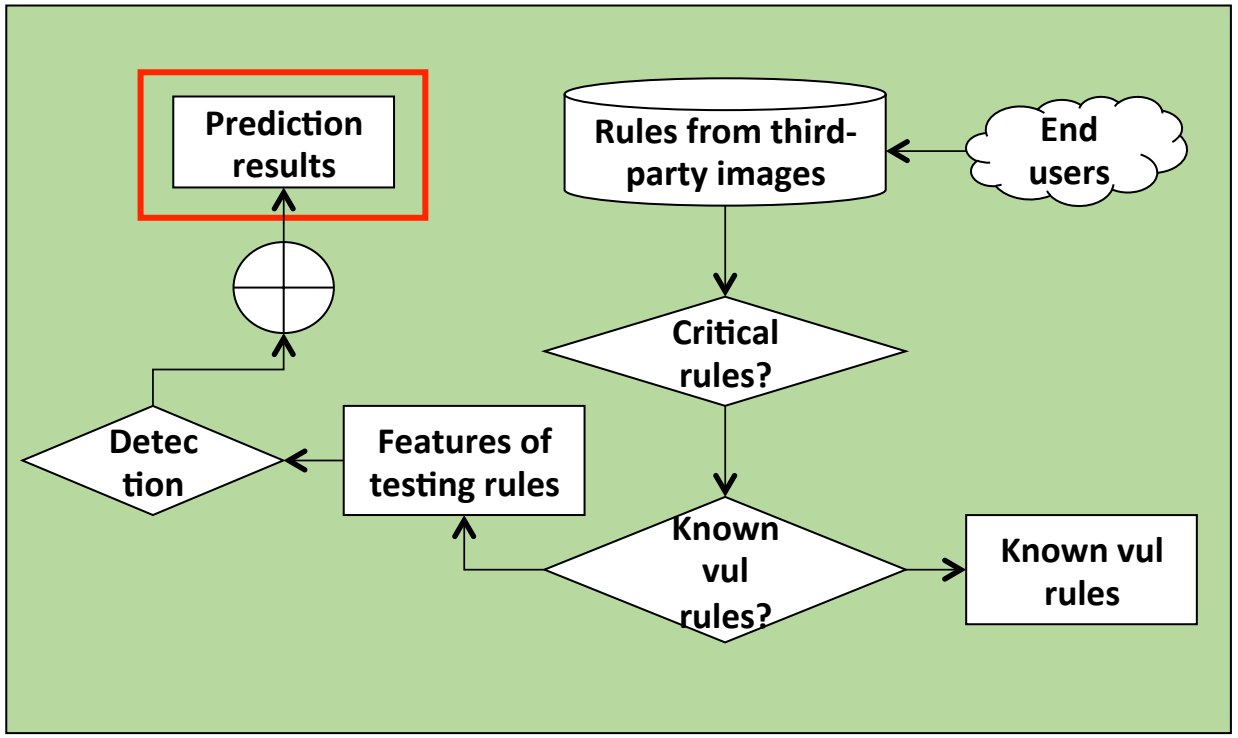
Detection Stage



Brand	# of vul rules	# of testing rules	Percent
COOLPAD	2919	10983	0.27
DOOV	1858	5785	0.32
GIONEE	3001	9472	0.32
HTC	13294	20523	0.65
HUAWEI	3223	11729	0.27
LEECO	8734	12607	0.69
LENOVO	3369	9196	0.37
LGE	2111	6297	0.34
MEITU	4032	12726	0.32
MEIZU	3855	9349	0.41
NUBIA	8924	12854	0.69
ONEPLUS	7640	10242	0.75
OPPO	11172	18093	0.62
QIKU	581	2009	0.29
SAMSUNG	123729	210249	0.59
SMARTISAN	277	717	0.39
SONY	11073	17312	0.64
TCL	2367	6334	0.37
VIVO	2483	6949	0.36
XIAOMI	10491	16420	0.64
ZTE	2758	7311	0.38

Evaluation

Detection Stage



Brand	# of vul rules	# of testing rules	Percent
COOLPAD	2919	10983	0.27
DOOV	1858	5785	0.32
GIONEE	3001	9472	0.32
HTC	13294	20523	0.65
HUAWEI	3223	11729	0.27
LEECO	8734	12607	0.69
LENOVO	3369	9196	0.37
LGE	2111	6297	0.34
MEITU	4032	12726	0.32
MEIZU	3855	9349	0.41
NUBIA	8924	12854	0.69
ONEPLUS	7640	10242	0.75
OPPO	11172	18093	0.62
QIKU	581	2009	0.29
SAMSUNG	123729	210249	0.59
SMARTISAN	277	717	0.39
SONY	11073	17312	0.64
TCL	2367	6334	0.37
VIVO	2483	6949	0.36
XIAOMI	10491	16420	0.64
ZTE	2758	7311	0.38

Evaluation

The distribution of problem access patterns

- We use the information of filecontext to do a transformation

Permission	/data	/system	/dev	/mnt	/adb_keys	/sys	/cache	/charger	/efs	/sdcard	/proc
link	7622	378	186	307	274	22	27	9	22	10	1
unlink	6842	312	133	257	247	21	17	12	11	11	0
create	6403	288	125	204	230	20	12	16	4	8	0
append	5572	253	97	211	194	20	12	10	4	5	0
write	5322	236	272	227	171	38	22	20	8	11	23
read	4296	234	226	175	144	27	23	3	10	10	0
open	2132	199	123	61	43	25	6	3	0	7	0
execute	363	119	193	34	5	29	16	17	11	5	3
ioctl	495	50	101	24	5	21	5	2	0	3	0
execmod	359	73	178	29	6	27	6	6	8	3	4
execute_no_trans	169	94	170	7	3	16	9	10	9	4	4
lock	322	36	60	24	1	19	2	2	0	4	0
search	261	81	23	18	6	5	2	0	0	1	0

Evaluation

The distribution of problem access patterns

- We use the information of filecontext to do a transformation

Permission	/data	/system	/dev	/mnt	/adb_keys	/sys	/cache	/charger	/efs	/sdcard	/proc
link	7622	378	186	307	274	22	27	9	22	10	1
unlink	6842	312	133	257	247	21	17	12	11	11	0
create	6403	288	125	204	230	20	12	16	4	8	0
append	5572	253	97	211	194	20	12	10	4	5	0
write	5322	236	272	227	171	38	22	20	8	11	23
read	4296	234	226	175	144	27	23	3	10	10	0
open	2132	199	123	61	43	25	6	3	0	7	0
execute	363	119	193	34	5	29	16	17	11	5	3
ioctl	495	50	101	24	5	21	5	2	0	3	0
execmod	359	73	178	29	6	27	6	6	8	3	4
execute_no_trans	169	94	170	7	3	16	9	10	9	4	4
lock	322	36	60	24	1	19	2	2	0	4	0
search	261	81	23	18	6	5	2	0	0	1	0

Evaluation

The distribution of problem access patterns (process)

Permission	Obj_class	Num
dyntransition	process	319
transition	process	308
getattr	process	107
ptrace	process	91
execstack	process	86
execheap	process	79
sigchld	process	79
sigkill	process	61
signal	process	59
setsched	process	55
setsockcreate	process	49
getsched	process	47
execmem	process	47
setfscreate	process	46
setexec	process	46
noatsecure	process	45
share	process	44
setcurrent	process	44
sigstop	process	37
signull	process	34
getpgid	process	32
setpgid	process	29
fork	process	24

Evaluation

The distribution of problem access patterns (process)

Permission	Obj_class	Num
dyntransition	process	319
transition	process	308
getattr	process	107
ptrace	process	91
execstack	process	86
execheap	process	79
sigchld	process	79
sigkill	process	61
signal	process	59
setsched	process	55
setsockcreate	process	49
getsched	process	47
execmem	process	47
setfscreate	process	46
setexec	process	46
noatsecure	process	45
share	process	44
setcurrent	process	44
sigstop	process	37
signull	process	34
getpgid	process	32
setpgid	process	29
fork	process	24

- EASEAndroid
  - {transition,dyntransition} process

Evaluation

The distribution of problem access patterns (process)

Permission	Obj_class	Num
dyntransition	process	319
transition	process	308
getattr	process	107
ptrace	process	91
execstack	process	86
execheap	process	79
sigchld	process	79
sigkill	process	61
signal	process	59
setsched	process	55
setsockcreate	process	49
getsched	process	47
execmem	process	47
setfscreate	process	46
setexec	process	46
noatsecure	process	45
share	process	44
setcurrent	process	44
sigstop	process	37
signull	process	34
getpgid	process	32
setpgid	process	29
fork	process	24

- EASEAndroid
  - {transition,dyntransition} process
- VSPMiner
  - More patterns

Evaluation

The distribution of problem access patterns (process)

Permission	Obj_class	Num
dyntransition	process	319
transition	process	308
getattr	process	107
ptrace	process	91
execstack	process	86
execheap	process	79
sigchld	process	79
sigkill	process	61
signal	process	59
setsched	process	55
setsockcreate	process	49
getsched	process	47
execmem	process	47
setfscreate	process	46
setexec	process	46
noatsecure	process	45
share	process	44
setcurrent	process	44
sigstop	process	37
signull	process	34
getpgid	process	32
setpgid	process	29
fork	process	24

Typical exploit patterns

Evaluation

The distribution of problem access patterns (capability)

Permission	Obj_class	Num
dac_read_search	capability	13
sys_boot	capability	11
sys_tty_congfig	capability	10
dac_override	capability	10
ipc_lock	capability	10
setpcap	capability	9
sys_time	capability	9
sys_rawio	capability	9
sys_admin	capability	9
audit_write	capability	9
fsetid	capability	8
net_broadcast	capability	8
net_bind_service	capability	8
net_admin	capability	8
sys_module	capability	8
sys_chroot	capability	8
fowner	capability	7
chown	capability	7
sys_resource	capability	7
sys_ptrace	capability	7
sys_nice	capability	7
net_raw	capability	7
mknod	capability	7
kill	capability	6
setgid	capability	6
setuid	capability	6

Evaluation

The distribution of problem access patterns (capability)

Permission	Obj_class	Num
dac_read_search	capability	13
sys_boot	capability	11
sys_tty_congfig	capability	10
dac_override	capability	10
ipc_lock	capability	10
setpcap	capability	9
sys_time	capability	9
sys_rawio	capability	9
sys_admin	capability	9
audit_write	capability	9
fsetid	capability	8
net_broadcast	capability	8
net_bind_service	capability	8
net_admin	capability	8
sys_module	capability	8
sys_chroot	capability	8
fowner	capability	7
chown	capability	7
sys_resource	capability	7
sys_ptrace	capability	7
sys_nice	capability	7
net_raw	capability	7
mknod	capability	7
kill	capability	6
setgid	capability	6
setuid	capability	6

- EASEAndroid
  - {kill,sys_admin,sys_ptrace,sys_chroot,setuid,setgid} capability

Evaluation

The distribution of problem access patterns (capability)

Permission	Obj_class	Num
dac_read_search	capability	13
sys_boot	capability	11
sys_tty_congfig	capability	10
dac_override	capability	10
ipc_lock	capability	10
setpcap	capability	9
sys_time	capability	9
sys_rawio	capability	9
sys_admin	capability	9
audit_write	capability	9
fsetid	capability	8
net_broadcast	capability	8
net_bind_service	capability	8
net_admin	capability	8
sys_module	capability	8
sys_chroot	capability	8
fowner	capability	7
chown	capability	7
sys_resource	capability	7
sys_ptrace	capability	7
sys_nice	capability	7
net_raw	capability	7
mknod	capability	7
kill	capability	6
setgid	capability	6
setuid	capability	6

- EASEAndroid
  - {kill,sys_admin,sys_ptrace,sys_chroot,setuid,setgid} capability
- VSPMiner
  - More patterns

Evaluation

The examples of vulnerable access patterns (devices)

Subject	Object	Object_class	permission
init	ion_device	process	dyntransition
logd	ion_device	chr_file	execute
logd	ion_device	chr_file	mounton
init	gpu_device	process	transition
shared_app	gpu_device	chr_file	execute
system_server	gpu_device	chr_file	execute
appdomain	input_device	chr_file	write
healthd	input_device	chr_file	write
untrusted_app	input_device	chr_file	write
appdomain	audio_device	chr_file	getattr
shell	audio_device	dir	search
untrusted_app	audio_device	dir	search

Evaluation

The examples of vulnerable access patterns (devices)

Subject	Object	Object_class	permission
init	ion_device	process	dyntransition
logd	ion_device	chr_file	execute
logd	ion_device	chr_file	mounton
init	gpu_device	process	transition
shared_app	gpu_device	chr_file	execute
system_server	gpu_device	chr_file	execute
appdomain	input_device	chr_file	write
healthd	input_device	chr_file	write
untrusted_app	input_device	chr_file	write
appdomain	audio_device	chr_file	getattr
shell	audio_device	dir	search
untrusted_app	audio_device	dir	search

Evaluation

The examples of vulnerable access patterns (devices)

Subject	Object	Object_class	permission
init	ion_device	process	dyntransition
logd	ion_device	chr_file	execute
logd	ion_device	chr_file	mounton
init	gpu_device	process	transition
shared_app	gpu_device	chr_file	execute
system_server	gpu_device	chr_file	execute
appdomain	input_device	chr_file	write
healthd	input_device	chr_file	write
untrusted_app	input_device	chr_file	write
appdomain	audio_device	chr_file	getattr
shell	audio_device	dir	search
untrusted_app	audio_device	dir	search

Evaluation

The examples of vulnerable access patterns (devices)

Subject	Object	Object_class	permission
init	ion_device	process	dyntransition
logd	ion_device	chr_file	execute
logd	ion_device	chr_file	mounton
init	gpu_device	process	transition
shared_app	gpu_device	chr_file	execute
system_server	gpu_device	chr_file	execute
appdomain	input_device	chr_file	write
healthd	input_device	chr_file	write
untrusted_app	input_device	chr_file	write
appdomain	audio_device	chr_file	getattr
shell	audio_device	dir	search
untrusted_app	audio_device	dir	search

Evaluation

The examples of vulnerable access patterns (devices)

Subject	Object	Object_class	permission
init	ion_device	process	dyntransition
logd	ion_device	chr_file	execute
logd	ion_device	chr_file	mounton
init	gpu_device	process	transition
shared_app	gpu_device	chr_file	execute
system_server	gpu_device	chr_file	execute
appdomain	input_device	chr_file	write
healthd	input_device	chr_file	write
untrusted_app	input_device	chr_file	write
appdomain	audio_device	chr_file	getattr
shell	audio_device	dir	search
untrusted_app	audio_device	dir	search

Evaluation

Case study: Assist privilege escalation

- *allow system_server system_data_file : file execute*
- Such rule is very helpful if we control *system_server*
  - Exploiting CVE-2015-1528, CVE-2016-5195 or CVE-2016-6707
- Otherwise, the exploit will be much more complex
  - For example, we need inject the *exp* into the memory of *system_server* process.

Evaluation

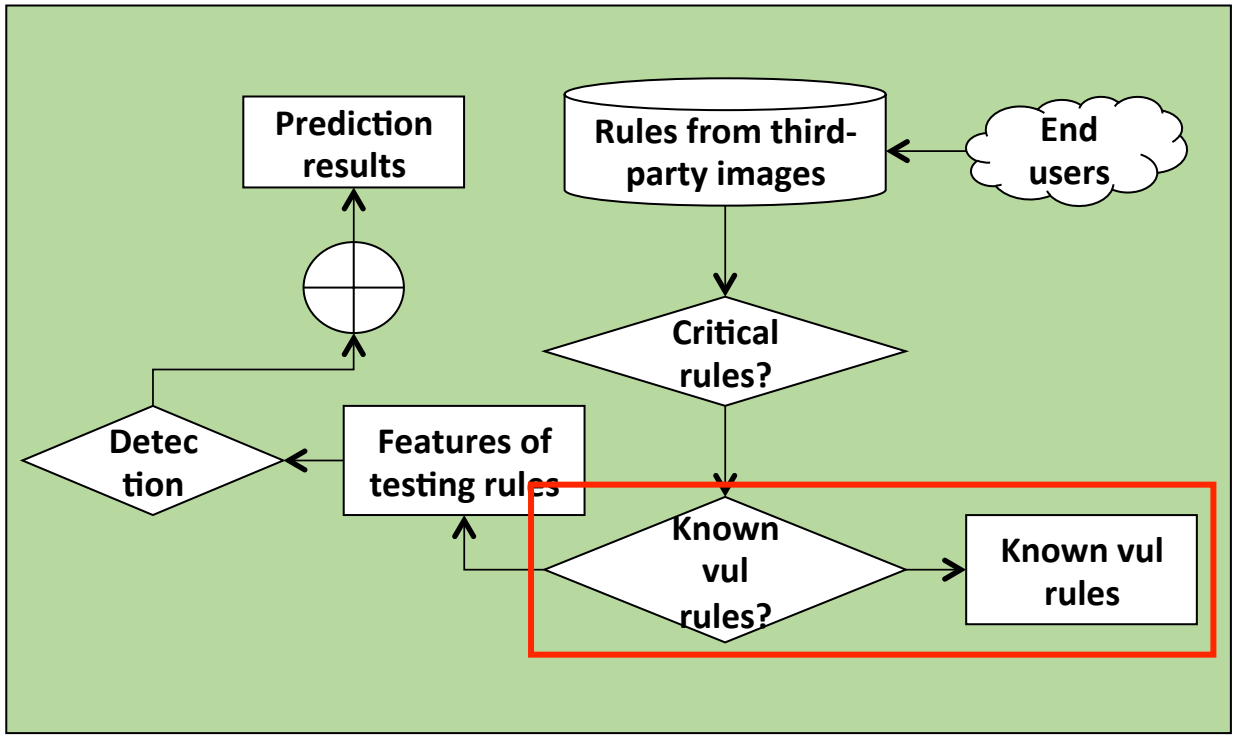
Typical exploit patterns

- obj: *system_file*, permission: *execute_no_trans*
- obj: *system_data_file*, permission: *execute*

Object	Filecontext	Permission	# of rules	Examples
system_file	/system(/*.)?	execute_no_trans	14	allow untrusted_app system_file: file execute_no_trans;
				allow shelldomain system_file: file execute_no_trans;
				allow debuggerd system_file : file execute_no_trans;
system_data_file	/data(/.)?	execute	27	allow shell system_data_file : file execute;
				allow untrusted_app system_data_file : file execute;
				allow system_server system_data_file : file execute;

Evaluation

Detection Stage

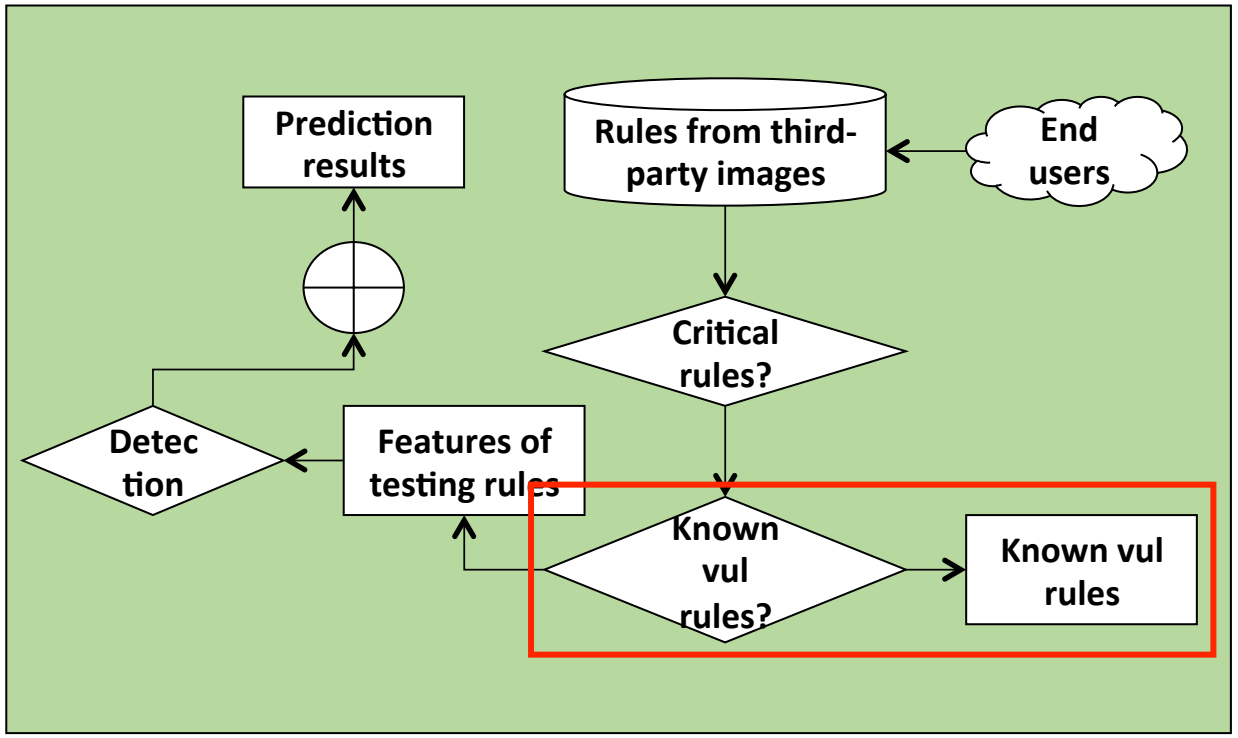


- The rule has been deleted in the AOSP
- **But** it also exists in a newer vendor version

Brand	# of later rules
COOLPAD	287
DOOV	44
GIONEE	286
HTC	128
HUAWEI	34
LEECO	106
LENOVO	136
LGE	64
MEITU	416
MEIZU	11122
NUBIA	109
ONEPLUS	86
OPPO	87
QIKU	17
SAMSUNG	233
SMARTISAN	47
SONY	124
TCL	84
VIVO	91
XIAOMI	11130
ZTE	11086

Evaluation

Detection Stage

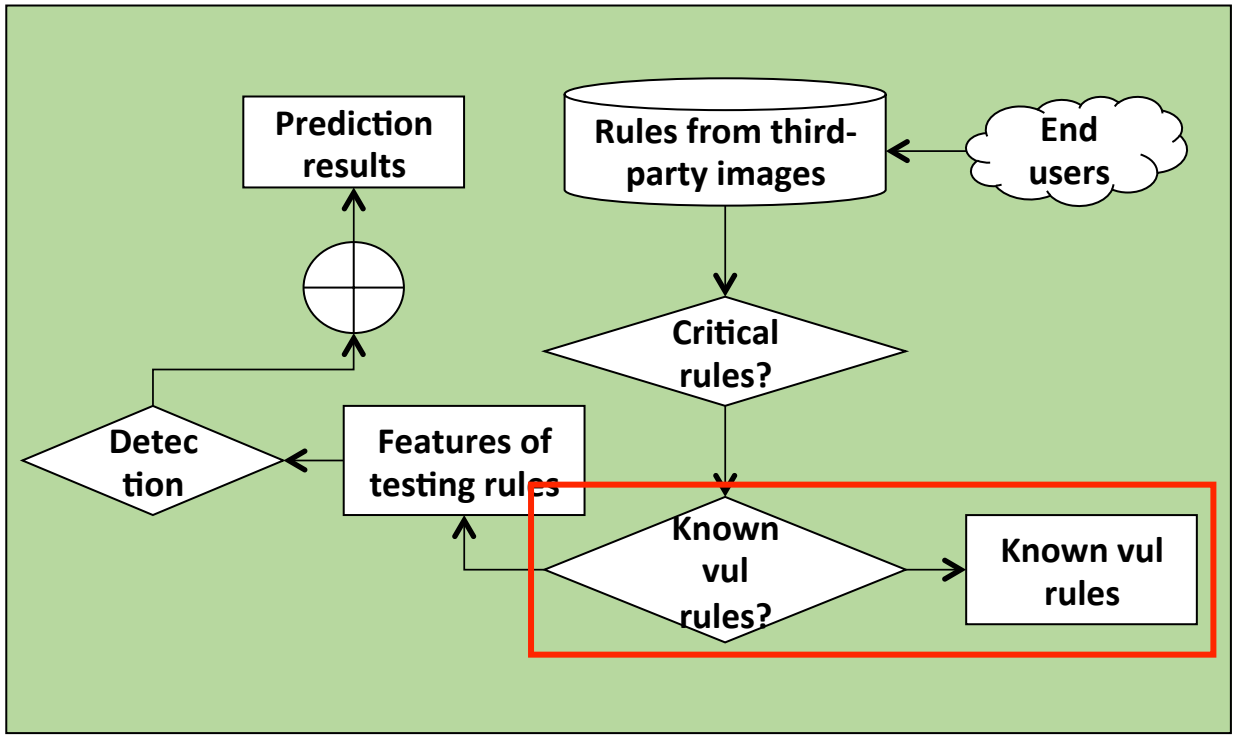


- The rule has been deleted in the AOSP
- **But** it also exists in a newer vendor version

Brand	# of later rules
COOLPAD	287
DOOV	44
GIONEE	286
HTC	128
HUAWEI	34
LEECO	106
LENOVO	136
LGE	64
MEITU	416
MEIZU	11122
NUBIA	109
ONEPLUS	86
OPPO	87
QIKU	17
SAMSUNG	233
SMARTISAN	47
SONY	124
TCL	84
VIVO	91
XIAOMI	11130
ZTE	11086

Evaluation

Detection Stage



- The rule has been deleted in the AOSP
- **But** it also exists in a newer vendor version

Brand	# of later rules
COOLPAD	287
DOOV	44
GIONEE	286
HTC	128
HUAWEI	34
LEECO	106
LENOVO	136
LGE	64
MEITU	416
MEIZU	11122
NUBIA	109
ONEPLUS	86
OPPO	87
QIKU	17
SAMSUNG	233
SMARTISAN	47
SONY	124
TCL	84
VIVO	91
XIAOMI	11130
ZTE	11086

Evaluation

Case study: Assist privilege escalation

- *allow init kernel : security load_policy*
  - Deleted since Android 6.0
  - But it is also exists in Samsung S7 with Android 6.x or later

- After controlling *init*, we can load a new policy

Summary

- An in-depth analysis on the state-of-the-art SEAndroid policy refining techniques and reveal their limitations.
- A new policy analysis tool, VSPMiner, to detect vulnerable SEAndroid policies in the wild by leveraging supervised machine learning.
- The results of VSPMiner suggest it is promising.
- As showcases, we demonstrate how to abuse vulnerable rules to assist privilege escalation.

Thanks