MACHINE LEARNING LAB


# Personalized, Browser-based Visual Phishing Detection Based on Deep Learning

Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, **Fabiano Tarlao**

# Phishing (1/3)

**Cheat** Internet **users**, **steal** sensitive **informations**

Phishing pages **identical to official** company pages

➔ login pages

There are ways to fight them, e.g. :

➔ **blacklisting** URLs

➔ URL **reputation**

# Phishing (2/3)

Phishing campaigns today

➔ focus on **small population** of users

➔ have **short lifetime**

**Traditional techniques** could be **ineffective**

➔ they need time to react, i.e:

➔ ..they need time for detecting and notify

# Phishing (3/3)

Need **defense** layer **close** to the user **browser**

Defense not based on

- ➔ Blacklisting
- ➔ URL-based reputation

**Complement** the **traditional techniques**

**Is** this type of defense **feasible**?

# Phishing Detector (PD)

We **investigate** feasibility of **detectors** that

- ➔ Protect **user-selected** set of websites
- ➔ Based on **webpage visual appearance**
- ➔ Integrates into the web browser, e.g. extension
- ➔ Simple and fast, **warn** user in **real time**

# Phishing Detector (Prerequisite)

**user-selected** set **S**, **sites** to protect

PD have **prior knowledge**:

→ **<protocol,domain>** for each **user-selected site**

# Phishing Detector - Workflow

The **browser loads a page** p from URL

PD performs in order:

1. take **screenshot** of page p
2. **Classify the screenshot** to be one protected site $x \in S$
3. Check **URL** to **match <protocol,domain>** for site *x*

It acts **as wise human**..

# Phishing Detector - Workflow

The **browser loads a page** p from URL

PD performs in order:

1. take **screenshot** of page p
2. **Classify the screenshot** to be one protected site $x \in S$
3. Check **URL** to **match <protocol,domain>** for site *x*

The difficult part, the problem

# Problem Statement

We consider **static list** S of protected **websites**

- ➔ $\#S = n,\ S = \{s_1,..,s_n\}$
- ➔ ..selected by the user

Problem **input** is a login **webpage screenshot**

Problem **output** are ***n+1* categorical values**:

- ➔ $y = j$ site j (< n+1) selected for protection
- ➔ $y = n+1$ sites NOT selected for protection

# Our Approach

**Solution** based on **Deep Learning**

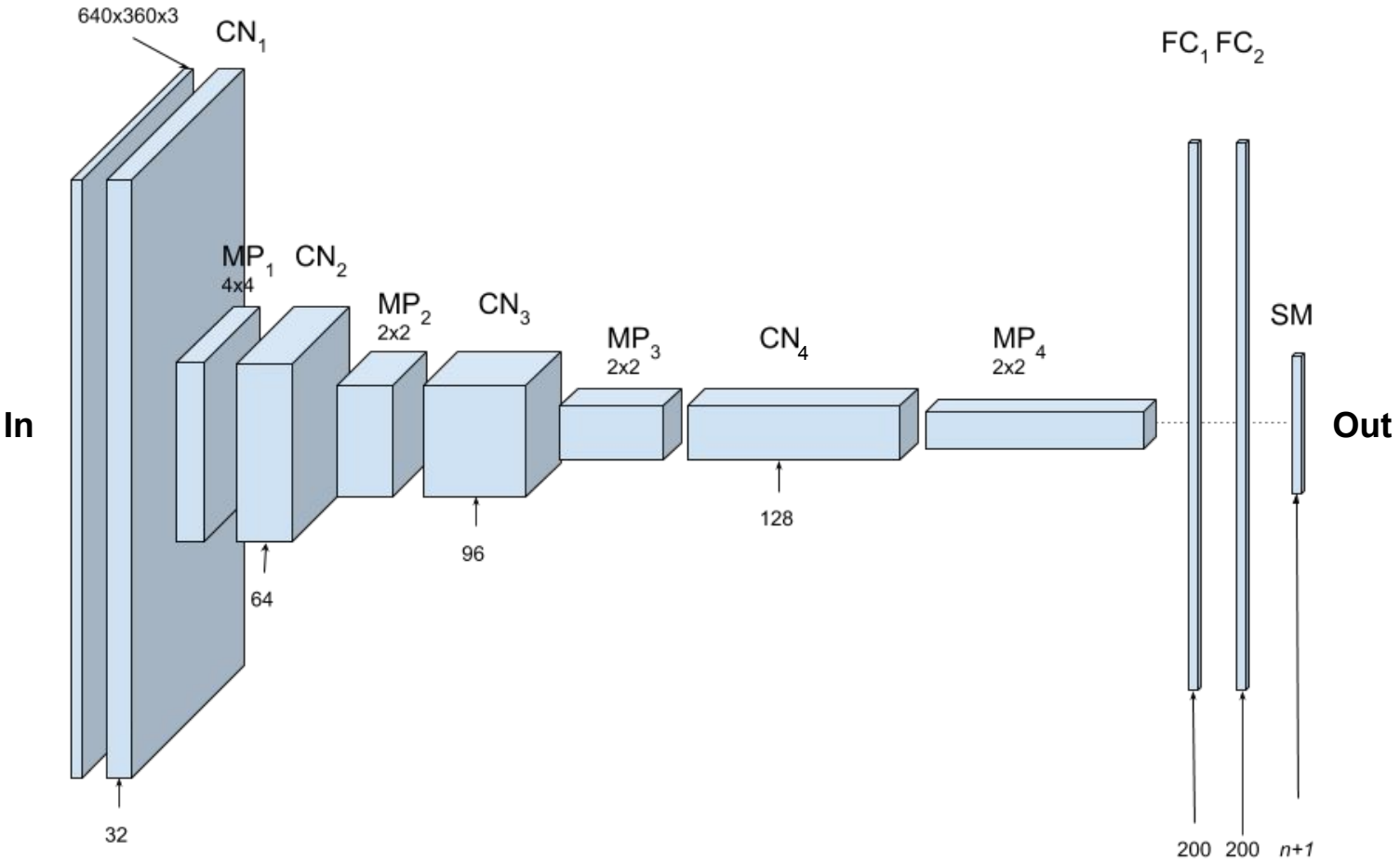
- ➔ Multiclass classifier
- ➔ Convolutional Neural Network (CNN)
- ➔ Implementation: Keras

**Input RGB image** *x*, CNN input layer:

- ➔ 640x360 (16:9) - 3 channels
- ➔ Obtained by cropping/extending and resizing *x*

**Output** is a class ***{1,..,n+1}***, **one-hot** encoded

# CNN Architecture



# Data

We acquired **dataset *D***

➔ **1500 screenshots** of official login pages

➔ from **100 websites**

100 websites:

➔ 30 most targeted (security reports)

➔ 70 most visited (Alexa ranking)

For each login page we have 15 screenshots

➔ 15 browser window **resolutions**

# Data Augmentation

Data augmentation

- Deep Learning: **need big datasets**

Generate a new screenshot from original one:

1. **crop/extend** original screenshot to 16:9
2. **circular shift** of random vertical/horizontal quantity
3. 50% cases, apply random **centered zoom**
4. 50% cases, **lighten** or **darken** image

# Training (1/3)

Select subset *S* of protected websites $\{s_1,..,s_n\}$

➔ *Categories {1,..,n+1}*

Partition *D* in three balanced sets

➔ *Training,* ***T***

➔ *Validation,* ***V***

➔ *Testing,* ***E***

Train model on **augmented data** $T_a$, $V_a$

# Training (2/3)

At **each epoch** randomly **generate** augmented $\boldsymbol{T_a}$

➔ 50% screenshot *x(s), s*$\in$*S*

➔ 50% screenshot *x(s), s*$\notin$*S*… i.e, class *n+1*

**Generate** augmented validation $\boldsymbol{V_a}$ (**once**)

➔ the same for all epochs

Testing **E**, **no augmentation**

# Training (3/3)

## Training details:

- Batch size, 32
- Stochastic Gradient Descent (SGD)
- Momentum, 0.9
- Learning rate, 0.1
- Gradient clipping, maximum norm value 1.0
- Dropout, probability 0.1 after CN layer, 0.3 after FC
- Epochs, 400
- Steps per training epoch, $12 \bullet (n+1) \bullet c_t$

## Validation at the end of each epoch

## The outcome is the model that performed better on $V_a$

# Experimental evaluation - Indexes

Evaluated the effectiveness of trained CNN on E:

- **CA**, Classification Accuracy
- **BCA** Balanced Classification Accuracy
- **MAR** Missed Alarm Ratio
  - $s \in S$ classified as $s \notin S$
- **$FAR_I$** False Alarm Ratio
  - $s_i \in S$ classified as a different $s_j \in S$
- **$FAR_U$** False Alarm Ratio
  - $s \notin S$ classified as $s_i \in S$
- **ET** Elaboration time [hh:mm]

# Experimental evaluation - Campaign

Experimented with $n \in \{5,10,15\}$

➔ *number protected sites*

For each *n*, randomly selected three sets *S*

➔ *S* is random subset of the 100 sites in D, #*S* = *n*

For each *S:*

➔ Generate sets *T,V* (..and E)

➔ 3 experiments by rotating *T,V,E* roles

27 experiments

# Experimental evaluation - Results

| $n_S$ | CA | BCA | $\mathrm{FAR}_I$ | $\mathrm{FAR}_U$ | MAR | ET |
|---|---|---|---|---|---|---|
| 5 | 99.0 | 99.2 | 0.0 | 1.0 | 0.7 | 4:49 |
| 10 | 98.0 | 98.2 | 0.1 | 2.1 | 1.7 | 9:22 |
| 15 | 98.4 | 98.6 | 0.3 | 1.6 | 1.1 | 14:17 |

Indexes averaged on 9 experiments

# Concluding remarks

Performed **feasibility study**

- ➔ Phishing Detector **close to the browser**
- ➔ **Based** on webpage **visual appearance**
- ➔ Deep Learning methods, viable approach

Results **highly encouraging**

- ➔ Good performance on all indexes

**Adds** complementary **capabilities** to existing tools

**Protecting 5-15** user-selected **sites** may be **useful**

# Next steps..

Classify webpages not exact replica of originals

➔ but **similar enough** to fool user

➔ to create **dataset** of real **phishing webpages**

Resilience to Adversarial attacks (?)

➔ Webpages forged to fool the classifier

# Questions