

Introduction

ooo

Methods

o
o
o

Evaluation

oo
o
oooo

Conclusions

o

Discussion

oo

Final presentation master lab

A visual artifact detector for ITS.APE

Felix Rossmann

2nd of April 2019

Introduction

ooo

Methods

o
o
o

Evaluation

oo
o
oooo

Conclusions

o

Discussion

oo

Structure

Introduction

Methods

Design decisions

Technical overview

Evaluation

Results: Detection rate

Results: Used resources

Conclusions

Discussion

What's ITS.APE and why does it need a detector?

ITS.APE: IT-Security Awareness Penetration Testing Environment

- ▶ Framework for human pentesting [Syk15]

What's ITS.APE and why does it need a detector?

ITS.APE: IT-Security Awareness Penetration Testing Environment

- ▶ Framework for human pentesting [Syk15]
- ▶ Deploys *artifacts* of a *type* based on *recipes*
- ▶ Measures the response to those artifacts

What's ITS.APE and why does it need a detector?

ITS.APE: IT-Security Awareness Penetration Testing Environment

- ▶ Framework for human pentesting [Syk15]
- ▶ Deploys *artifacts* of a *type* based on *recipes*
- ▶ Measures the response to those artifacts

When to start measuring? ⇒ Visibility in user's desktop

Artifact's visual cues – examples

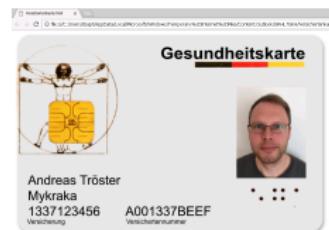
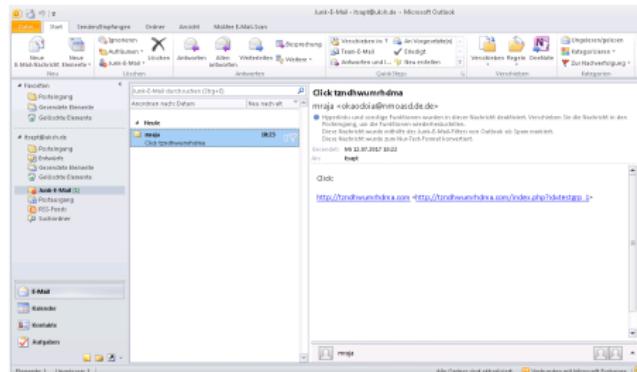
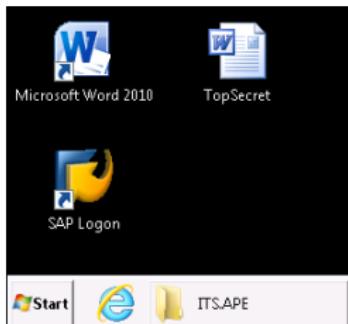


Figure: Examples for artifacts' reference images

A new tool: The VAD

VAD: Visual artifact detector

Measures of quality:

1. High rate of successful detection (*sensitivity, detection rate*)
2. Low resource consumption (esp. runtime)

A new tool: The VAD

VAD: Visual artifact detector

Measures of quality:

1. High rate of successful detection (*sensitivity, detection rate*)
2. Low resource consumption (esp. runtime)

Target environment: Low-tier office hardware (Windows 7)

⇒ Runtime of at most several seconds

⇒ No CUDA, Windows 7 32 bit console application

Software design

- ▶ Separate tool from ITS.APE client service
- ▶ Same machine as client: User's privacy

Major design decisions

- ▶ Image recognition library ⇒ OpenCV [tea19]
- ▶ Wrapper Emgu CV (3.4.3) to use in C# [Hua18]
- ▶ .NET Framework (4.7.2) [Kri18]
- ▶ Installer setup

How is it implemented?

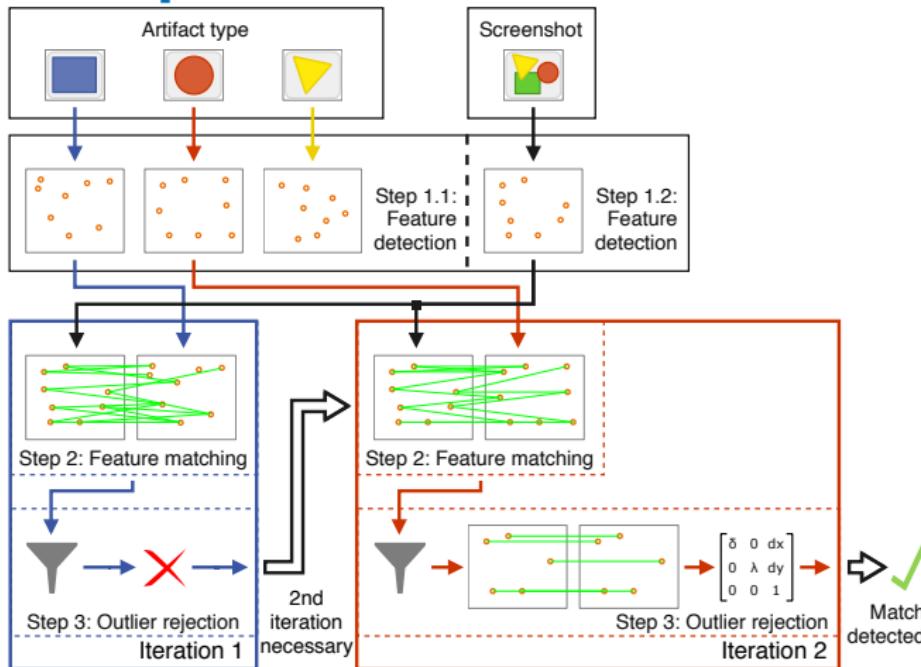


Figure: Exemplary recognition process diagram.

Evaluation of the implementation

- ▶ Virtual machine, Oracle's VirtualBox [Ora19]
- ▶ Virtual CPU with 1.4 GHz
- ▶ 2GB of RAM
- ▶ Windows 7 without any other software

Evaluation of the implementation

- ▶ Virtual machine, Oracle's VirtualBox [Ora19]
- ▶ Virtual CPU with 1.4 GHz
- ▶ 2GB of RAM
- ▶ Windows 7 without any other software

First setup for evaluation:

- ▶ 25 artifact types from recipes repository (A 1 to A 25)
- ▶ 365 screenshots generated (see next slide)
- ▶ 9125 executions of the VAD

Introduction

ooo

Methods

o
o
o

Evaluation

o
o
o
o
o

Conclusions

o

Discussion

oo

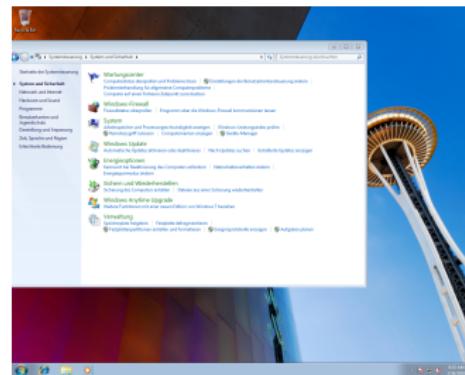
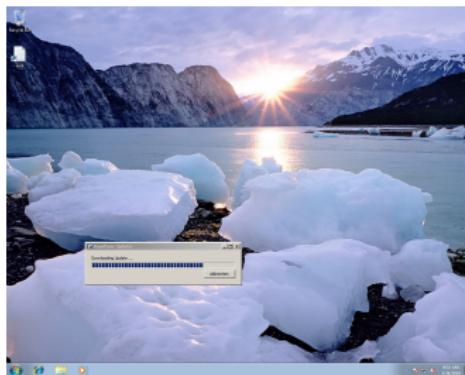
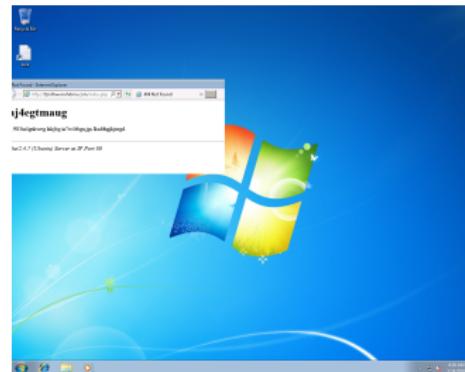


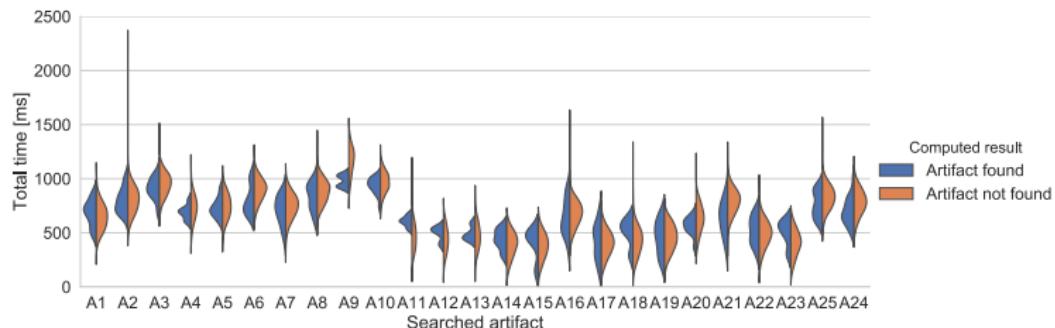
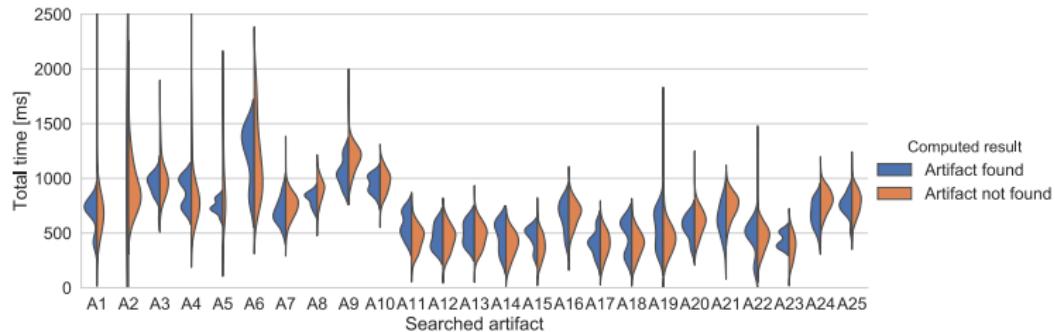
Figure: Examples for screenshots of first evaluation setup.

Results for the detection rate

| | A 1 | A 2 | A 3 | A 4 | A 5 | A 6 | A 7 | A 8 | A 9 | A 10 | A 11 | A 12 | A 13 |
|----|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
| TP | $\frac{10}{10}$ | $\frac{9}{10}$ | $\frac{15}{15}$ | $\frac{7}{10}$ | $\frac{9}{10}$ | $\frac{4}{5}$ | $\frac{20}{20}$ | $\frac{8}{15}$ | $\frac{4}{5}$ | $\frac{11}{15}$ | $\frac{5}{5}$ | $\frac{4}{5}$ | $\frac{4}{5}$ |
| TN | $\frac{355}{355}$ | $\frac{355}{355}$ | $\frac{345}{350}$ | $\frac{355}{355}$ | $\frac{355}{355}$ | $\frac{360}{360}$ | $\frac{345}{345}$ | $\frac{350}{350}$ | $\frac{360}{360}$ | $\frac{345}{350}$ | $\frac{360}{360}$ | $\frac{360}{360}$ | $\frac{360}{360}$ |
| FP | $\frac{0}{355}$ | $\frac{0}{355}$ | $\frac{5}{350}$ | $\frac{0}{355}$ | $\frac{0}{355}$ | $\frac{0}{360}$ | $\frac{0}{345}$ | $\frac{0}{350}$ | $\frac{0}{360}$ | $\frac{5}{350}$ | $\frac{0}{360}$ | $\frac{0}{360}$ | $\frac{0}{360}$ |
| FN | $\frac{0}{10}$ | $\frac{1}{10}$ | $\frac{0}{15}$ | $\frac{3}{10}$ | $\frac{1}{10}$ | $\frac{1}{5}$ | $\frac{0}{20}$ | $\frac{7}{15}$ | $\frac{1}{5}$ | $\frac{4}{15}$ | $\frac{0}{5}$ | $\frac{1}{5}$ | $\frac{1}{5}$ |
| | Artifact | | | | | | | | | | | | |

| | A 14 | A 15 | A 16 | A 17 | A 18 | A 19 | A 20 | A 21 | A 22 | A 23 | A 24 | A 25 | Total |
|----|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|---------------------|
| TP | $\frac{5}{5}$ | $\frac{5}{5}$ | $\frac{20}{20}$ | $\frac{11}{20}$ | $\frac{5}{5}$ | $\frac{18}{20}$ | $\frac{10}{10}$ | $\frac{40}{40}$ | $\frac{10}{10}$ | $\frac{5}{5}$ | $\frac{20}{20}$ | $\frac{14}{15}$ | $\frac{273}{305}$ |
| TN | $\frac{360}{360}$ | $\frac{360}{360}$ | $\frac{345}{345}$ | $\frac{345}{345}$ | $\frac{360}{360}$ | $\frac{345}{345}$ | $\frac{355}{355}$ | $\frac{325}{325}$ | $\frac{355}{355}$ | $\frac{360}{360}$ | $\frac{345}{345}$ | $\frac{350}{350}$ | $\frac{8810}{8820}$ |
| FP | $\frac{0}{360}$ | $\frac{0}{360}$ | $\frac{0}{345}$ | $\frac{0}{345}$ | $\frac{0}{360}$ | $\frac{0}{345}$ | $\frac{0}{355}$ | $\frac{0}{325}$ | $\frac{0}{355}$ | $\frac{0}{360}$ | $\frac{0}{345}$ | $\frac{0}{350}$ | $\frac{10}{8820}$ |
| FN | $\frac{0}{5}$ | $\frac{0}{5}$ | $\frac{0}{20}$ | $\frac{9}{20}$ | $\frac{0}{5}$ | $\frac{2}{20}$ | $\frac{0}{10}$ | $\frac{0}{40}$ | $\frac{0}{10}$ | $\frac{0}{5}$ | $\frac{0}{20}$ | $\frac{1}{15}$ | $\frac{32}{305}$ |
| | Artifact | | | | | | | | | | | | |

Results of used resources



Used Resources: Details? Factors?

First setup: Heterogeneity of artifact types; Too many confounding variables

⇒ Second setup

Used Resources: Details? Factors?

First setup: Heterogeneity of artifact types; Too many confounding variables

⇒ Second setup

- ▶ Eight artifact types Q 0 to Q 7, two reference images r_a, r_b
- ▶ Two different screenshots: Artifact present (s_{ap}) and absent (s_{aa})
- ▶ Each combination 1000 times
- ▶ 16000 executions of the VAD

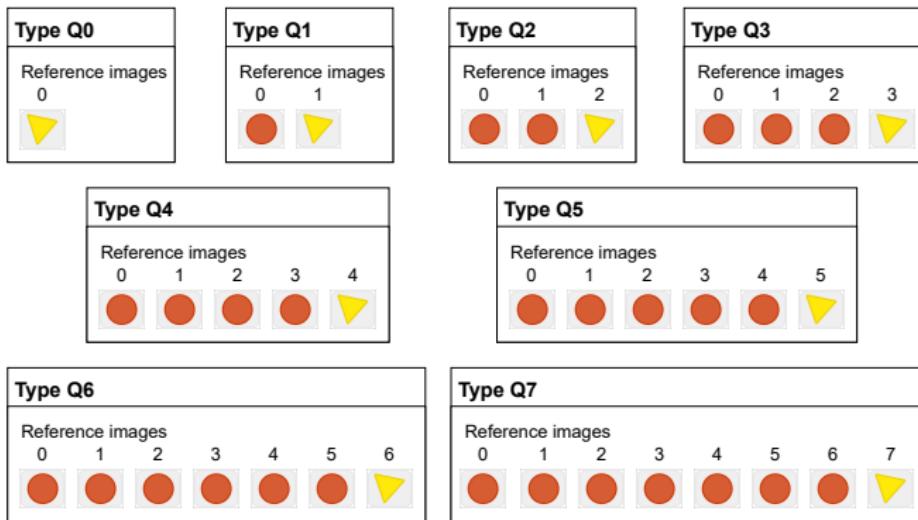


Figure: Structure of the artifact types Q0 to Q7.

Results for setup 2

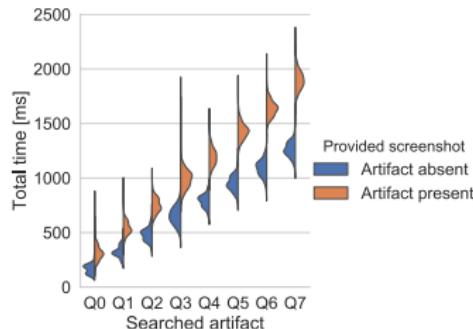
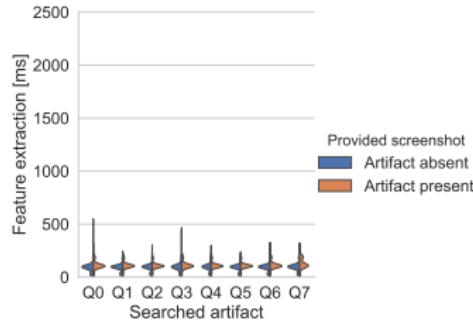
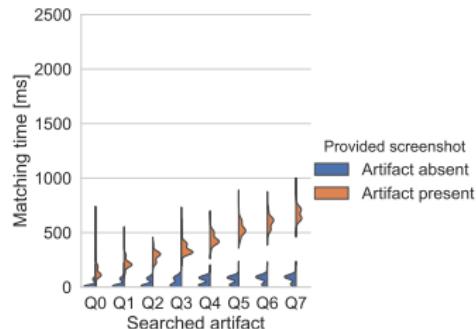
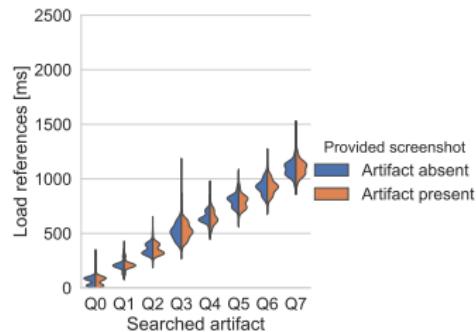


Figure: Execution time distributions for artifacts Q0 to Q7.

Other resources

Measured at random times during all executions.

- ▶ 100% CPU usage
- ▶ About 30 to 45MB RAM usage

Conclusions

- ▶ Successful implementation regarding detection
- ▶ Room for improvement for execution duration, but success for average

Conclusions

- ▶ Successful implementation regarding detection
- ▶ Room for improvement for execution duration, but success for average

Future work could include:

- ▶ Improvement of caching mechanism

Introduction

ooo

Methods

○
○
○

Evaluation

○○
○
○○○○

Conclusions

○

Discussion

●○

Thank you for listening!

Do you have questions, remarks?

Bibliography

 **HUANG, Canming:**

Emgu CV: OpenCV in .NET (C#, VB, C++ and more).

Website, Dezember 2018. –

http://www.emgu.com/wiki/index.php/Emgu_CV

 **KRISHNA, Preeti:**

Announcing the .NET Framework 4.7.2.

Website, April 2018. –

<https://devblogs.microsoft.com/dotnet/announcing-the-net-framework-4-7-2/>

 **ORACLE:**

Oracle VM VirtualBox.

Website, März 2019. –

<https://www.virtualbox.org/>

 **SYKOSCH, Arnold:**

IT-Security Awareness Penetration Testing – ITS.APT.

Website, Juli 2015. –

<https://itsec.cs.uni-bonn.de/itsapt>

 **TEAM, The O.:**

OpenCV library.

Website, März 2019. –

<https://opencv.org/>