Trellix

DotNet Malware Analysis Workshop

Botconf 2024

Max Kersten
Malware Analyst

Table of contents

About me

About the workshop

Virtual safety

Common techniques

The DotNet Framework

Samples

Configuration extraction

Q&A

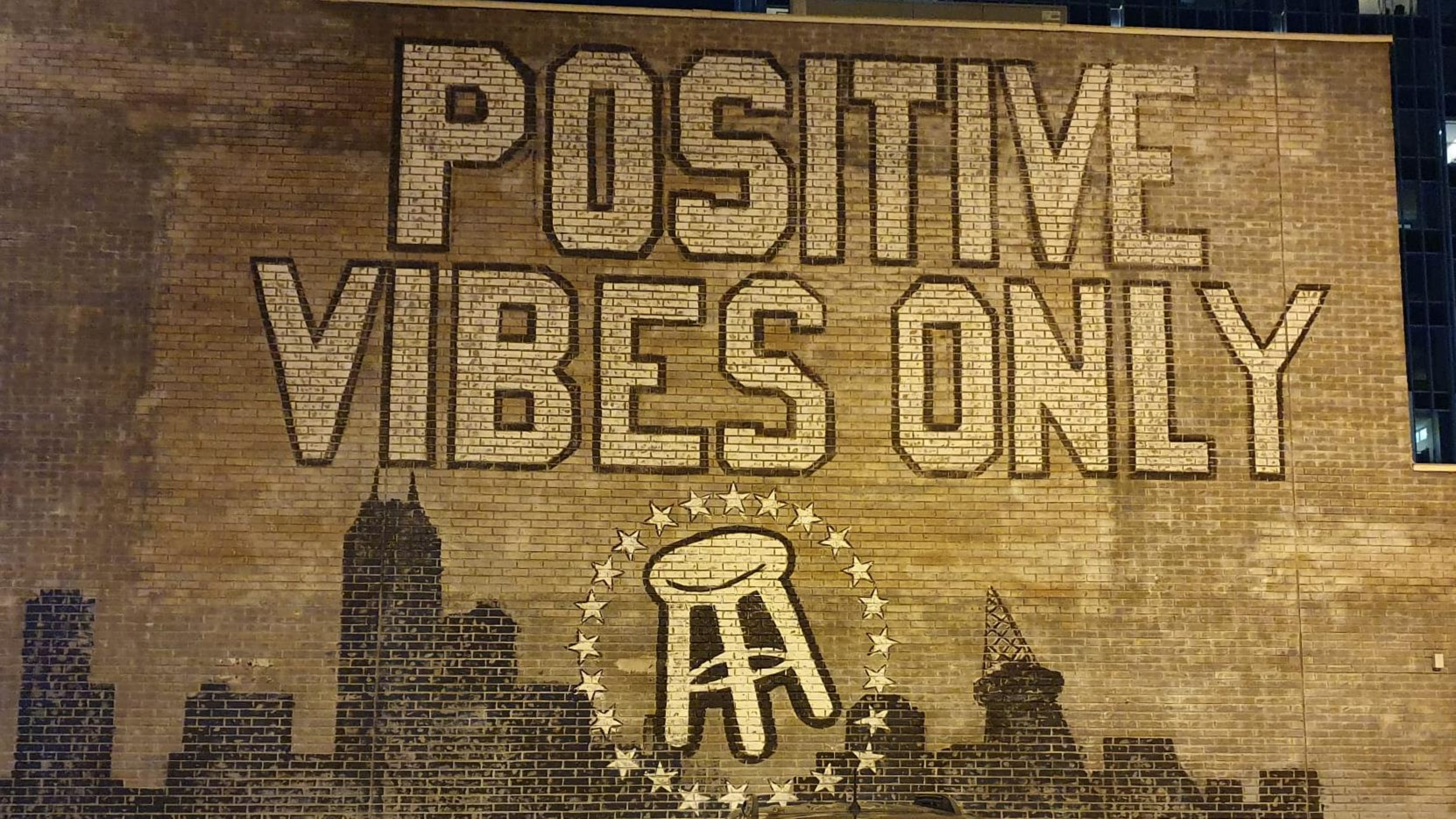
About me

- Max ‘Libra’ Kersten (@Libranalysis, @libra@infosec.exchange)
  - Spoke at numerous conferences (BlackHat USA/EU/MEA/Asia, DEFCON, Botconf, and others)
- Malware analyst and reverse engineer
- Working for Trellix’ Advanced Research Center
  - Published DotDumper
- I write blogs about reverse engineering
  - Including my free Binary Analysis Course
- My tools are open-sourced on GitHub
  - Such as AndroidProjectCreator and the Mobile Malware Mimicking Framework

Trellix

Who are you

A brief introduction round


POSITIVE
VIBES ONLY

About the workshop

Aims to teach concepts	Loaders, reflection, (managed) hooking
DotNet lends itself well for analysis	Current tooling is easy-to-use
Focus on the analyst’s mindset	Avoid needless details

Trellix

Virtual safety

Virtual machines

Snapshots

Old, but not defunct, samples

Common techniques

- Modular and staged malware
- Downloaders
- Persistence
- Process injection
- Obfuscation

Trellix

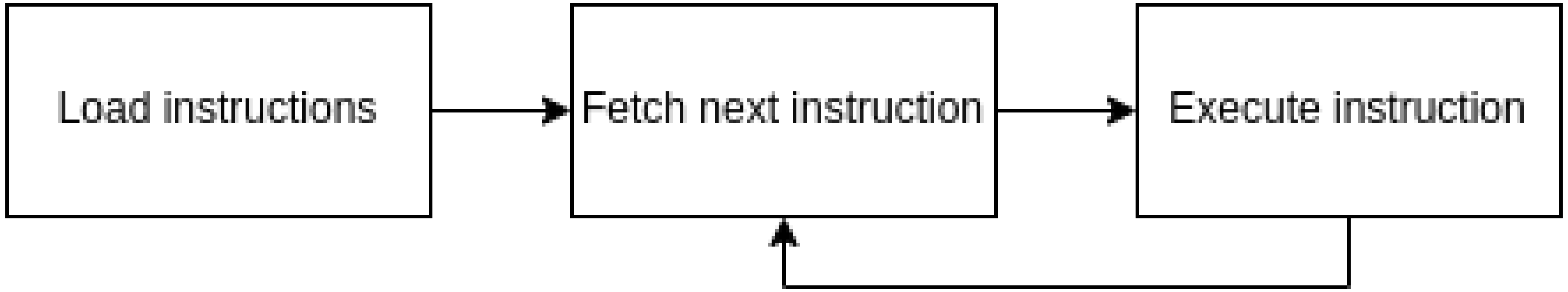
The DotNet Framework

VM-based architecture

Core versus Framework

Just-In-Time compiled

Trellix


Load instructions
Fetch next instruction
Execute instruction

DotNet Internals

What is what?

JIT

Reflection

Types

Trellix

DotNet Internals

- JIT
  - Assembly, IL, and decompiled code
  - VB.NET and C# equally easy to display based on the AST
- Reflection
  - Code to execute code (invoke code)
  - Use code in variables
- Types
  - Preserved in the Intermedia Language, unlike plain ASM
  - Assembly: type versus language
  - Type: a class
  - MethodInfo and MethodBase

The samples

An overview

Wipers

Test the tooling

Understand the analyst's mindset

DotNet RAT

Observe obfuscation and multi-stage techniques

CyaX-Sharp

Determine capabilities

Try to remember as much as possible

Trellix

Wipers

Getting started

Stage 1

Get used to the tooling

How are files wiped?

Stage 2

What are the prerequisites prior to wiping files?

Stage 3

What overlap can you find between the samples?

Trellix

DotNet RAT

What to focus on

Stage 1

How is the next stage loaded?

What obfuscation techniques do you encounter?

Stage 2

What checks are performed?

What benign software is downloaded?

Stage 3

What is the RAT's family?

How is the malware persisted?

Trellix

CyaX-Sharp

What to do

Understanding the structure

How is the loader designed?

What impedes the analysis?

Situational awareness

What is the loader checking for?

Why are such checks in place?

Invoking the next stage

How does it work?

Can you spot a pattern?

Trellix

Configuration extraction

Gather data

- Insight into trends and TTPs

Avoid manual work

- Avoid mistakes
- Mundane

Two types

- Family specific
- Generic

Trellix

Family specific automation

Reflection

Extraction logic

Static, dynamic, or combined

Depends on the scale

Trellix

CyaX-Sharp unpacking

What to do

Find the loader's payload and configuration

Extract the payload

Parse the configuration

Trellix

Generic unpacking

Hooking

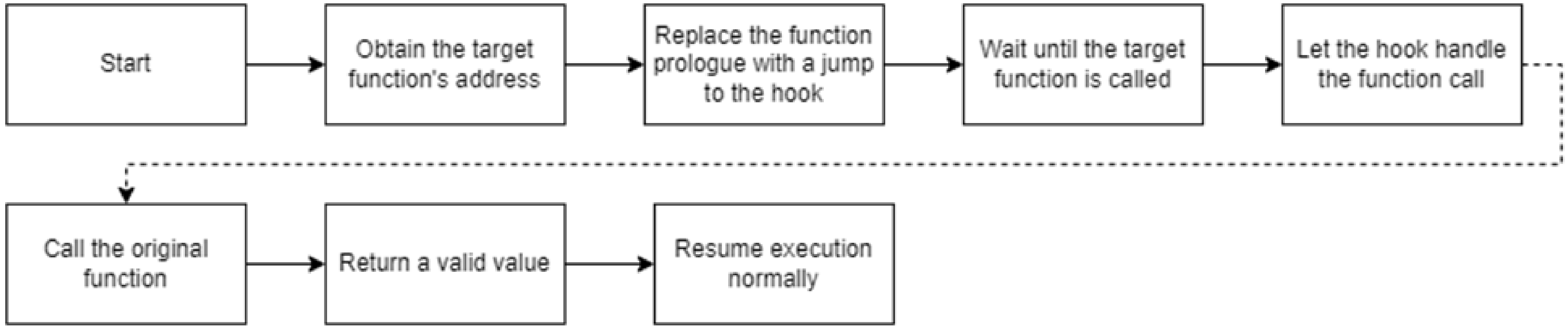
Managed
Unmanaged

Avoid recursion

Static, dynamic, or combined

Depends on the scale

Trellix


Start
Obtain the target function's address
Replace the function prologue with a jump to the hook
Wait until the target function is called
Let the hook handle the function call
Call the original function
Return a valid value
Resume execution normally

Generic unpacking

A step-by-step guide

Find a suitable target function

What is commonly used?

Are there any roadblocks?

Plan your hook

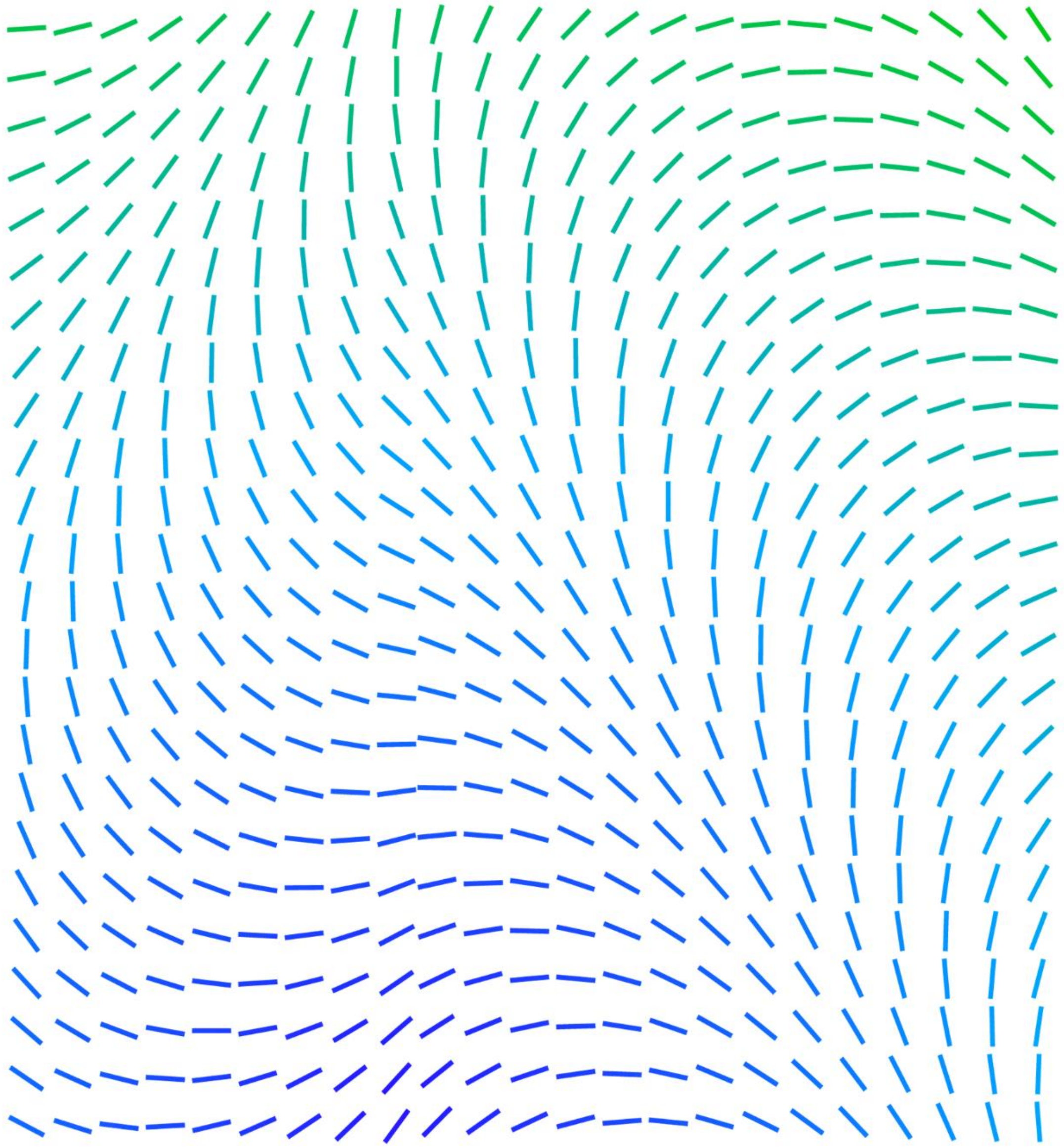
What do you need?

How will perform your actions?

Test your method

Start with a small proof-of-concept version

Trellix



Q&A

For questions, you can also reach out to me via @Libranalysis, @libra@infosec.exchange, or Max Kersten on LinkedIn