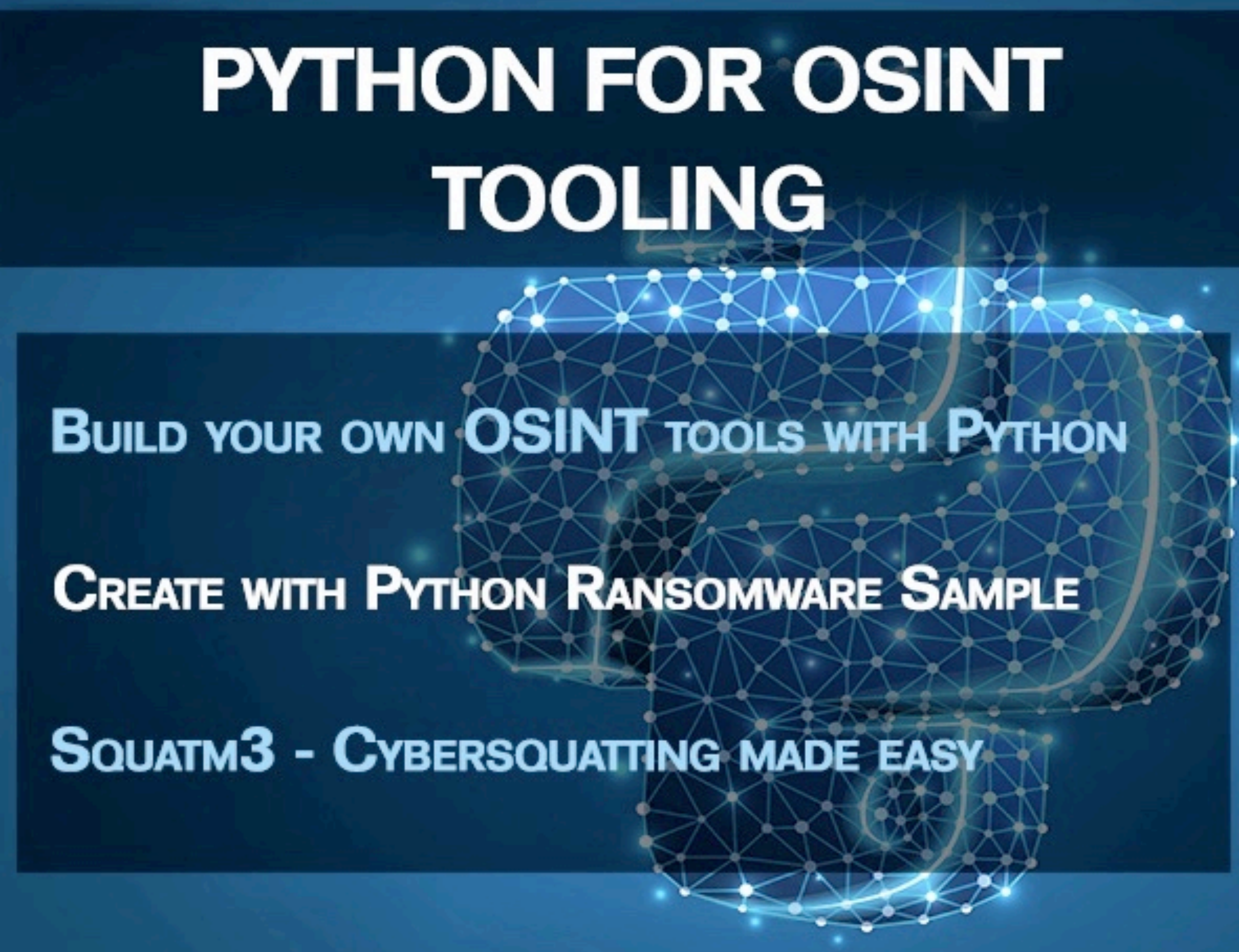
Hakin9

PRACTICAL PROTECTION

IT SECURITY MAGAZINE

VOL.15, NO. 04

PYTHON FOR OSINT TOOLING



Build your own OSINT tools with Python

Create with Python Ransomware Sample

Squatm3 - Cybersquatting made easy

AND MORE...

Hakin9

TEAM

Editor-in-Chief
Joanna Kretowicz
joanna.kretowicz@eforensicsmag.com

Editors:

Marta Sienicka
sienicka.marta@hakin9.com

Dominika Zdrodowska
dominika.zdrodowska@eforensicsmag.com

Marta Strzelec
marta.strzelec@eforensicsmag.com

Bartek Adach
bartek.adach@pentestmag.com

Proofreader:
Lee McKenzie

Senior Consultant/Publisher:
Paweł Marciniak

CEO:
Joanna Kretowicz
joanna.kretowicz@eforensicsmag.com

Marketing Director:
Joanna Kretowicz
joanna.kretowicz@eforensicsmag.com

DTP
Marta Sienicka
sienicka.marta@hakin9.com

Cover Design
Hiep Nguyen Duc
Joanna Kretowicz

Betatesters & Proofreaders

Lee McKenzie

Hammad Arshed

Ali Abdollahi

Robert Fling

Paul Mellen

Bernhard Waldecker

Avi Benchimol

Amit Chugh

Kevin Goosie

Publisher
Hakin9 Media Sp. z o.o.
02-676 Warszawa
ul. Bielawska 6/19
Phone: 1 917 338 3631

www.hakin9.org

All trademarks, trade names, or logos mentioned or used are the property of their respective owners.

The techniques described in our articles may only be used in private, local networks. The editors hold no responsibility for misuse of the presented techniques or consequent data loss.

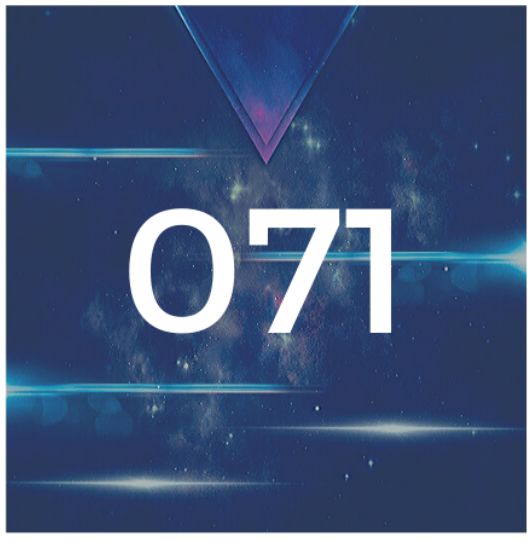
Contents

Build your own OSINT tools with PYTHON

by Felix Castan

OWASP Maryam - Open-source Intelligence (OSINT) Framework

by Saeed Dehqan



Importance of Reconnaissance - FinalRecon tool

by Lohitya Pushkar (thewhiteh4t)

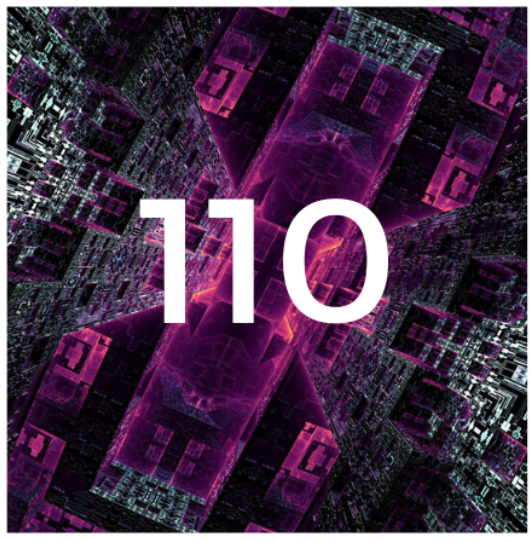
Creating a Simple Ransomware

by Pedro Duarte

Contents

Data Representation Conversion

by Rafael J. Lara L. a.k.a 0xcafecafe



Squatm3 - Cybersquatting made easy

by Davide Cioccia
Stefan Petrushevski

EISEN: A Python Package For Deep Learning

by Ozan Oktay

Ethical hacking possibilities in Kali Linux environment

by Petar Cisara
Robert Pinter

Contents

A Code Injection Method for Rapid Docker Image Building

by Yujing Wang
Qinyang Bao

Ransomware threat and its impact on SCADA

by Anzor Lors
Arvind Kumar

Build your own OSINT tools with PYTHON

Felix Castan

ABOUT THE AUTHOR

Felix Castan

Working as a systems engineer since 1995, specialized in the implementation and management of high-end Unlx systems (IBM, Sun / Oracle), storage systems, backup solutions and databases. The last 10 years, my activity has been oriented to the design and implementation of disaster recovery solutions.

Study computer engineering at the 'Universidad Politécnica de Madrid' (1988-1994) and master in cybersecurity from the Camilo José Cela University (2017-2019).

Ethical hacking certified, IBM AIX certified and EMC storage implementation certified.

Email: felixcastancid@gmail.com

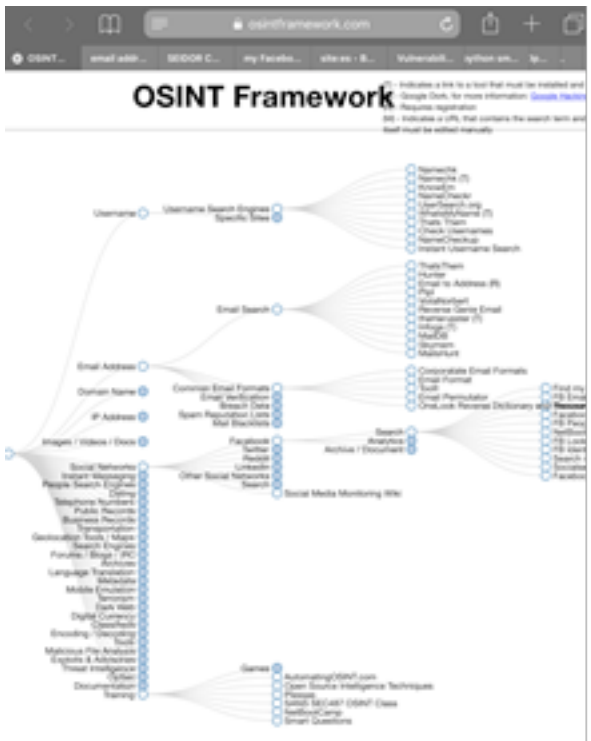
Linkedin: https://es.linkedin.com/in/félix-castán-a3490730

Open Source Intelligence (OSINT) is data collected from publicly available sources to be used in an intelligence context. In the intelligence community, the term "open" refers to publicly available sources.

Nowadays, the use of information technologies has become usual for everyone. People and companies 'need' to be continuously on-line and it is increasingly common to do all kinds of online transactions using our mobile devices or our computers; we reserve vacations, we make purchases, we inform ourselves, procedures with the government or financial entities are completed online and we even carry our social life through applications on the network. As a consequence of this, there is more and more information regarding us or our companies on the internet.

In this way, in a world in which we depend more and more on telematic services to carry out day-to-day tasks, it is essential to be able to operate with them safely. Since cyberspace has become a field of operations for all types of criminals, it is necessary to increase security measures in the field of information technologies more and more, and as technical means of protection become extreme, attacks on our systems are increasingly complex and are increasingly supported in 'social engineering' since the security devices implanted in the different corporations are increasingly difficult to cross and the most effective way to sneak into the target infrastructure of our 'attack' are the users, who are the weakest link in the security chain of our target. For this reason, as professionals dedicated to cybersecurity to carry out the work of 'pentesting' properly, it is necessary to obtain as much information as possible about the target, in particular, everything related to the people who are part of or are related in some way (employees, suppliers, partners...) with the organization against which we are going to direct our attack in order to design attack strategies that allow us to infiltrate the network of the organization that is the target of our pentest.

For this, there are many and very effective automatic tools to acquire open source information (OSINT) of which, among many others, we can highlight recon-ng, Maltego, Theharvester ..., however, in my experience, these tools have been very effective in collecting information regarding domains, web servers, mail servers, IOT devices, etc..., but I have not achieved great results in the task of collecting information regarding people. In my opinion, this is due to several causes; on the one hand, since it is unstructured information that is found in different places with different formats, it is difficult to have a standard tool for these tasks, on the other hand, I develop my professional work in Spain and I believe that these tools are made and tested in other countries, so they are fed with information in places where there is little information about entities in my country. Finally, for my work, I am limiting myself to free tools, and I understand that there are proprietary tools that make use of the official APIs of different sources of information, especially social networks, and allow them to obtain a greater amount of information. For all this that has been exposed, and last, also because deep down I love to program, I think it is important to be able to develop your own tools.



Methodology

When developing a tool to obtain information from open sources, we will follow the next steps:

1. **Purpose:** Before starting, it is essential to know what information we want to obtain and for what purpose, so that we limit the amount of information obtained and why it is useful for our purposes. For example, if we want to make a social engineering attack by sending massive emails, it should be enough for us to obtain as many email addresses as possible. However, if we want to make a more specific attack, we must seek information about each individual to know their hobbies, work and family situation, concerns, etc., which makes it easier for us to reach them and to be 'fished'.

2. **Source Analysis**: The Internet is an immense ocean of information, so to make our search for information feasible we must limit the sources in which we are going to look for this type of information. Once the sources are determined, since it is not information structured, we have to see how this information is presented in order to define the appropriate extraction algorithm. This is perhaps the most important and complex part of the whole process.

3. **Coding**: It is the easiest and the one we like the most. It consists of defining the algorithm and writing the program in Python.

4. **Test**: You have to test the application in order to check its correctness and effectiveness with test cases that cover different situations and different environments.

In this article, we are going to see two different ways of approaching data collection; on the one hand, we are going to try to extract it from the website of the target entity and on the other hand, we will do it using search engines like Google.

Below we will show some examples of tools written in Python to obtain information related to people, for this, as we have already mentioned, we will follow two different strategies when looking for information. On the one hand, we will perform a recursive search of the web links from a specific domain (for example, our target's website) and on the other hand, we will rely on search engines to do searches related to our target and analyze the pages found .

This information search can be directed, which previously requires a more or less exhaustive study of the website to be explored to obtain details of how the information we are going to search for is structured (for example, the search for people on websites where their employees publish, partners, sympathizers ...) and on the other hand, we can carry out the search by brute force by exploring the entire web to see if we find the information we require (phones, emails, social networks ...)

Python is a widely used programming language among cybersecurity professionals and/or amateurs. In fact, many of the existing tools that are included in distributions such as 'Kali Linux' are written in Python. There are two main versions of Python, Python2 (more specifically, Python2.7) and Python3, whose last stable version is 3.8.2. Both are

OWASP Maryam - Open-source Intelligence (OSINT) Framework

Saeed Dehqan

ABOUT THE AUTHOR

Saeed Dehqan

Top performing security research with strong proficiency in finding security bugs, exploiting it and how to solve them. Perform security projects working closely with the hacking tools industry, including information gathering, scanning and exploitation. Self-motivated, with education and diverse experience in security problems-solving.

Open-source intelligence (OSINT) is a method of using open source tools to collect information and analyze them for a specific purpose. OSINT can be very helpful for hackers to use to garner data about particular organizations. Today, using open-sources like Bing, Google, Yahoo, etc., to gather data is one of the important steps for reconnaissance and this is a common task. It should be a tool to automate this routine. One of the best tools in this field is OWASP Maryam.

INTRODUCTION

OWASP Maryam is a modular/optional open source framework based on OSINT and data gathering. Maryam is written in Python programming language and it's designed to provide a powerful environment to harvest data from open sources and search engines and collect data quickly and thoroughly. If you have skill in Metasploit or Recon-ng, you can easily use it without prerequisites, and if not, it's easy to use.

GETTING STARTED

For installing, there's no need for a lot of modules to install, It just needs a "requests" library. Maryam uses Regex to crawl web pages. Using Regex speeds up the program. So users don't need to install modules like bs4 or lxml.

Note: Currently, this tool is optimized for Linux and Darwin. So it may not work for operating systems such as Windows.

Prerequisites

Maryam requires Python 3.6+ and for package installation, it also uses Python package manager PiPl (pip).

Requires

- requests

Install

The repository can be loaded using the following command:

```
git clone https://github.com/saeeddhqan/maryam.git
```

Next step is install the requirements:

```
pip3 install -r requirements
```

The installation is finished and you can run with:

Importance of Reconnaissance - FinalRecon tool

Lohitya Pushkar
(thewhiteh4t)

ABOUT THE AUTHOR

Lohitya Pushkar

Cyber Security Researcher & Analyst

I have created multiple open source tools for the infosec community and I am learning about Red Teaming, Network Penetration Testing, Exploit Analysis and OSINT.

Blog : https://thewhiteh4t.github.io

GitHub : https://github.com/thewhiteh4t

Twitter : https://twitter.com/thewhiteh4t

LinkedIn : https://www.linkedin.com/in/lohityapushkar

Importance of Reconnaissance

Reconnaissance is a process of obtaining information by observation and other detection methods about a certain target. From an attackers point of view, it is an important step to steal confidential information and that is why it is important in penetration testing.

A proper recon provides us with detailed information like open ports, active services, specification of servers, routing, important files, hidden links, etc. When performing a penetration test, it is crucial to begin with reconnaissance; it is possible that the information collected during the recon phase plays a critical role during exploitation as well as the post-exploitation phase of the test.

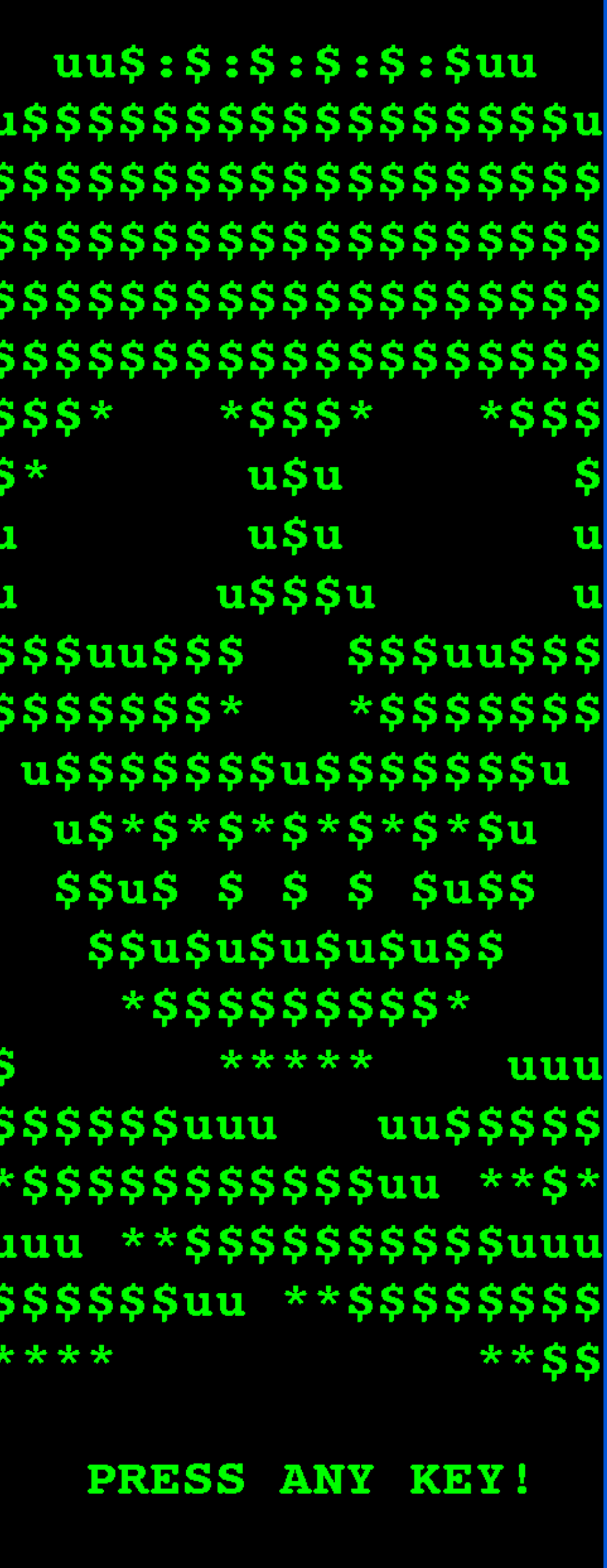
What is FinalRecon?

FinalRecon is a fast and automated web reconnaissance tool written in Python. During a penetration test we need to execute a set of tools in the reconnaissance phase along with some manual efforts. The goal was to remove the need of multiple tools in favour of one single tool that is automatic and, most importantly, fast.

Over the years, we have used various online services to get data like header, whois, dns information, etc., and tools like nslookup, dirb, etc. While all these tools and services are good, there are many and they all have different commands and features. FinalRecon, on the other hand, contains a long and growing list of features; it does not use already available tools. Instead, each feature is written in Python and optimized for both speed and accuracy. Along with this, the structure of the tool is modular, so adding new features is simple.

Availability

At the time of writing this article, FinalRecon is available on two amazing Linux distributions: **BlackArch Linux** and **Tsurugi Linux.**



Creating a Simple Ransomware

Pedro Duarte

ABOUT THE AUTHOR

Pedro Duarte

My name is Pedro Duarte, I'm from Portugal and currently on my first year of Computer Engineering in Aveiro University.

I've always been curious about the world of IT security but I do mostly freelance web development, from web sites to web crawling.

I don't have a LinkedIn yet but you can find me on dev: https://dev.to/pedroduarte536

Disclaimer:

This paper is meant for educational purposes only, I do not participate in, or encourage participating in, any criminal activity. Understanding how someone with bad intentions would attack you is the best way to be safe, to protect and defend yourself and others, and that's what you should use this for.

Introduction

A ransomware, by definition, is a software that in some way blocks access to a machine or part of it, usually asking for money to fix it. Typically, this means encrypting files inside the victim's computer, often stealing those and blackmailing the owner stating that they will make them public if they don't give in to their conditions. The targets are usually companies and not individuals, as their computers store sensitive data that can represent huge losses if lost or made available to the public.

During this tutorial we will be creating a simple ransomware with Python. It will be divided into two scripts - the server (that should be run in the attacker's machine) and the client (that should be run in the victim's machine).

Getting Started

During this tutorial, Python 3.7.4 wa used. It is probably going to work fine with any Python 3 version but if you are using Python 2, you will need to adapt the code to work on your computer.

You will also need the following Python modules: os, socket and cryptography (os should already be installed by default).

By the end of this tutorial, we should have created functional server and client scripts, with functionality as described next:

<u>server</u>

- let the client connect to it;
- receive and send decryption keys.

<u>client</u>

- encrypt files in the folder where it is and folders inside that;
- connect to the server;
- send the key to decrypt to the server;

Data Representation Conversion

Rafael J. Lara L. a.k.a 0xcafecafe

ABOUT THE AUTHOR

Rafael J. Lara L.

I'm from Caracas, Venezuela, but living in Panama. I got a bachelor's degree in Computer Science at Universidad Central de Venezuela, 2003 and a master's degree in Information Security at Universitat Oberta de Catalunya, 2007. I have over 30 years in the IT sector and over 8 years in cybersecurity. Actually, I work as penetration tester in Cybersec Asesores and InteliCorp Seguridad for a broad range of companies in banking, brokerage, media, elections and financial sectors.

You can reach me anytime at:

- Email: rafael@rafademia.com
- Twitter: @rjlaral1
- LinkedIn: https://www.linkedin.com/in/rjlaral

Introduction

In Penetration Testing, it's very common to choose Python as scripting language to perform some automation task, create Proof of Concepts, exploits and so on. It's well-known that Python is a versatile and easy-to-use language for this purpose.

In some cases, we could face situations when data types and its representation are a key matter to accomplish a successful activity, by example, when interacting with a web service facility, an API service, performing a packet inspection or packet analysis, manipulating cryptography material (as plaintext/ciphertext representation and conversion, key manipulation and so on), do a reverse engineering or malware analysis, or making an exploit. In fact, in several scenarios.

In RSA, it's a well-accepted practice to transform plaintext input (as ASCII or UTF-8 string) to hexadecimal representation (stored in a 'latin1' encoded string or sequence of single bytes) and then in decimal representation before encryption; and transform encryption output from decimal number (integer) to hexadecimal representation and then in 'latin1' encoded string to output ciphertext as expected. A similar process is applied to recover plaintext from ciphertext.

In packet analysis/inspection, reverse engineering and malware analysis, some (numeric) values are represented in hexadecimal big-endian or little-endian, how can we convert it?

Recently, I faced a situation where I had to intercept and capture an HTTP communication dialog. The data captured was represented as "\xab\xcd\xdf", that is: character "\" followed by character "x" followed by character "a" and so; but I needed Python to treat this data as a byte sequence representation, where '\xab' meaning byte conformed by hexadecimal 'a' followed by hexadecimal 'b', how could we convert it in that way?

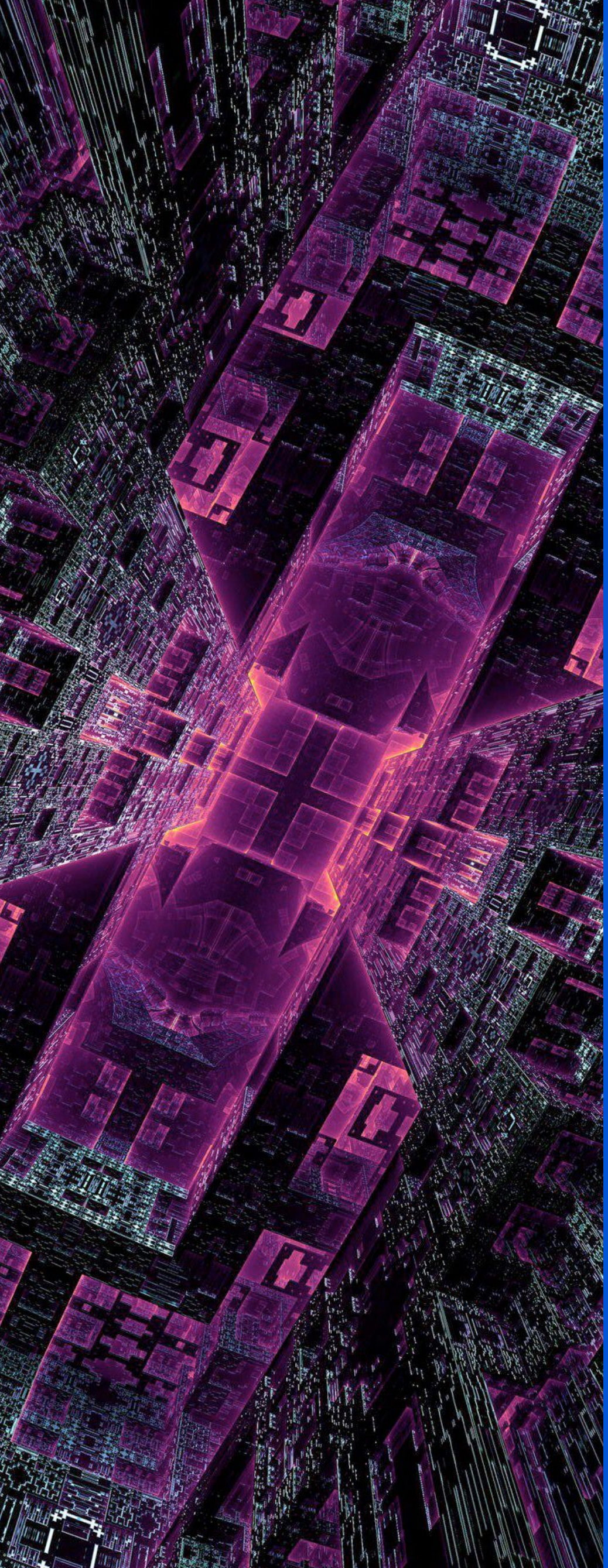
It would be helpful to have a guideline, a snip list or cheat sheet that shows us how to do such conversions.

This article assumes the reader has previous knowledge of Python and basic concepts of computer science (like endianness; binary, hexadecimal and decimal numeric representation). Also, all examples are based in Python 3. I love the "one-line solution" Python philosophy, so I tried to do all conversions in one-line code fashion.

Let's get our hands dirty!

Conversions

I don't pretend to show you an exhaustive list for all kinds of representation conversions we could do in Python; instead I encourage readers to do further research in this topic and adapt these features to his/her own needs.



Squatm3 - Cybersquatting made easy

Davide Cioccia

Stefan Petrushevski

ABOUT THE AUTHOR

Davide Cioccia

He is the founder of DCODX.com, a cyber security consulting firm providing security consultancy services, security trainings and security solutions to fight cybercrime. Security Engineer at ING and Speaker and trainer at multiple security conferences worldwide: like BlackHat EU and Asia, OWASP AppSec USA and Romania, DevSecCon Boston, DevSecCon Seattle.

Website: https://dcodx.com

Email: d.cioccia@dcodx.com

LinkedIn:

https://www.linkedin.com/in/davidecioccia/

https://www.linkedin.com/company/dcodx/

ABOUT THE AUTHOR

Stefan Petrushevski

He is a Cyber Security specialist with vast experience in security evaluation, security research, and building of secure products, in multiple verticals. He is a core contributor to the Zero Science Lab (www.zeroscience.mk) project since 2008 where he is involved in security research, penetration testing, security evaluation as well as consulting and training services.

At the moment, he also occupies a Security Chapter Lead role at ING Bank leading a team that supports hundreds of DevOps teams to deliver secure software.

Website https://www.zeroscience.mk

Email: stefan@zeroscience.mk

LinkedIn: https://linkedin.com/in/stefanpetrushevski

Twitter/Github: @ztefan

As a penetration tester, have you ever wondered what might be the best domain to buy for your social engineering/phishing campaign? As an SoC security analyst, have you ever looked for possible similar domains that could lead to a massive data breach?

In this article, we introduce Squatm3 and Squatm3gator, two tools written in Python 3, developed for security analysts and penetration testers, to detect and perform cybersquatting attacks before it is too late. Individuals or organizations can use Squatm3 or Squatm3gator to identify squatted domains, generated using one or multiple of the following techniques:

- **Homoglyph attacks**

  Homoglyphs are characters with shapes that are nearly identical or similar to each other

- **Substitution attacks**

  Substituting every letter with the respective number a=4, t=7 etc

- **Flipping attacks**

  Flipping two letters next to each other (for example, yahoo.com and yaoho.com)

- **Removal attacks**

  Removing one letter per time (for example, yahoo.com and yaho.com)

- **TLD substitution**

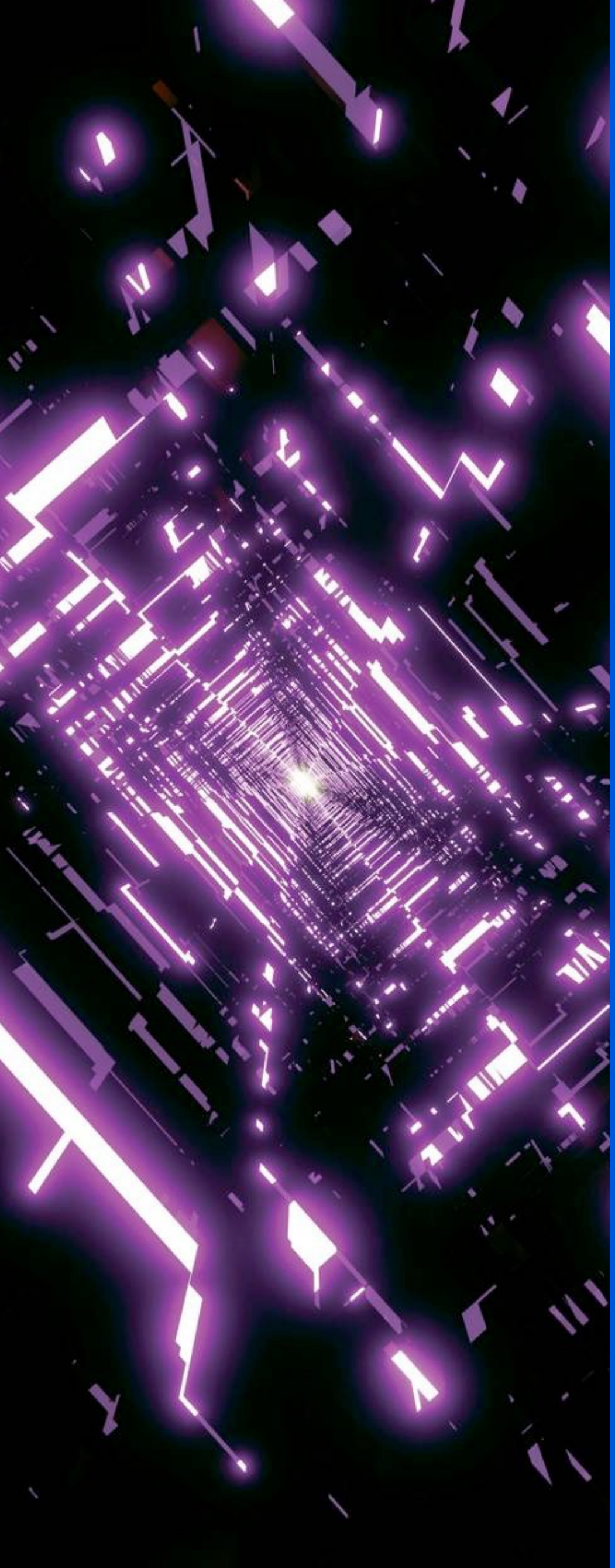
  The original TLD is substituted with a list of common TLD (for example, .com, .eu, .org)

But what exactly is cybersquatting? Before we go into the details of the tools and their implementation, let's have a quick look on how this phenomenon is born.

Cybersquatting quick recap

The word cybersquatting is an extension of the legal definition of "squatter," which apparently was first used in 1788 to describe "one that settles on property without right or title or payment of rent."[1]

Nowadays, cyber-domains are becoming more and more tight to the brand name of a company. Many fintech and small companies that focus their business online are using domains as branding material. This was not the case in the 90s when many companies did not yet understand the importance of having a presence online and a registered trademark that will be used in the next decades from people around the globe. This situation has favored the advent of cybersquatting in its simplest form: "*action of attempting to profit by purchasing domain name, and later reselling or licensing those names back to the companies that developed the trademark*"[2]. But cybersquatting became more



EISEN: A Python Package For Deep Learning

Ozan Oktay

ABOUT THE AUTHOR

Ozan Oktay

Ozan Oktay is a Research Associate in Computing Department at Imperial College London (ICL), where he is working with Prof. Daniel Rueckert. His research focuses on developing novel machine learning methodologies for medical image analysis, in particular for image super-resolution, semantic image segmentation, and object localisation. Besides his employment at ICL, he is leading the core development team at ThinkSono Ltd as a machine learning advisor. Prior to joining the Biomedia Group at ICL, he completed his Bsc and Msc studies at Ecole Polytechnique Federale de Lausanne, and worked in Siemens Corporate Research (NJ, USA) and ABB (Baden, CH) for two years as a researcher.

Eisen is an open source Python package making the implementation of deep learning methods easy. It is specifically tailored to medical image analysis and computer vision tasks, but its flexibility allows extension to any application. Eisen is based on PyTorch and it follows the same architecture of other packages belonging to the PyTorch ecosystem. This simplifies its use and allows it to be compatible with modules provided by other packages. Eisen implements multiple dataset loading methods, I/O for various data formats, data manipulation and transformation, full implementation of training, validation and test loops, implementation of losses and network architectures, automatic export of training artifacts, summaries and logs, visual experiment building, command line interface and more. Furthermore, it is open to user contributions by the community. Documentation, examples and code can be downloaded from http://eisen.ai.

1. Introduction

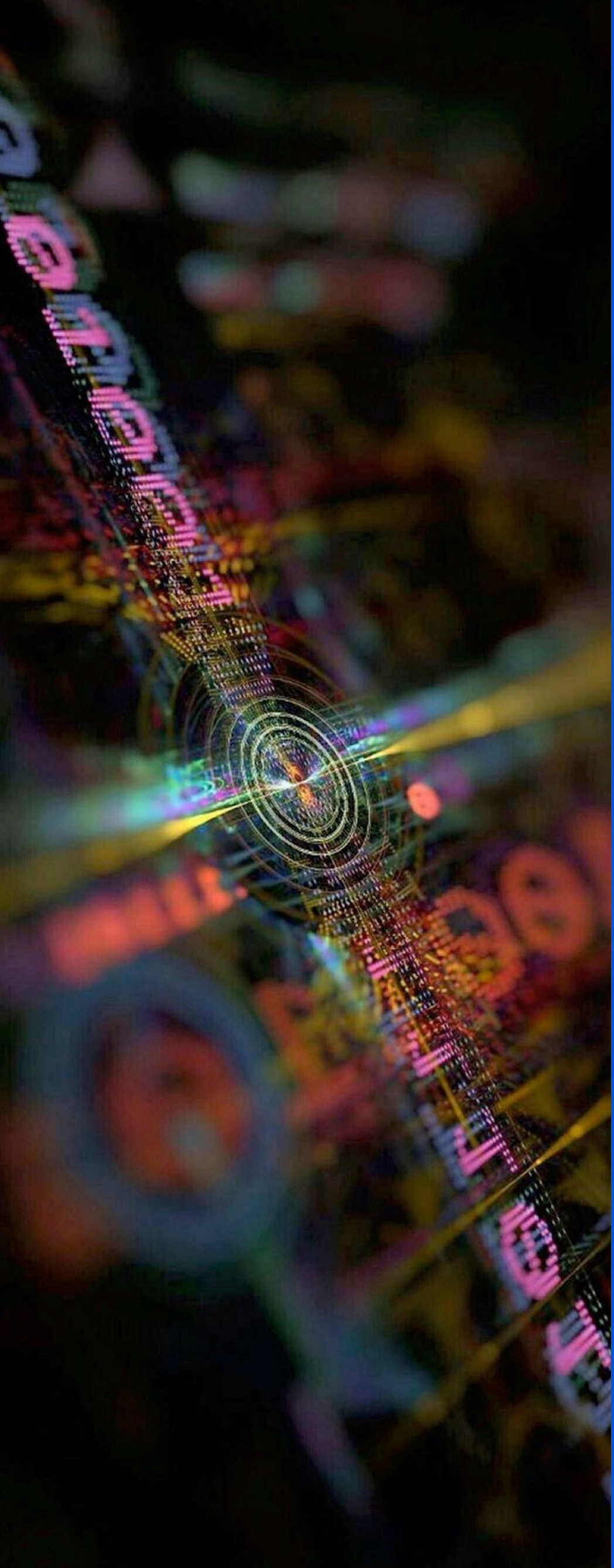
Deep learning methods have recently emerged as extremely effective tools to solve challenging tasks in a wide range of domains. Computer vision has been revolutionized by the introduction of deep learning approaches. Medical image analysis, as a branch of computer vision, has not been an exception: the vast majority of recently published papers in this domain is based on deep learning and the use of convolutional neural networks (CNNs).

Most open source implementations of recent methods are built from scratch. Almost every published work leverages custom made data processing routines as well as custom training, validation and testing loops and often relies on personalized model implementations. This makes development of new approaches a labor intensive procedure. Furthermore, whenever an existing approach is applied to a new dataset or whenever it is necessary to change part of an existing implementation to accommodate new functionality, multiple changes need to be made to the code base and, depending on its architecture, silent bugs and unnecessary complexity might appear.

Performance benchmarks, which are crucial to reveal the true capabilities of newly proposed approaches, are also not easily obtainable and often misleading, as researchers have often to rely on their own implementations of, at least, part of the comparing method implementation. This is often error prone since this code is often untested and some implementation detail can be missed by even expert programmers.

Finally, whenever research changes hands and project owners with different backgrounds and skills follow one another, these custom architectures need to be re-evaluated, understood and hopefully expanded in order to incorporate new work. Convoluted custom designs represent a big disadvantage in this scenario as creating new compatible code and reusing existing modules is challenging and often impossible.

These are just some of the issues motivating the development of Eisen. Our goal is to provide an "opinionated" framework for the development of deep learning approaches for vision tasks. In this sense, Eisen proposes a modular architecture where each module performs a very limited amount of functionality through a standard interface and a very readable implementation. Modules can be mixed and matched, each with its specific role, and used during training, validation, testing as well as model serving in order to realize complex functionality. Each module is easy to read, is documented and follows a standard implementation.



Ethical Hacking Possibilities In Kali Linux Environment

Petar Cisara

Robert Pinter

ABOUT THE AUTHOR

Petar Cisar

Petar earned a PhD degree in Information Systems. He works at the University of Criminalistic and Police Studies in Belgrade as associate professor. The spheres of his interest are information technology, computer and telecommunications networks, network security, soft computing and computation intelligence. He is an IEEE member - Computer Society Technical Committee on Security and Privacy and a member of the International Society for the implementation of fuzzy-theory with its seat in Budapest, as well as an external member of the public body of Hungarian Academy of Sciences and Arts. In addition, he is a member of the Serbian Chamber of Engineers.

ABOUT THE AUTHOR

Robert Pinter

Robert is a professor at Subotica Tech - College of Applied Sciences. He obtained his MSc degree at the Electrical Engineering Faculty at the Budapest University of Technology. He defended his PhD thesis at the Technical Faculty “Mihajlo Pupin” in Zrenjanin, Serbia, in 2012. He teaches various computer science courses in the field of programming languages and mobile application development. The main research area in his scientific works is improving the efficiency of e-learning with the application of novel technologies and new teaching methodologies.

This article deals with the problem of ethical hacking and security of computer systems. When we talk about security of an information system, we actually mean the primary three attributes of the system: confidentiality, integrity and availability. There are various approaches that aim to identify existing security weaknesses and security assessments. One of them is using Kali Linux operating system with its integrated effective tools specially adapted to the realization of various types of attacks. The paper gives a general overview of some Kali attacking possibilities on both client and server side and highlights their specificities. The undoubted benefit of this operating system is a large collection of different hacking tools in one place, which significantly facilitates vulnerability assessment and security testing.

Introduction

In general, four main categories (or phases) of information security assessments can be identified (Hertzog, 2017): a vulnerability assessment, a compliance (audit) test, a traditional internal/external penetration test, and an application assessment. There are various methods that aim to identify existing security weaknesses and security assessments (Allen, 2014). One of them is using tools from Kali Linux operating system (OS).

Kali Linux is a Debian-based Linux distribution focused on advanced penetration testing and ethical hacking. It contains several hundred tools aimed at a wide range of information security tasks, such as penetration testing, security examinations, computer forensics and reverse engineering (Pritchett, 2013). The term hacking refers to identifying and exploiting security weaknesses in computer systems and/or networks.

Tools within Kali package are very diverse and can be divided into the following categories (Kali Linux Tools): Information gathering, Vulnerability analysis, Wireless attacks, Web applications, Exploitation tools, Forensics tools, Stress testing, Sniffing and spoofing, Password attacks, Maintaining attacks, Reverse engineering, Hardware hacking and Reporting tools (Fig. 1).

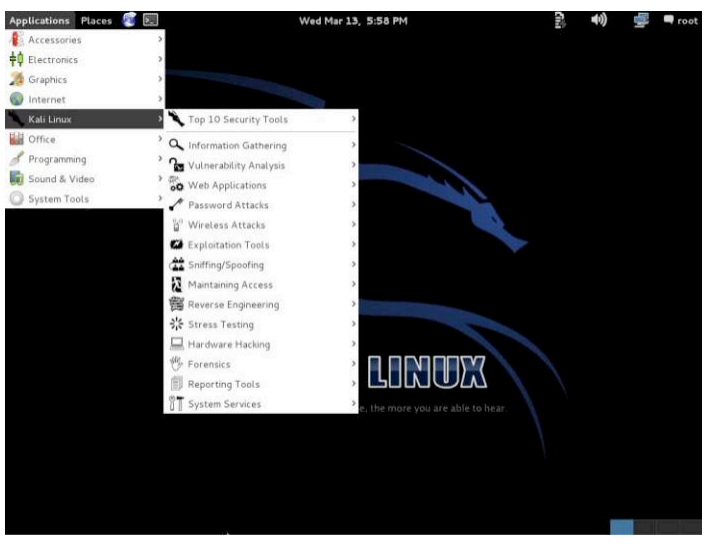


Fig. 1. Kali Linux integrated tools

Kali Linux contains frequently used security testing tools such as: Nmap (port scanner), Wireshark (packet analyzer), John The Ripper (password cracker), Aircrack-ng (software suite for penetration testing wireless LANs), Nikto (web server scanner), Sqlmap (tool for detecting and exploiting SQL injection flaws and taking over of database servers),

A Code Injection Method for Rapid Docker Image Building

Yujing Wang

Qinyang Bao

ABOUT THE AUTHOR

Yujing Wang

Department of Mechatronics Engineering

Waterloo, ON,Canada

ABOUT THE AUTHOR

Qinyang Bao

As a self-motivated and enthusiastic Mechatronics Engineering student at the University of Waterloo, I have always aspired to become a professional engineer and entrepreneur. Either way, I want to help improve the world we live in as well as bring meaning to the lives of others and myself. Working towards this goal, I have acquired 1+ year of experience working with multiple programming languages and CAD software. In my last co-op, I built an IoT data transmission system consists of Raspberry Pi clients and a Flask website.

Docker images are composed of multiple layers, each of which contains a set of instructions and an archive of files. Layers allow Docker to separate a large build task into smaller ones, such that when a part of the program is changed, only the corresponding layer needs to be changed. Yet the current implementation has major inefficiencies that make the rebuilding of an image unnecessarily slow when changes in bottom layers are required: uneven content distribution amongst layers, the need to rebuild an entire layer during update, and the rebuild fall-throughs in many cases. In this article, we propose a code injection method that overcomes these inefficiencies by targeting only the changed layer and then bypassing the layer's content checksum. This process is developed specifically for an interpreted language such as Python, where changes can be detected explicitly via text diff tools and run as-is without compilation. We then demonstrate that this method can accelerate the rebuild time, effectively reducing the O(n) where n = size of layer rebuild time to O(1), whereas for compiled languages, literal code injection cannot guarantee integrity in compiled machine code. Expanding on the same code injection principle, multi-layer targeted code injection will be addressed in a future discussion.

INTRODUCTION

When deploying an application on a single machine can no longer match its expanding usage, developers split it into microservices, and deploy them on multiple machines. They package modules along with its dependencies in an image to guarantee consistency across platforms and deploy these images in containers that are running instances of their images. For system-level processes, developers use a Linux Container (LXC), which uses Linux namespace, a kernel feature that partitions a set of resources for a set of processes exclusively, along with control group (cgroup), a kernel feature that limits and isolates machine resource usage [1]. On an application level, developers use Docker that builds an image from instructions given in a Dockerfile. Docker reads each run line of instruction, which is made up of an ”Instruction” and its ”Arguments”, executes it, and stores results in files in an image layer [2]. Each layer generated will be assigned a permanent UUID in SHA-256; each revision of a layer will be given a checksum in SHA-256. These values are stored in an image's manifest, so Docker knows which image a layer belongs to and which revision it should use. If a developer changes the content of a layer, the layer's ID remains the same, but its checksum varies. Fig. 1 shows Docker building an image layer with id b248b9e23166 from a command ”FROM python:alpine” using cache. Notice that after each build, Docker informs the user of each layer's ID. To examine what command each layer corresponds to, the user can run ”docker history image:tag”. By default, all layers are stored in ”/var/lib/docker/overlay2/”. A developer can export the image by ”docker save image:tag>file.tar” and load it by ”docker load < file.tar”. The layer by layer architecture speeds up the image building process and makes it more flexible. When Docker creates a new image, it first searches in a registry to determine if an exact image layer already exists. If Docker finds such a layer, it keeps a reference to the layer and proceeds to the next layer even if the layer is from a different image. This process is called ”layer deduplication” and is used most commonly for common base layers such as ”From ubuntu”. When building a new revision of an image, Docker uses layers built in previous revisions for unchanged layers. This process is referred to as ”caching”, and is used when building a new version of the image where only a small portion of the code is updated.

Ransomware threat and its impact on SCADA

Anzor Lors

Arvind Kumar

ABOUT THE AUTHOR

Anzor Lors

Versatile, detail-oriented specialist with superb academical background (MSc, MEng, BEng). Highly developed sense of responsibility and ability to adapt to changing priorities, contrive and implement new solutions to technical problems, and therefore effective at completing tasks with minimal supervision. Ambitious and driven by a challenge, pro-active and determined individual with diversified experience in different areas – R&D/Science, IT and Energy Sector. Enthusiastic communicator, confident to handle multiple tasks in team-oriented environment when required.

ABOUT THE AUTHOR

Arvind Kumar

A passionate developer with experience in managing a technology department of a British fund management company. Strong experiences in website development, UX/UI development and Fintech tools. Acquired deep knowledge in business and management through family business and further academics. My interests also expand into cyber security, Internet of Things and telecommunications engineering as these were crucial parts of my educational background.

Modern cyber crimes have exponentially grown over the last decade. Ransomware is one of the types of malware that is the result of sophisticated attempts to compromise modern computer systems. The governments and large corporations are investing heavily to combat this cyber threat against their critical infrastructure. It has been observed over the last few years that Industrial Control Systems (ICS) have become the main target of Ransomware due to the sensitive operations involved in the day to day processes of these industries. As the technology is evolving, more and more traditional industrial systems are replaced with advanced industry methods involving advanced technologies such as the Internet of Things (IoT). This technology shift helps improve business productivity and keeps companies globally competitive in an aggressive market. However, the systems involved need secure measures to protect integrity and availability, which will help avoid any malfunctioning to their operations due to the cyber-attacks. There have been several cyber-attack incidents in the healthcare, pharmaceutical, water cleaning and energy sectors. These ICS's are operated by remote control facilities and a variety of other devices such as programmable logic controllers (PLC) and sensors to make a network. Cyber criminals are exploring vulnerabilities in the design of these ICS's to take the command and control of these systems and disrupt daily operations until ransomware is paid. This article will provide critical analysis of the impact of the ransomware threat on SCADA systems.

I. INTRODUCTION

Since information technology integration and networking became more affordable and available around the world, industrial companies found a means of cost savings by connecting facilities together and controlling distributed systems from a single control center. There are numbers of increasingly diverse and extensively connected sets of technologies controlling significant parts of the global process within different sectors, such as pharmaceutical, electrical grid, oil refineries and pipelines, food manufacturing and modern rail systems used for logistics and public transportation every day. All these divisions of the advanced control technologies are represented by Industrial Control Systems (ICS). This includes components that allow them to increase their efficiency, accountability, and safety (SCADA, distributed control systems and programmable logic controllers). But at the same time, due to vulnerabilities in their security design, these systems are also prone to breaches [1]. While the components of the ICS's are heterogeneously connected, it means more efforts are required to ensure that there is an efficient and secure communication between these components. Unfortunately, due to the complexity of its design and lack of security framework, these critical systems are prone to cyber-attacks [2].

Nowadays, due to the ease of knowing the vulnerabilities of the security module in any ICS, whether connected via internet or local control units, it is getting easier for potential attackers to intrude and take down the operations of an entire system [3].

The cyber threats that industrial control system's operators face today are more challenging than ever before since the volume, types, and severity of cyber-attacks against ICS are rapidly increasing [4]. Operators across a range of industries disclosed that cyber intrusions in their networks had physically disrupted, and in some cases destroyed their systems. Cybercriminals expanded tactics and developed novel techniques for profiting off operational technology (OT) breaches, including selling access to supervisory control and data acquisition (SCADA) networks and