OCR MINER

OCR tabanlı hassas veri tespit yazılımı

https://github.com/melihi/ocr_miner

Geliştirici : Melih İşbilen
Tarih : 24/08/2023

https://www.linkedin.com/in/melih-i-325109a8/
melihisbilen[at]tutanota.de

İçerik

Giriş

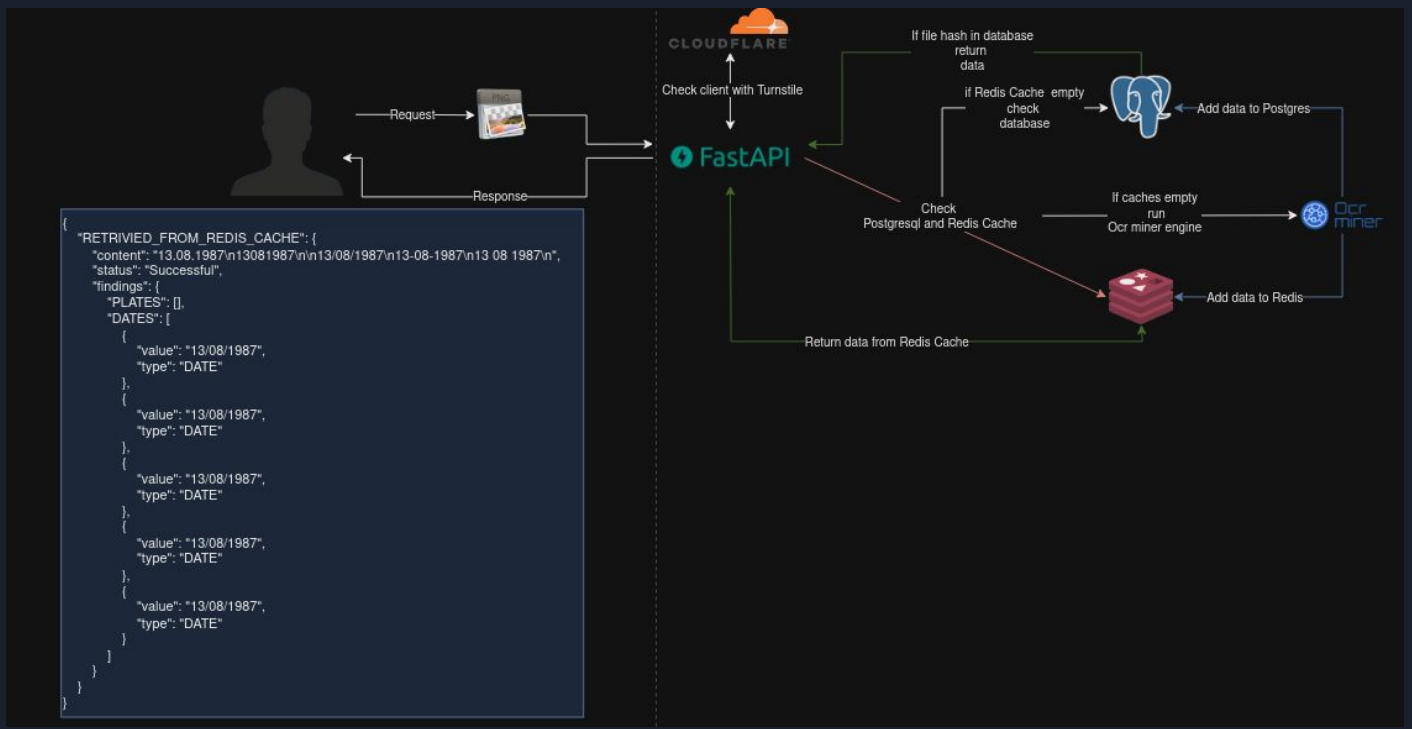
Giriş

Ocr miner, OCR teknolojisini kullanarak resimleri içerisinden önemli verileri elde eden bir yazılımdır. OCR teknolojisi, görüntüler veya yazılı metinler üzerindeki karakterleri tanıma ve yazıya çevirme yeteneğini ifade eder.

Ocr Miner potansiyel verileri regex ile tespit etmektedir. Elde edilen veriler 2. bir doğrulamadan geçirilir. Örneğin kredi kartı , Tc kimlik numaraları gibi belirli bir algoritmaya sahip veriler kendi algoritmalarına uyup uymadığı test edilir. Email, domain gibi bazı verilerde doğrulama için kütüphanelerden yararlanılmıştır. Ocr miner 10 farklı kritik veri tipini tanıyabilmektedir.

- Telefon numarası
- TC Kimlik, US Social Security Number, Europe VAT
- Kredi kartı
- Plaka
- Tarih
- Email
- Domain
- Url
- Hash
- Combolist

Elde edilen veriler Postgresql ve Redis'e dosya hash'i ile eklenerek cache'de tutulur. Kullanıcı tarafından girilen dosyanın hash'i önce Redis orada yoksa Postgresql üzerinde aranır. Hash bulunmazsa o zaman resim işlenmeye başlar.

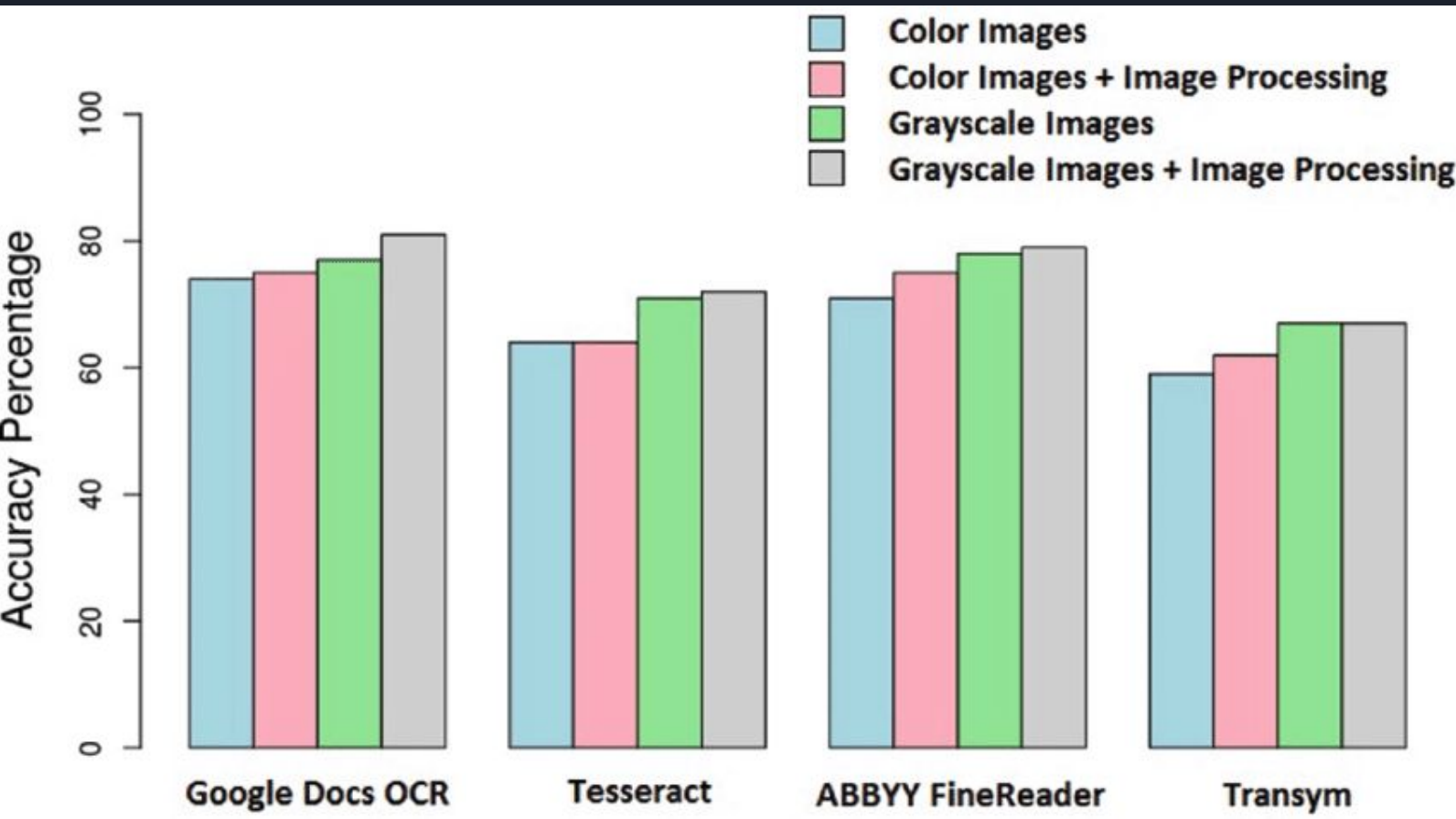


Giriş

Tesseract , Leptonica kütüphanesini kullanarak resimler üzerinde bazı iyileştirmeleri otomatik olarak yapmaktadır fakat bu iyileştirmeler yeterli seviyede değildir. Ocr miner kullanıcı tarafından verilen resim üzerinde grayscaling yapmaktadır . Bu iyileştirmeler Tesseract geliştiricilerinin önerileri doğrultusunda tasarlanmıştır.

Ocr miner resim üzerinden daha fazla ve daha doğru veri elde etmek için Opencv kütüphanesini kullanmaktadır. 2016 yılında Mehdi Assefi tarafından yayımlanan makaleye göre siyah beyaz resimler , renkli resimlere göre daha doğru sonuçlar almamızı sağlamaktadır. Bunlara ek olarak image processing başarıyı daha da arttırmaktadır. Ocr miner şimdilik image processing desteklememektedir.

https://github.com/tesseract-ocr/tessdoc/blob/main/ImproveQuality.md



Assefi, Mehdi. (2016). OCR as a Service: An Experimental Evaluation of Google Docs OCR, Tesseract, ABBYY FineReader, and Transym. ISCV.

Giriş

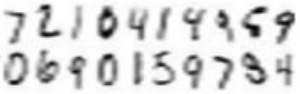
Tesseract , kütüphanesinin tercih edilme sebebi özgür yazılım olmasıdır. Tesseract, Google ve ABBYY gibi alternatiflerine göre el yazısı okuma , bulanık resim , çoklu dil içeren resimler gibi konularda geri olsada dijital rakamları okuma konusunda Google'dan bile daha iyi sonuç vermektedir.

Extracted characters					
Image category	Existing characters	Google Docs OCR	Tesseract	ABBY FineReader	Transym
Digital images	1834	1613 (87.95%)	1539 (83.91%)	1528 (83.31%)	1463 (79.77%)
Machine-written characters	703	569 (80.94%)	549 (78.09%)	574 (81.65%)	554 (78.81%)
Machine-written digits	211	191 (90.52%)	193 (91.47%)	193 (91.47%)	194 (91.94%)
Hand-written characters	2036	1254 (61.59%)	984 (48.33%)	1204 (59.14%)	960 (47.15%)
Hand-written digits	43	29 (67.44%)	11 (25.58%)	25 (58.14%)	10 (23.26%)

Quality attribute	Google Docs OCR	Tesseract	ABBYY FineReader	Transym
Open-source	No	Yes	No	No
Available online	Yes	No	Yes	No
Available as a SDK	No	Yes	Yes	Yes
Available as a Service	Yes	Could be	No	No
Multilingual support	Yes	Yes	Yes	Yes
Free	Yes	Yes	No	No
Operating systems	Any	Linux, Mac OS X, Windows	Linux, Mac OS X, Windows	Windows

Assefi, Mehdi. (2016). OCR as a Service: An Experimental Evaluation of Google Docs OCR, Tesseract, ABBYY FineReader, and Transym. ISCV.

Giriş

Image category	Sample image	Google Docs OCR	Tesseract	ABBYY FineReader	Transym
Digital images		RAE BUTLER BUILDING	RAB BUTLER BUILDING	Failed!	RAB BUTLER BUILDING
Machine-written characters		Explain that StUff	Explain that Stuff!	Explain that Stuff	Explain that Stuf f !
Hand-written digits		72104 9j8 DG90 I 597B4	7L/MI‘1M’7 0(9401547’54	721O4 I D 94 I 59 4	7LI04 I Y Db90 I 597b4
License plate number		Failed!	W8 02 H 6886	IND HB 02 H 6886	WB02 W 6886
Barcodes		0.1234" 56789	01234"56789	01234 5b789	Failed!
Digital receipt		Total Oued 53.80 CREDIT CARD 53.80	Total Owed 53.8) CREDIT FARO 53.8)	Total Owed 53.80 CREDIT CARD 53.80	T...tal Uved 53. REDIT t-FRD 53
Skewed images		Failed!	Failed!	Failed!	Failed!
Noisy images		Failed!	Failed!	Failed!	Failed!
Blurred images		agency	Failed!	Failed!	ingen(y
Multi-oriented text		A/ S/o ho, Awesome! abcdefghijk 1234567890	YG//o Awesome! 3 3 abcdefghijk 1234567890	Awesome!	Awesome! Go abcdefghijk 1234567890

Assefi, Mehdi. (2016). OCR as a Service: An Experimental Evaluation of Google Docs OCR, Tesseract, ABBYY FineReader, and Transym. ISCV.

Kullanılan Teknolojiler

Kullanılan Teknolojiler

- PostgreSQL
- Redis
- Python3.11
  - SqlAlchemy
  - FastAPI
  - Python-magic
  - DynaConf
  - Poetry
  - Pydantic
  - Click
  - email-validator
- Tesseract
- OpenCv
- Bootstrap
- Docker
- Google Cloud

Veri Tespit Teknikleri

Veri Tespit Teknikleri

Ocr Miner kullanıcıdan gelen resim üzerinde grayscaling işlemi gerçekleştirmektedir . Ardından Tesseract ile elde edilen veri üzerinde Regex + ek doğrulama adımları ile en fazla ve en doğru veriye ulaşmayı amaçlamaktadır. Veri doğrulama işlemleri sırasında aktif yöntemler kullanılmamaktadır. Örneğin elde edilen domain tehdit aktörleri tarafından daha önce kullanılmış ve aktif olmayabilir . Ya da tehdit aktörlerine ait sistemlerde iz bırakmak operasyon güvenliğini etkileyebilir.

Kredi Kartı Tespiti

Kredi kartlarının tespiti için bir adet regex kullanılmıştır. Elde edilen potansiyel Kredi kartları Luhn algoritmasına sokularak geçerli olup olmadığı test edilir.

Luhn Algoritması, özellikle kredi kartı numaralarının geçerliliğini kontrol etmek için kullanılan bir sayı doğrulama yöntemidir. Kredi kartı numaralarının geçerliliğini hızlı bir şekilde kontrol etmek amacıyla kullanılan bu algoritma, yanlış yazılmış veya hatalı girilmiş numaraların tespit edilmesine yardımcı olur.

Luhn algoritması adımları:

1. Öncelikle kredi kartı numarasının 0–15 aralığında karşılık değerleri eşleştirilir
2. En sağdan başlayarak 0–15 aralığında tek numaraya karşılık gelen değerler toplanır
3. En sağdan başlayarak 0–15 aralığında çift numaraya karşılık gelen değerler sıralanır , sonrasında bu değerlerin her biri 2 ile çarpılır. Eğer çarpım sonucunda 2 haneli sayı elde edilirse bu sayılar kendi aralarında toplanır.
4. Son olarak, bu toplamın bir ondalık sayı olup olmadığı kontrol edilir. Eğer bu toplam 10'a tam bölünüyorsa, kredi kartı numarası geçerlidir.

Ocr Miner tarafından tespit edilebilen kart yazım formatları:

- 3530-1113-3330-0000
- 6011000990139424
- 5105 1051 0510 5109

```
CC_PATTERNS = [r"(?:\d{4}[ -]?){3}\d{4}"]
```

Resim 12.1 Kullanılan Regex

Hash Tespiti

Hash tespiti için 5 farklı regex kullanılmıştır . Tespit edilen hash değerleri :

- Md5
- Md4
- Sha1
- Sha256
- Sha512
- NTLM

```
HASH_PATTERNS = [
    r"[a-f0-9]{32}",
    r"[a-f0-9]{64}",
    r"(\\$NT\\$)?[a-f0-9]{32}",
    r"[a-f0-9]{40}",
    r"[a-f0-9]{128}",
]
```

Resim 13.1 Kullanılan Regex

Kullanılan regex paternleri a karakterinden f karakterine , 0'dan 9'a kadar olan ve 32 , 64 , 40 ve 128 karakter uzunluğuna sahip olan string ifadeleri potansiyel hash olarak değerlendirilmektedir .

Elde edilen bu string ifadelerin geçerli hash olup olmadığını tespit etmek için Ocr Miner bu ifadelerin entropi değerini hesaplamaktadır . Shannon Entropisi, bir veri dizisinin veya bir rastgele değişkenin tahmin edilebilirliğini ölçmek için kullanılan bir kavramdır. Shannon Entropisi ne kadar yüksekse, veri dizisinin tahmin edilmesi o kadar zor ve rastgelelik seviyesi o kadar yüksektir. Shannon Entropisi, bilgi kuramı ve istatistiksel analizlerde sıklıkla kullanılır.

49 adet md5 hash'inin Shannon entropi değerini hesaplayıp topladığımızda 177.49 çıkmaktadır . Hash'e dönüştürdüğümüz ifadeler rockyou.txt listesinin ilk 49 satırıdır.

Ardından 177.49 / 49 işlemi ile ortalama entropi 3.62428571429 olarak bulunmuştur. Bu istatistik sebebi ile regex ile bulunan ifadelerin entropi değeri 3'ün altında ise hash olarak kabul edilmemektedir .

$$H(X) = \sum -p_i \log_2 p_i$$

Resim 13.2 Shannon Entropy

Hash Tipi	Entropi
Md-5	3.62428571429
Sha-1	3.6912244898
Sha-256	3.82163265306
Sha-512	3.91081632653

Resim 13.3 Her test için aynı 49 kelime kullanılmıştır.

ID Tespiti

Ocr miner Social security number(SSN), TC Kimlik No, VAT(Value Added Tax) bilgilerini tespit etme kabiliyetine sahiptir .

Value Added Tax (VAT), mal ve hizmetlerin üretim ve dağıtım aşamalarında eklenen ve son tüketici tarafından ödenen bir tüketim vergisi sistemidir. VAT, genellikle ülkelerin kamu gelirlerini artırmak, vergi yükünü yaymak ve vergi kaçakçılığını azaltmak amacıyla kullanılır. Farklı ülkelerde farklı oranlarda ve yöntemlerle uygulanabilir. Regex ile bulunan VAT değerleri pyvat kütüphanesi ile doğrulanmaktadır .

Amerika Birleşik Devletleri'nde (ABD), "Social Security Number" (SSN) olarak adlandırılan bir kimlik numarası bulunmaktadır. SSN, Amerikalı vatandaşlar, kalıcı ikamet eden yabancılar ve çalışma iznine sahip kişilere verilen benzersiz bir kimlik numarasıdır. SSN formatı “AAA-GG-SSSS” şeklindedir . SSN matematiksel bir algoritmaya sahip değildir . Bu sebeple “-” ve hane sayıları kontrol edilmektedir .

Türkiye Cumhuriyeti Kimlik Numarası (TC Kimlik No), 11 haneli benzersiz bir kimlik numarasıdır. Bu numara, Türkiye'deki her bireye doğumla birlikte verilir. Kimlik numarası, aşağıdaki algoritma ile oluşturulur:

- İlk 9 hane, doğum tarihini (yıl, ay, gün) yansıtır.
- 10. hane, cinsiyeti belirtir. Tek sayılar erkek, çift sayılar kadın cinsiyeti için kullanılır.
- 11.hane ise, diğer 10 hane için oluşturulan bir doğrulama rakamıdır. Diğer hanelerin toplamının 10'a göre modu alınarak hesaplanır.

```
ID_PATTERNS = [
    r"[1-9]{1}[0-9]{9}[02468]{1}",
    r"\b(?!000|.+0{4})(?:\d{9}|\d{3}-\d{2}-\d{4})\b",
    r"([A-Z]{2}\d{9,12})"
```

Resim 14.1 Kullanılan Regex

Plaka Tespiti

Plaka tespiti için 6 adet Regex kullanılmaktadır . Desteklenen ülkeler :

- Türkiye
- USA
- Almanya
- Fransa
- Rusya
- Çin

Dünyadaki plaka standartları benzer olduğu için her ne kadar 6 ülke için özel Regex'ler hazırlanmış olsada 6'dan fazla ülke tespit edilebilmektedir .

Tespit edilen potansiyel plakalar ikinci doğrulamaya tabi olmaktadır. Bulunan string değer 2.bir regex ile test edilir. Buna ek olarak string'de ki karakterlerin alfanümerik olduğu doğrulanır .

- *Amerika Birleşik Devletleri (ABD): ABD'de plaka formatları eyaletlere göre değişebilir, ancak genellikle 1-7 karakter uzunluğunda olabilir.*
  - *Örneğin: "123ABC", "ABC123".*
- *Almanya: Almanya'da plakalar genellikle 2 harf ve ardından 1 ila 4 rakamdan oluşur.*
  - *Örneğin: "AB123", "XY9999".*
- *Fransa: Fransa'da plakalar genellikle 2 rakam, ardından tire (-) veya boşluk, sonra 2 rakam, tekrar tire veya boşluk, ve son olarak 2 rakamdan oluşur. Örneğin: "12-34-56", "99 99 99".*
- *Rusya: Rusya'da plakalar genellikle 1 harf, ardından 3 rakam ve son olarak 2 harfden oluşur.*
  - *Örneğin: "A123BC", "1A234B".*
- *Çin: Çin'de plakalar genellikle 1 harf, ardından 1 harf veya 1 rakam, ve son olarak 4 veya 5 harf veya rakamdan oluşur.*
  - *Örneğin: "A123BC", "AB123CD".*

```
# turkey, usa , germany,french,russia,china
PLATE_PATTERNS = [
    r"\b\d{2}\s?[A-HJ-NPR-TV-Z]{1,3}\s?\d{2,4}\b|\b\d{2}-?[A-HJ-NPR-TV-Z]{1,3}-?\d{2,4}\b",
    r"\b[A-HJ-NPR-Z0-9]{1,7}\b",
    r"\b[A-Z]{1,2}\s?[A-Z0-9]{1,4}\s?[A-Z0-9]{1,4}\b",
    r"\b[A-Z0-9]{2}-?[A-Z0-9]{2}-?[A-Z0-9]{2}\b",
    r"\b[A-Z0-9]{1}\d{3}[A-Z0-9]{2}\b",
    r"\b[A-Z]{1}[A-HJ-NP-Z]{1}[A-HJ-NP-Z0-9]{4,5}\b",
]
```

Resim 15.1 İlk kullanılan Regexler

```
r"^(?=.*[a-zA-Z])(?=.*\d).+$"
```

Resim 15.2 Doğrulama için kullanılan Regex

Tarih Tespiti

Tarih tespiti için bir adet regex kullanılmaktadır . Tespit edilen potansiyel tarih verileri gün/ay/yıl formatına sokulmaktadır.

```
DATE_PATTERNS = [
    r"([1-9]|[12][0-9]|3[01])(|\/|\.|\-|\s)?(0[1-9]|1[12])\2(19[0-9]{2}|200[0-9]|201[0-8])"
]
```

Resim 16.1 İlk tespit için Regex

```
13.08.1987
13081987
13/08/1987
13-08-1987
13 08 1987
```

Resim 16.2 Tespit edilen farklı tiplerde tarihler

Çeşitli formatlardaki tarih verileri tespit edilebilmektedir . Regex verileri tanıdığında list veri yapısı içerisinde tuple veri yapıları şeklinde veri döndürmektedir.

```
[('13', '.', '08', '1987'),
 ('13', '', '08', '1987'),
 ('13', '/', '08', '1987'),
 ('13', '-', '08', '1987'),
 ('13', ' ', '08', '1987')]
```

Elde edilen bu veri aşağıda görülen algoritma ile gün/ay/yıl formatında normalize edilmektedir .

```
for date in found_data:
    extract_date = ""
    for string in date:
        for word in string:
            if word.isdigit():
                extract_date += word
        if extract_date[-1] != "/":
            extract_date += "/"
    # delete last / in date string  Example: 11/11/1987/
    extract_date = extract_date[:-1]
```

Resim 16.3 Tespit edilen verileri normalize eden kod

Domain Tespiti

Domain tespiti için bir adet regex kullanılmıştır. Tespit edilen veriler 2.bir doğrulamadan geçirilmiştir. Tespit edilen potansiyel veri , IANA üzerinden elde edilen TLD listesi içerisinde bulunmayan bir TLD'ye sahip ise geçersiz kabul edilir.

- https://data.iana.org/TLD/tlds-alpha-by-domain.txt

Ocr Miner başlatıldığında liste çeklir ve yazılım sonlandırılana kadar kullanılır. Bu sayede gereksiz network trafiği engellenir ve performans artar.

```
DOMAIN_PATTERNS = [
    r"(?:[_a-z0-9](?:[_a-z0-9-]{0,61}[a-z0-9])?\.)+(?:[a-z](?:[a-z0-9-]{0,61}[a-z0-9])?)"
]
```

Resim 17.1 İlk tespit için Regex

```
def verify_domain(domains: list) -> bool:
    """Check TLD from IANA tld list. This list Updating automatically by IANA.

    Args:
        domain (str): domain name.
    """

    tmp = []
    try:
        for domain in domains:
            if domain.split(".")[-1].upper() in IANA_CONTENT:
                tmp.append({"value": domain, "type": "DOMAIN"})
    except requests.exceptions.ConnectionError as e:
        logging.warning(e)
    return tmp
```

Resim 17.2 Doğrulama yapan fonksiyon

```
    "GQ",
    "AE",
    "IS",
    "LV",
    "PRO",
    "FM",
    "TIPS",
    "MS",
    "SA",
    "APP",
]

IANA_URL = "https://data.iana.org/TLD/tlds-alpha-by-domain.txt"


def download_iana():
    try:
        global IANA_CONTENT
        tlds = requests.get(IANA_URL)      You, 14 hours ago • first comm
        if tlds.status_code == 200:
            IANA_CONTENT = tlds.text.split("\n")

    except Exception as e:
        logging.warn("An error occurred while download iana list:", e)
```

Resim 17.3 TLD listesi inerken hata olursa kullanılmak üzere en fazla kullanılan 100 TLD listesi statik olarak tutulmaktadır

Url Tespiti

Url verilerinin tespiti için bir adet regex kullanılmıştır. Regex ile elde edilen potansiyel verileri doğrulamak için Urllib kütüphanesi kullanılmıştır. Elde edilen verileri doğrulamak için aktif yöntemler kullanılmamıştır.

```
URL_PATTERNS = [
    r"(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-z]{2,4}/)(?:[^\s()<>]+|\(([^\s()<>]+|(\([^\s()
    <>]+\)))*\))+(?:\(([^\s()<>]+|(\([^\s()<>]+\)))*\)|[^\s`!()\[\]{};:'\".,<>?«»“”‘’]))"
]
```

Resim 18.1 Kullanılan Regex

```
parsed_url = urlparse(url[0])
if parsed_url.scheme and parsed_url.netloc:
    tmp.append({"value": url[0], "type": "URL"})
```

Resim 18.2
URL'nin "scheme" (örneğin, "http" veya "https") ve "netloc" (alan adı) bileşenlerinin boş olmadığını kontrol eder.

Email Tespiti

Email tespiti için bir adet regex kullanılmaktadır. Elde edilen potansiyel veriler 2. bir doğrulamaya tabi olur. Bu doğrulama için email_validator paketi kullanılmaktadır .

https://pypi.org/project/email-validator/

```
EMAIL_PATTERNS = [r"\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,7}\b"]
```

Resim 19.1 Kullanılan Regex

```
for email in found_data:
    try:
        mail = validate_email(email, check_deliverability=False)
        tmp.append({"value": mail.normalized, "type": "EMAIL"})

    except EmailNotValidError as e:
        logging.warn(e)
```

Resim 19.2 Verilerin doğrulanması

Combolist Tespiti

"email:password" formatında bulunan büyük metin dosyaları, genellikle "password list" veya "password dump" olarak adlandırılır. Bu dosyalar, kullanıcıların e-posta adresleri ve kullanıcı adları ile ilişkilendirilmiş parola kombinasyonları içerir.

Bu tür dosyalar, siber saldırganlar tarafından kullanıcı hesaplarını ele geçirme amacıyla kullanılabilir. Büyük veri sızıntıları veya güvenlik ihlalleri sonucu elde edilen bu veriler, saldırganların çeşitli hizmetlerdeki hesapları ele geçirmeye çalışmasına olanak sağlar.

Ocr Miner combolist verilerini tespit ederken bir adet regex kullanmaktadır. Elde edilen veriler için bir adet ":" karakterini olması şartı kontrol edilir. Eğer "http://" gibi bir ifade geçiryorsa bu veride en az iki adet ":" karakteri bulunmak zorundadır .

```
COMBOLIST_PATTERNS = [r"[a-zA-Z0-9_.+-]+@[a-zA-Z0-9.-]+:[a-zA-Z0-9._-]+"]
```

Resim 20.1 Kullanılan Regex

Telefon Tespiti

Telefon numaralarının tespiti için bir adet regex kullanılmıştır . Kullanılan regex'in sabit hatlar ve mobil hat numaralarını çeşitli formatlarda tanıyabilmektedir.

- 202-918-2132
- +12029182132
- 202-918-2132
- +905053535555
- (505) 3535555
- (201) 874-8593
- 02231231212
- 090553123121
- 5331231212
- 090.553123121

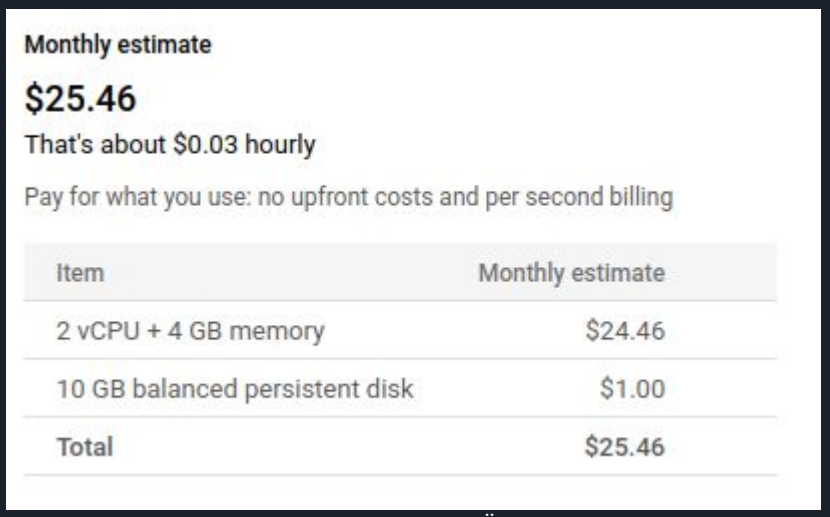
```
PHONE_PATTERNS = [
    r"[\+]?[(]?[0-9]{3}[)]?[-\s\.]?[0-9]{3}[-\s\.]?[0-9]{4,6}",
]
```

Resim 21.1 Kullanılan Regex

Deployment

Deployment

Ocr miner için Google Cloud platformu tercih edilmiştir. Bunun sebebi yeni kullanıcılara 300$ dolar ücretsiz kullanım hakkı tanımasıdır. 4 çekirdek işlemci , 2GB ram ve 10GB depolama alanına sahip E2 Medium makinesinin aylık ücreti $25.46 dolardır. Basit bir hesapla ortalama 11 ay boyunca bu makinayı kullanabilmektedir.



Monthly estimate

$25.46

That's about $0.03 hourly

Pay for what you use: no upfront costs and per second billing

Item	Monthly estimate
2 vCPU + 4 GB memory	$24.46
10 GB balanced persistent disk	$1.00
Total	**$25.46**

Resim 23.1 E2 Medium Ücret Tablosu

Production ortamlarında FastApi için önerilen Gunicorn kullanmaktır . Bunun sebebi Gunicorn'un daha gelişmiş process yönetimlerine sahip olmasıdır. Örneğin uygulamamız crash olduğunda Gunicorn ile bu durumu yönetmek daha kolaydır. FastApi ASGI standartlarına göre tasarlanmıştır bu sebepten genellikle Uvicorn ile kullanılmaktadır . Gunicorn kullanımını gerçekleştirebilmek için farklı yöntemler kullanılmaktadır. Örneğin:

```
gunicorn ocr_miner.api.ocr_miner_api:APP --workers 2 --worker-class
uvicorn.workers.UvicornWorker --bind 0.0.0.0:8000
```

WSGI, Python web uygulamalarını sunucularla tek bir thread veya process üzerinde çalıştırır ve gelen istekleri sırayla işler. Bu yaklaşım ağır işlem gerektiren isteklerin sunucuyu yavaşlatabileceği anlamına gelir. Geleneksel Python web frameworkleri (Flask , Django) genellikle WSGI ile uyumludur.

ASGI, Python web uygulamalarının birden fazla isteği aynı anda işlemesi için tasarlanmıştır, bu da daha iyi performans ve ölçeklenebilirlik anlamına gelir. Asenkron bir yapıya sahiptir, yani gelen istekleri ve yanıtları aynı anda işleyebilir.

https://fastapi.tiangolo.com/deployment/server-workers/

https://docs.gunicorn.org/en/latest/settings.html#bind

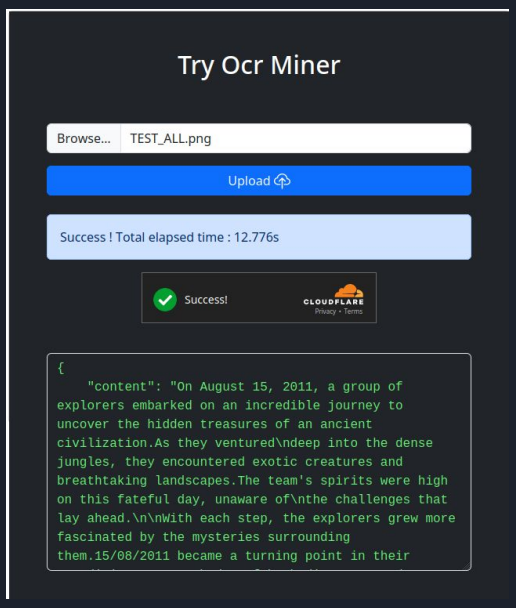
Sonuç

Sonuç

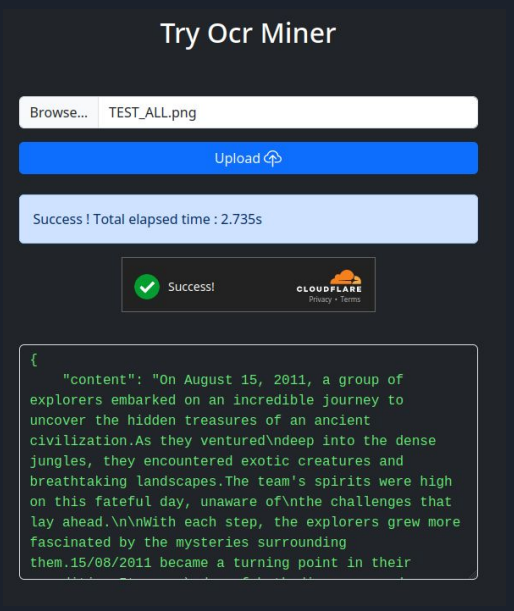
Kısıtlı sürede gerçekleştirilen yoğun çalışma ile verilerin tespiti konusunda başarılı sonuçlar elde edilmiştir . Yapılan optimizasyonlar sayesinde ortalama 12sn'de okunan veriler 3sn ortalamalarına düşürülmüştür .

Uygulanan bazı optimizasyonlar :

- IANA TLD listesi gibi veriler elde edilirken yapılan istekler , döngüler yazılım başlatıldığı anda bir defa gerçekleştirilerek bu işlemlerin tekrar tekrar gerçekleşmesi önlendi.
- Upload edilen dosya üzerinde işlem yaparken , `file.file.read()` fonksiyonu ile sürekli veri okunması sorunu düzeltildi. Bir kere okunup gerekli yerlerde kullanıldı.
- OpenCv tarafından üretilen grayscale resimleri diske yazmayarak hem disk i/o hemde disk kapasitesi verimli kullanılmıştır.



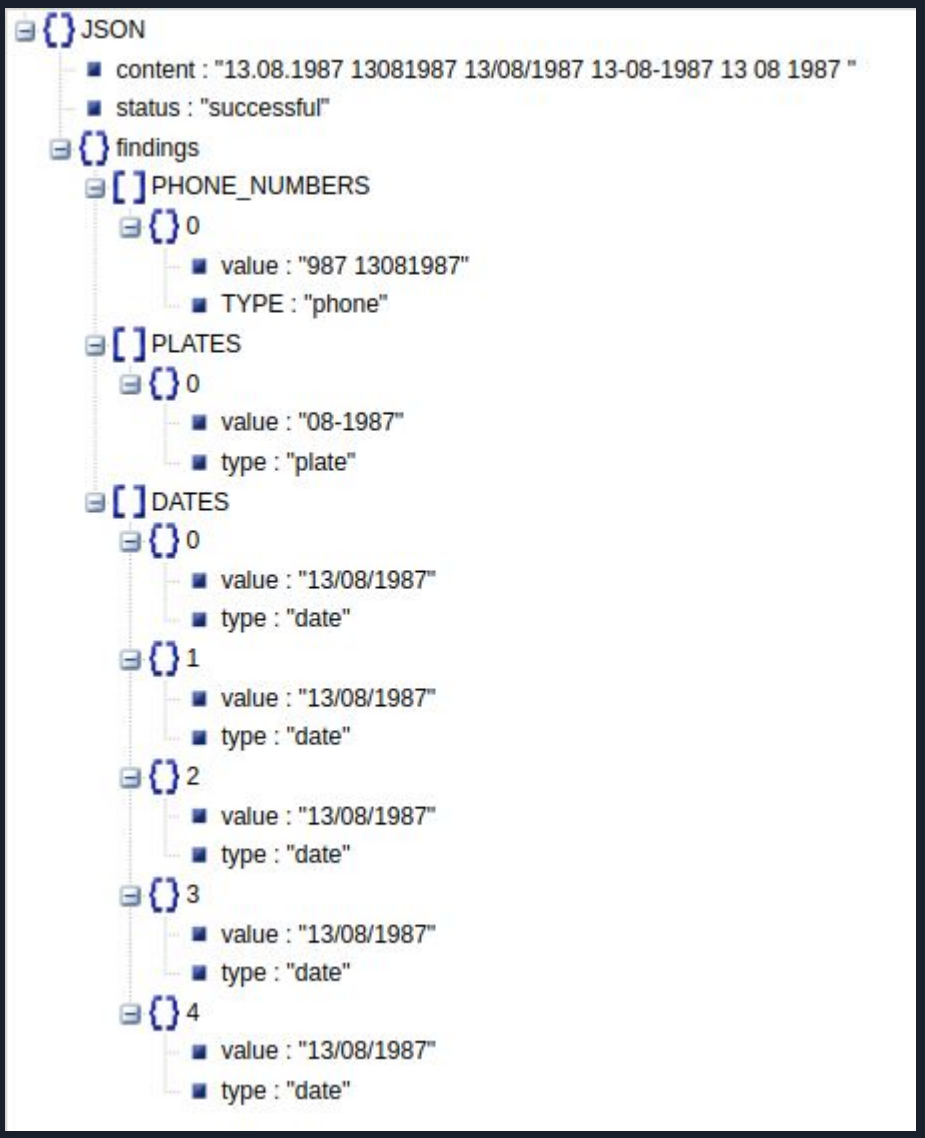
Resim 25.1 Optimizasyon öncesi 12.776s



Resim 25.2 Optimizasyon sonrası 2.735s

Sonuç

```
13.08.1987
13081987
13/08/1987
13-08-1987
13 08 1987
```



Sonuç

```
+1 202-918-2132
+12029182132
1-202-918-2132
+905053535555
(505)3535555
(201) 874-8593
02231231212
0223 123 12 12
00905531231212
+90.5331231212
0090.5531231212
```

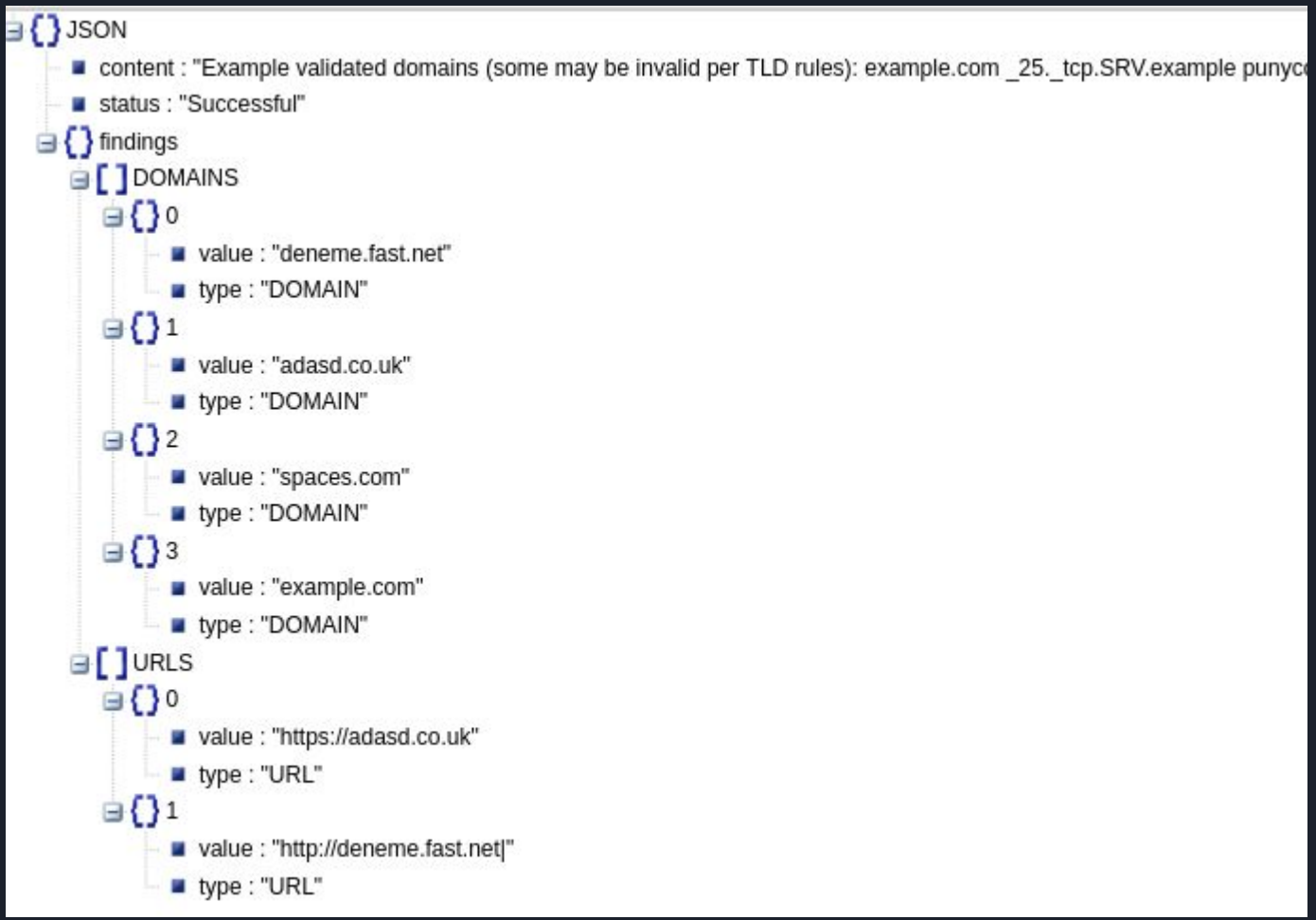

JSON
- content : "+1 202-918-2132 +12029182132 1-202-918-2132 +905053535555 (505) 3535555 (201) 874-8593 02231231212 0223 123 12 12 0905531231212 +90.5331231212 0090.5531231212| "
- status : "successful"
- findings
  - PHONE_NUMBERS
    - 0
      - value : "202-918-2132"
      - TYPE : "phone"
    - 1
      - value : "+12029182132"
      - TYPE : "phone"
    - 2
      - value : "202-918-2132"
      - TYPE : "phone"
    - 3
      - value : "+905053535555"
      - TYPE : "phone"
    - 4
      - value : "(505) 3535555"
      - TYPE : "phone"
    - 5
      - value : "(201) 874-8593"
      - TYPE : "phone"
    - 6
      - value : "02231231212"
      - TYPE : "phone"
    - 7
      - value : "090553123121"
      - TYPE : "phone"
    - 8
      - value : "5331231212"
      - TYPE : "phone"
    - 9
      - value : "090.553123121"
      - TYPE : "phone"
  - ID_NUMBER
    - 0
      - value : "12029182132"
      - TYPE : "UNKNOWN"
  - PLATES
    - 0
      - value : "3535555"
      - type : "plate"

Sonuç

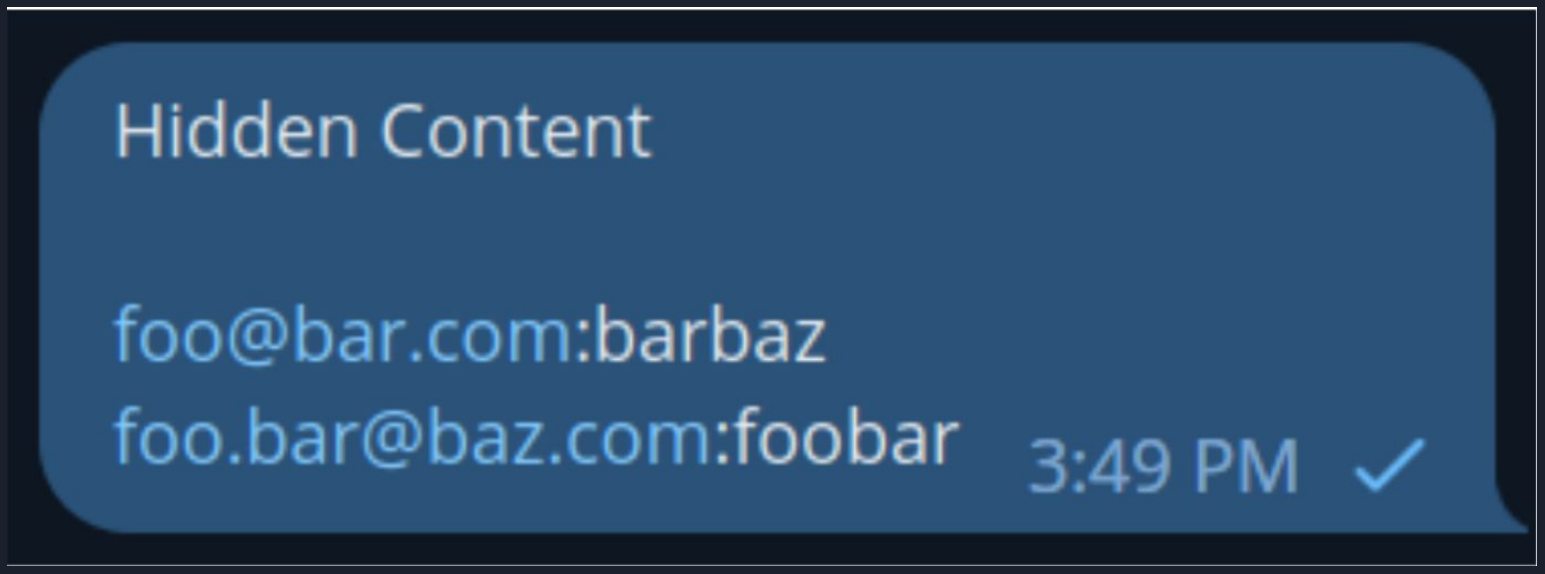

```
"BE0121431789",
"BG121431789",
"BG1214317890",
"CY12143178X",
"CY12143178L",
"CZ12143178",
"CZ121431789",
"CZ1214317890",
"DE121431789",
"IN12144312142",
"DK12143178",
"EE121431789",
"EL121431789",
"GR121431789",
"ESX12143178",
"ES12143178X",
"ESX1214317X",

56879284492
60925736682
57853151038


ssn
010809851
010 80 9851
001-26-4753
```

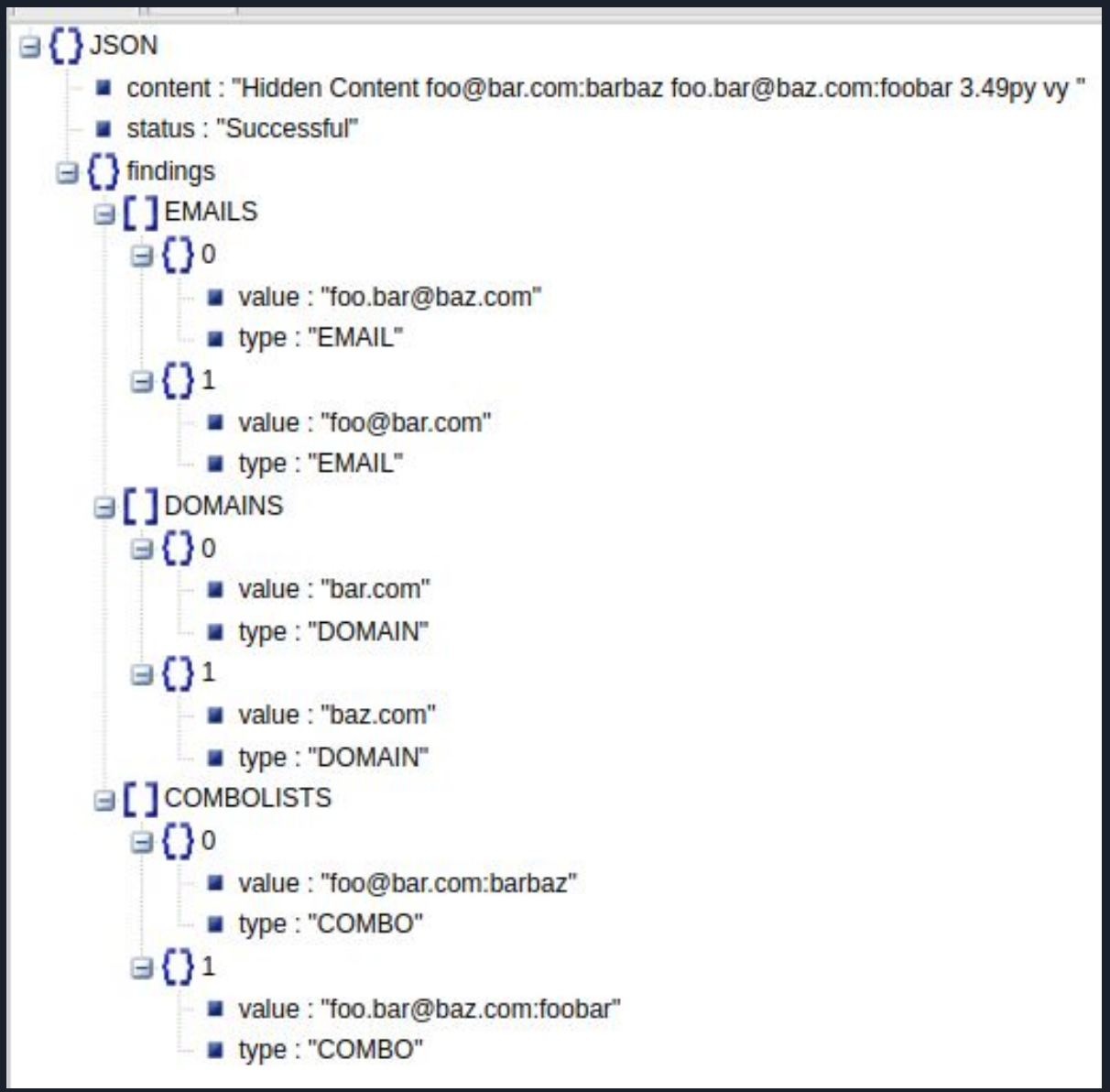
Sonuç

```
Example validated domains (some may be invalid per TLD rules):

example.com
_25._tcp.SRV.example
punycoded-idna.xn--zckzah
under_score.example
https://adasd.co.uk
http://deneme.fast.net
Example invalid domains:

192.0.2.1
has spaces.com
easy,typo.example
domain\.escapes.invalid
no_trailing_.invalid
-leading-or-trailing-.hyphens.invalid
```



Sonuç


Hidden Content
foo@bar.com:barbaz
foo.bar@baz.com:foobar
3:49 PM


JSON
content : "Hidden Content foo@bar.com:barbaz foo.bar@baz.com:foobar 3.49py vy "
status : "Successful"
findings
EMAILS
0
value : "foo.bar@baz.com"
type : "EMAIL"
1
value : "foo@bar.com"
type : "EMAIL"
DOMAINS
0
value : "bar.com"
type : "DOMAIN"
1
value : "baz.com"
type : "DOMAIN"
COMBOLISTS
0
value : "foo@bar.com:barbaz"
type : "COMBO"
1
value : "foo.bar@baz.com:foobar"
type : "COMBO"

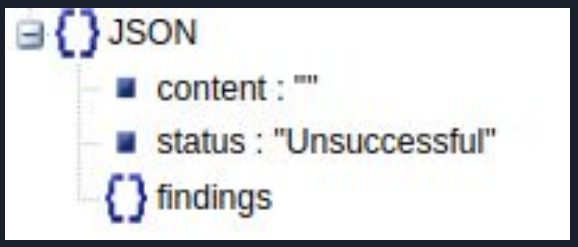
Sonuç

Herkese selam! Bugün sizinle cc paylaşımı yapacağım. Artık burada daha aktif olmaya karar verdim. Daha fazla içeriğe sahip olmak için beni mutlaka takip edin. Paylaştığım cc yi direk kullanabilirsiniz checklenmiştir.

6011779370011770

Daha fazla içerik için telegram kanalımı da takip edebilirsiniz https://t.me/foo

Özelden ulaşmak isteyenler için mail adresim foo_bar06@gmail.com


JSON
content : ""
status : "Unsuccessful"
findings