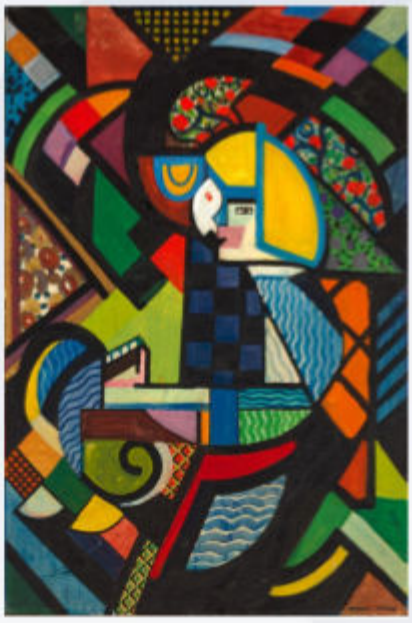
Technical Report

DomainTools

Comparing Images Such as Screenshots Using Perceptual Hashes

Joe St Sauver, Ph.D.
Distinguished Scientist

Saÿen's daughter-in-a-rocker.jpg



daughter-in-a-rocker-grayscale.jpg

Perceputal Hash Distance: 0.03125
Threshold is: 0.2 --> Images are Similar

Table of Contents

Executive Summary

DomainTools routinely collects screenshots for base ("registrable") domains. Along with our proprietary risk scores, those screenshots are one of the most-appreciated (if currently under-leveraged) artifacts provided in DomainTools Iris Investigate and other company products.

Unlike most data elements DomainTools collects, screenshots are not programmatically accessible by customers. If you want to access a screenshot the company collected, to ensure compliance with the company's terms of service, you must use the company's GUI web interface, or collect screenshots of interest yourself directly, outside of the DomainTools interface.

Once screenshots have been obtained, being able to mechanically determine screenshot similarity can enable an analyst to find "look-alike" sites used for phishing, brand-infringement purposes, or to find clusters of likely-related sites based on a graphical comparison of screenshots. That's a powerful capability, but one which is not without some limitations.

We illustrate the process with gambling-related domains, and conclude with those limitations.

Code for the various programs used in the report are provided in appendices.

Part I. Getting Access to Screenshots

Introduction

DomainTools routinely collects and archives screenshots for base registrable domains. Along with our proprietary risk scores, screenshots are one of the most prominent and customer-appreciated (if currently "under-leveraged") artifacts provided in Iris Investigate interface and other company products.

What's Special About Screenshots? Screenshots differ from risk scores and other API-accessible DomainTools data in that they are one of the few data elements NOT available via the DomainTools API[1]. Screenshots are normally only accessible via the company's interactive GUI web interfaces:

- `https://research.domaintools.com/research/screenshot-history/`
- `https://iris.domaintools.com/investigate/`(under "Screenshot History")

We'll show you examples of what those GUI interfaces look like later in this report. We've also previously discussed the process of collecting bulk screenshots via Iris Investigate pivot tables in the company's blog.

To ensure compliance with the company's terms of service, we assume that users interested in collecting screenshots will either manually request and then manually save those screenshots from the company's existing Iris Investigate GUI interface (right click --> Save Image As...) or collect their own screenshots (external to DomainTools), perhaps using Pyppeteer as described in Section 9 of our earlier ""Bang_Question:" A Tutorial Proof-of-Concept Cyber Investigative Framework" whitepaper[2].

Part II: Comparing Pairs of Images

"Semper idem"
["Always the same"]

Comparing Pairs of Images

Having retrieved screenshots of interest, we need to figure out how we're going to compare pairs of images with an eye toward identifying similar screenshots. For the purposes of this document, we'll highlight two basic approaches:

- **Identifying IDENTICAL Images:** In the simplest of cases, imagine a set of images, some identical. If true identical images are all we care about, an easy way to uncover them uses classic cryptographical hashes. If we were to compute a classic cryptographic hash (such as an MD5[3] or SHA256 checksum) for those, the identical images would

[1] https://www.domaintools.com/resources/api-documentation/
[2] https://www.farsightsecurity.com/assets/documents/bang-question-report.pdf
[3] https://docs.python.org/3/library/hashlib.html

have identical hashes. For example, assume we're in a directory full of screenshots in JPEG format.

We can compute md5sums for all those JPEGs and save them to a text file by saying:

```
$ md5sum *.jpg > md5sum.txt
```

We then check for any duplicate checksums by saying:

```
$ awk '{print $1} < md5sum.txt | sort | uniq -c | sort -nr | more
   12 2e1135ffbe5db284a4ade7806d4ef75d
   10 d79830e58328a6903fc2d4ccc4d69661
   9 6bb2e6a8e51b361b983b5bf690dd2cee
   9 06c4f0367d944e321515a299fb5a79f7
   6 c0e3eccbd5602137cf94194279a7aa0f
    6 961ea8cb093d6d43db8814630822f750
   6 93044285c70f9a6d3c7dde8c7020a296
   6 84cef7bfd93f0d448afe6c4d2e7be002
   6 5fcdefe5215e9de1fb3a12e591d89335
[etc]
```

We can find the images that match a given checksum by grep'ing the `md5sum.txt` file:

```
$ grep 2e1135ffbe5db284a4ade7806d4ef75d md5sum.txt
2e1135ffbe5db284a4ade7806d4ef75d  777-poker.es.jpg
2e1135ffbe5db284a4ade7806d4ef75d  777poker.ch.jpg
2e1135ffbe5db284a4ade7806d4ef75d  777poker.nl.jpg
2e1135ffbe5db284a4ade7806d4ef75d  poker-777.at.jpg
2e1135ffbe5db284a4ade7806d4ef75d  poker-777.ch.jpg
2e1135ffbe5db284a4ade7806d4ef75d  poker-777.pt.jpg
2e1135ffbe5db284a4ade7806d4ef75d  poker777.at.jpg
2e1135ffbe5db284a4ade7806d4ef75d  poker777.ch.jpg
2e1135ffbe5db284a4ade7806d4ef75d  poker777.com.es.jpg
2e1135ffbe5db284a4ade7806d4ef75d  poker777.com.mx.jpg
2e1135ffbe5db284a4ade7806d4ef75d  poker777.lu.jpg
2e1135ffbe5db284a4ade7806d4ef75d  poker777.ro.jpg
```

In this case, while there are twelve different screenshots (apparently associated with different web sites), they are all bit-for-bit identical.

- **Identifying SIMILAR Images:** Other times, two images might be quite "similar" even though there may be "minor" differences. Having even a one pixel difference will be enough to make classic hash functions return totally different values, but there are

"approximate" hashes that can be used to tell if two different images tend to "look similar" to humans.

For this report, we're going to use the open source Thorn perceptual hash library[4]. This library was originally created to facilitate identification of child sexual abuse materials ("CSAM")[5] even when those images may have been intentionally modified in an attempt at thwarting forensic identification and blocking/reporting. Fortunately, the Thorn hash libraries work great for more mundane tasks such as comparing screenshots, too. Comparing images with perceptual hashes requires two steps:

- Computing the perceptual hash[6] for each image, and then
- Computing the Hamming distance[7] between the ref image and the comparator

If that "distance" is <= a specified threshold value, the images are said to be **similar** to each other; if the distance is greater than the threshold, they're said to be **different**.

This is not an exhaustive list of potential approaches for image comparison. There are both simpler approaches (such as using the MSE[8]) and more complex approaches (such as machine learning-based approaches[9]), but those are out of scope for this document.

Proof of Concept Code for Perceptual Hashing

To concretely demonstrate the power of perceptual hashes, we decided to do some tests on a sample image (as shown on the cover of the report). The image we selected was from H. Lyman Saÿen, a U.S. scientist and inventor who worked in the area of high voltage electricity and the design of X-ray tubes, but who also quite an accomplished painter[10] before dying prematurely at the age of 43.

We obtained a 6.5MB JPEG (1994x3000px@300dpi) copy of Saÿen's "Daughter in a Rocker"[11] (1917-1918) from the Smithsonian American Art Museum Open Access collection (image in the public domain under CC0,[12] image credit "a gift from H. Lyman Saÿen to his nation").

We then proceeded to modify that base image in a variety of ways, checking to see whether the modified image continued to be identified as "similar" to the original. For the purposes of our tests, we

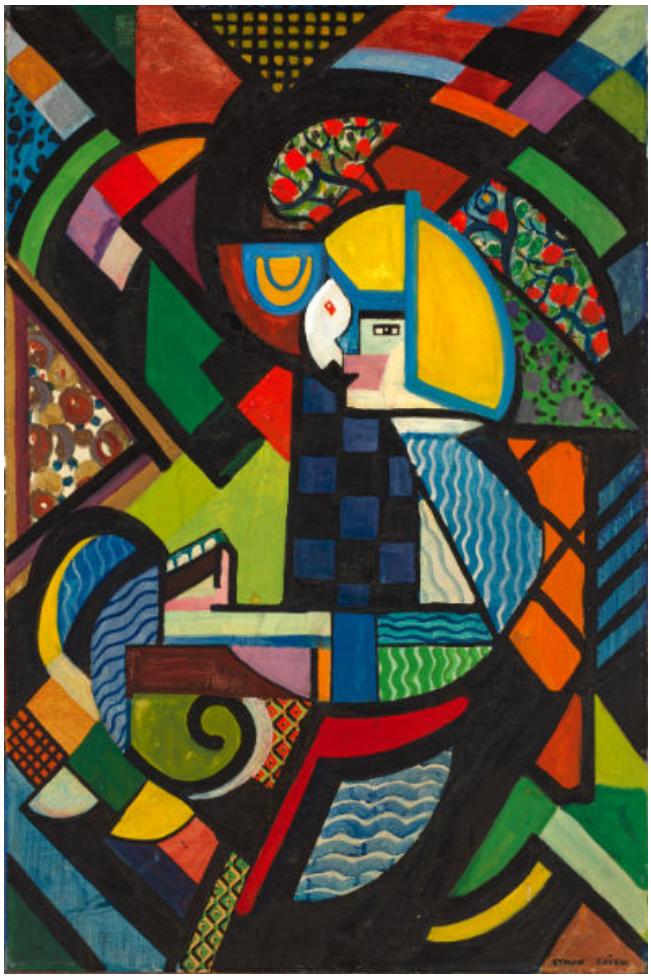
[4] https://github.com/thorn-oss/perception/
[5] Important Reminder: if you believe you've come across potential child sexual abuse material, DO NOT attempt to investigate it yourself. In the U.S., report it to the National Center for Missing and Exploited Children (https://www.missingkids.org/) or your local FBI Innocent Images Task Force. Outside the U.S., report potential CSAM to INHOPE (https://www.inhope.org/), the Virtual Global Taskforce (https://nationalcrimeagency.gov.uk/virtual-global-taskforce/), or Europol.
[6] An excellent introduction to perceptual hashes: https://www.phash.org/docs/
[7] https://en.wikipedia.org/wiki/Hamming_distance
[8] "Mean Square Error: Love It or Leave It?" https://ece.uwaterloo.ca/~z70wang/publications/SPM09.pdf
[9] https://github.com/thulab/DeepHash
[10] https://en.wikipedia.org/wiki/H._Lyman_Sa%C3%BFen
[11] https://www.si.edu/object/daughter-rocker%3Asaam_1967.6.4
[12] https://creativecommons.org/share-your-work/public-domain/cc0/

set the threshold similarity "distance" value to be 0.20, which should result in < 1% chance of failure to identify genuinely similar images. The Python3 code to compare pairs of images is quite short and shown here (as well as included as an Appendix 3 to this report).

```
$ cat compare_two_images.py
#!/usr/local/bin/python3

""" Compare two images using perceptual hashes """

import sys
from PIL import Image

# https://github.com/thorn-oss/perception/tree/master
from perception import hashers

if len(sys.argv) != 3:
     raise ValueError("Error: must supply two images as arguments")

image1 = str.rstrip(sys.argv[1])
image2 = str.rstrip(sys.argv[2])

img1 = Image.open(image1)
img2 = Image.open(image2)

# Perceptual hash static definitions --
https://github.com/thorn-oss/perception
hasher = hashers.PHash(hash_size=16)
my_hash1 = hasher.compute(img1)
my_hash2 = hasher.compute(img2)

# recommended threshold for PHash (hash_size=16), "expected false positive
rate of <1%"
threshold = 0.2

distance=hasher.compute_distance(my_hash1, my_hash2)
print("Perceptual hash distance is: "+str(distance))
print("Threshold is: "+str(threshold))
```

Let's now test the perceptual hashing routines on our Saÿen image, beginning with the images from the cover of the report (all images reduced in size for inclusion here):

daughter-in-a-rocker.jpg



daughter-in-a-rocker-grayscale.jpg

A sample run of our comparison code looks like:

```
$ ./compare_two_images.py daughter-in-a-rocker.jpg
daughter-in-a-rocker-grayscale.jpg
Perceputal hash distance is: 0.03125
Threshold is: 0.2
```

That sample run shows that the two images are reported to be similar, even though one had been substantially changed by being converted from color to 8-bit grayscale using Graphic Converter 11.[13] I think we can all agree with this assessment.

Now let's test some other subtly changed (and not-so-subtly changed) images.

[13] https://www.lemkesoft.de/en/products/graphicconverter/download

There are many ways you can change an image. Some obvious changes we tested included:

1) Does Image Format Matter? Testing the original image (in JPEG[14] format at 6.5MB) vs the original image in an alternative format (TIFF[15] format at 343.1MB), the phash distance was 0.00000 – **image format apparently had zero impact on the perceptual hash values.**

2) What If We Were to Scale the Size of the Image? Testing the original image in JPEG format vs a copy of the **image scaled to 3/4ths the original image size** (1496x2250px@300dpi), the phash distance was 0.00000 – **scaling also apparently had zero impact on the perceptual hash.**

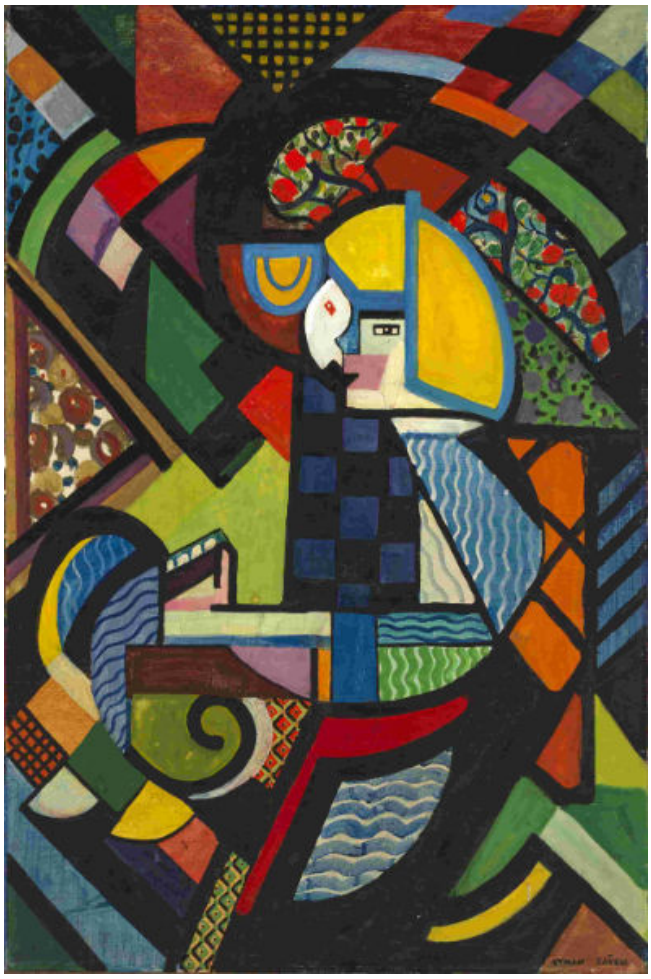
3) What If We Were to Change the Image Resolution? Testing the original 300dpi image vs a copy of the image that had been resaved at **just 100 dots per inch** (665x1000px@100dpi, 86KB), the phash distance was 0.00000 – **changing the image's resolution apparently had zero impact.**

On the following pages, we review some additional tests which had a greater than zero impact on the perceptual hash scores.

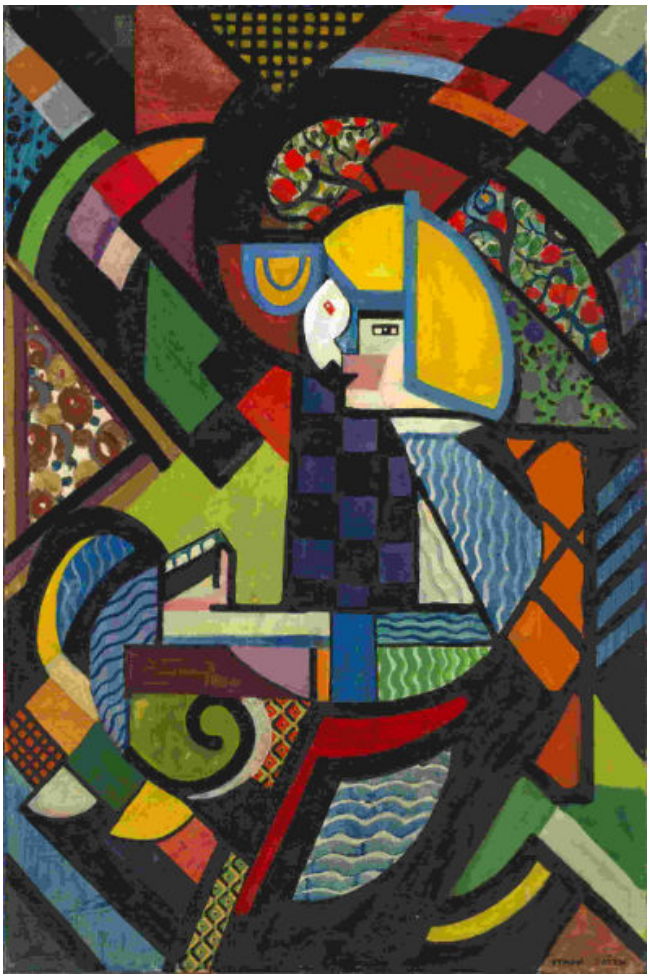
[14] https://en.wikipedia.org/wiki/JPEG
[15] https://en.wikipedia.org/wiki/TIFF

4) What If We Change the JPEG Quality Factor While Saving A Copy of the Image? Resaving the image with a JPEG quality factor of **10** (or even **1!**) resulted in a phash distance of 0.0078125 relative to the base image (we got the same phash score for both of our low quality JPEG factors). **Changing the JPEG quality factor, even to the lowest of levels, had negligible impact on the phash even though the low quality JPEG format images acquired visually-noticeable artifacts.**

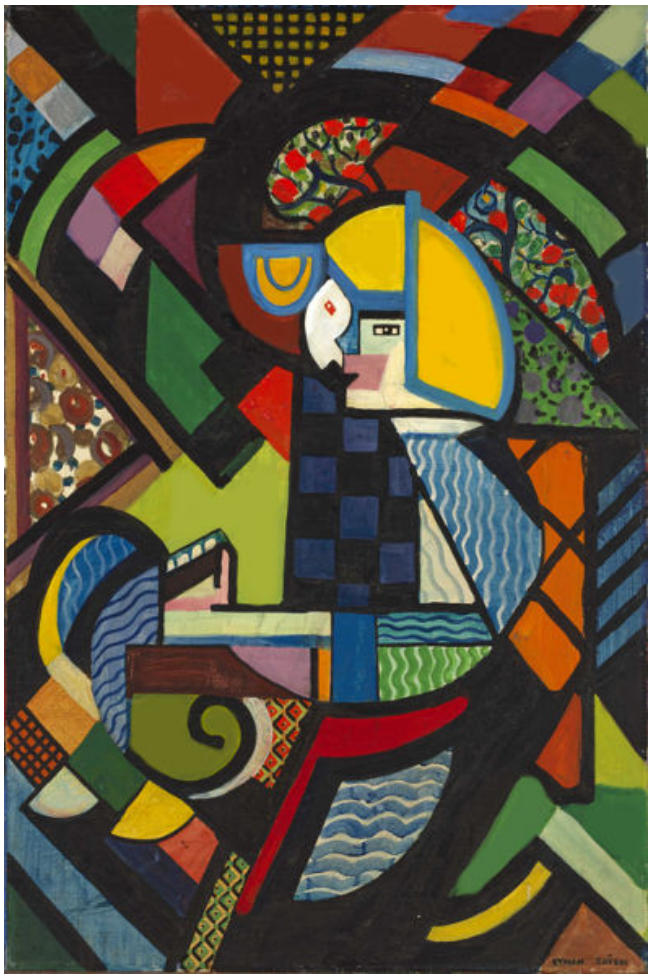
daughter-in-a-rocker-**jpeg-quality-factor-10**.jpg



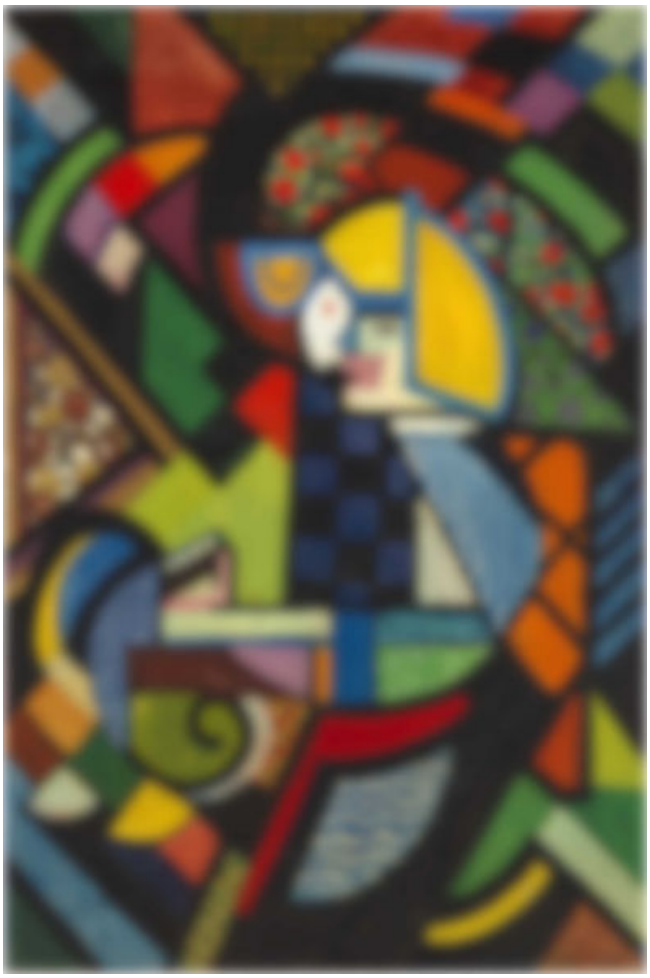
daughter-in-a-rocker-**jpeg-quality-factor-1**.jpg



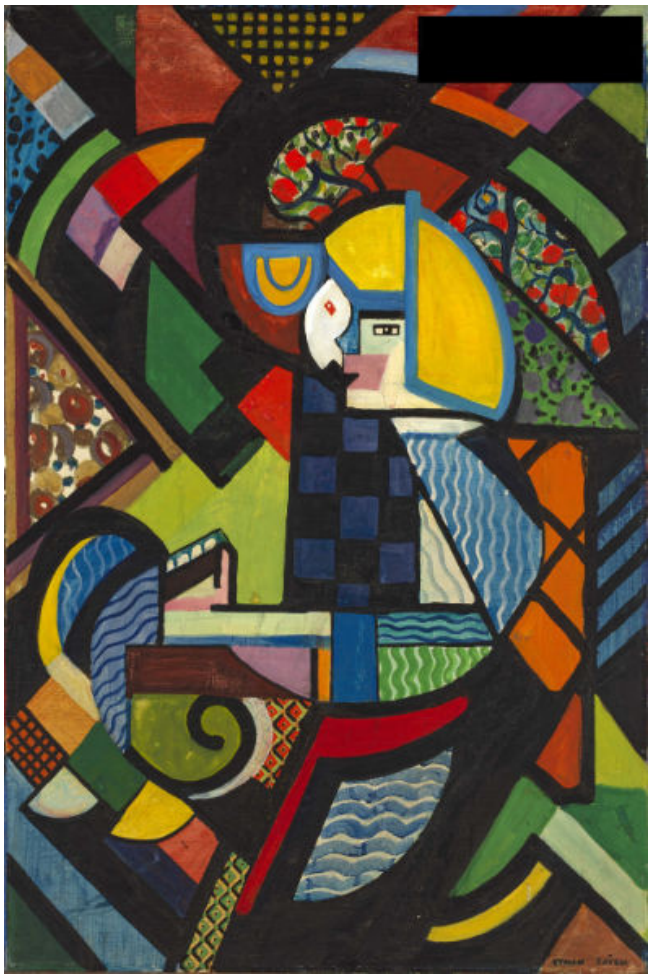
5) Orig. vs a copy after having **"streakier" color blocks "filled in," phash dist=0.015625**



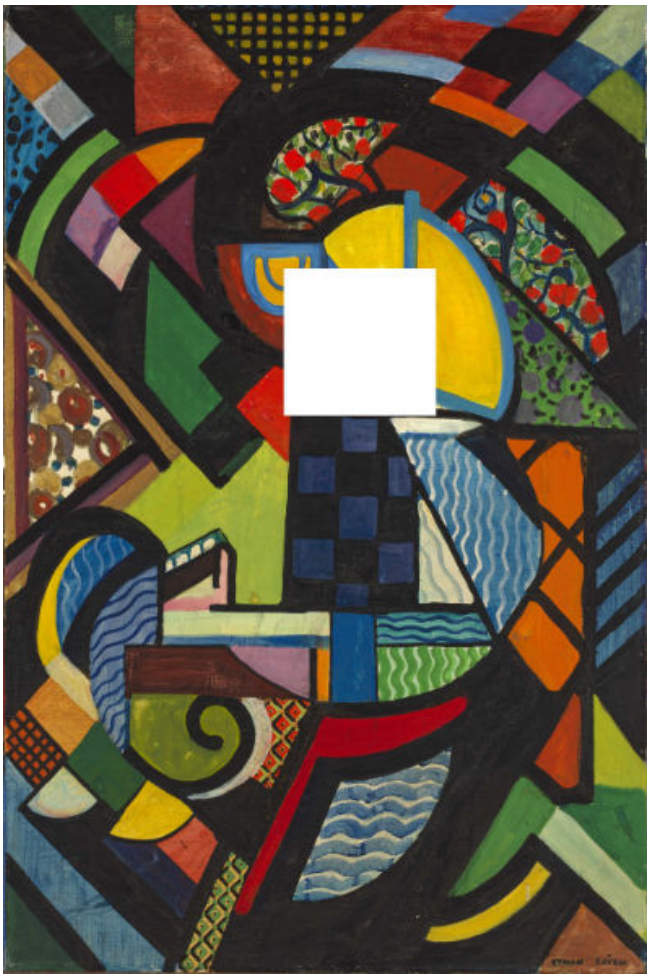
6) Orig. vs. copy after **a Gaussian blur (set at a factor of 20), phash dist=0.0390625**



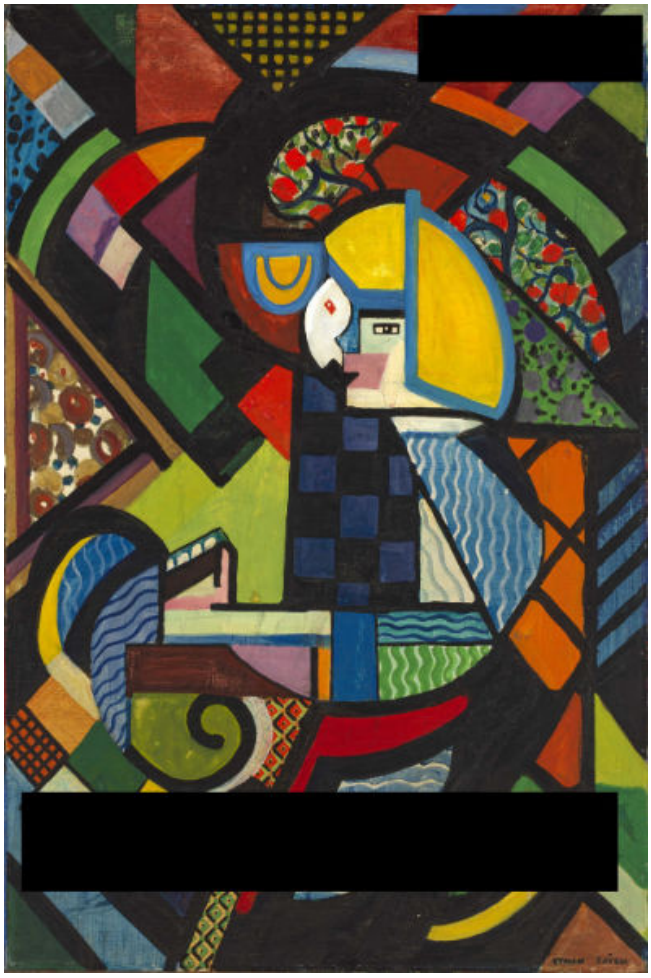
7) Orig. vs. copy w/**black box added NE corner: phash dist=0.0703125**



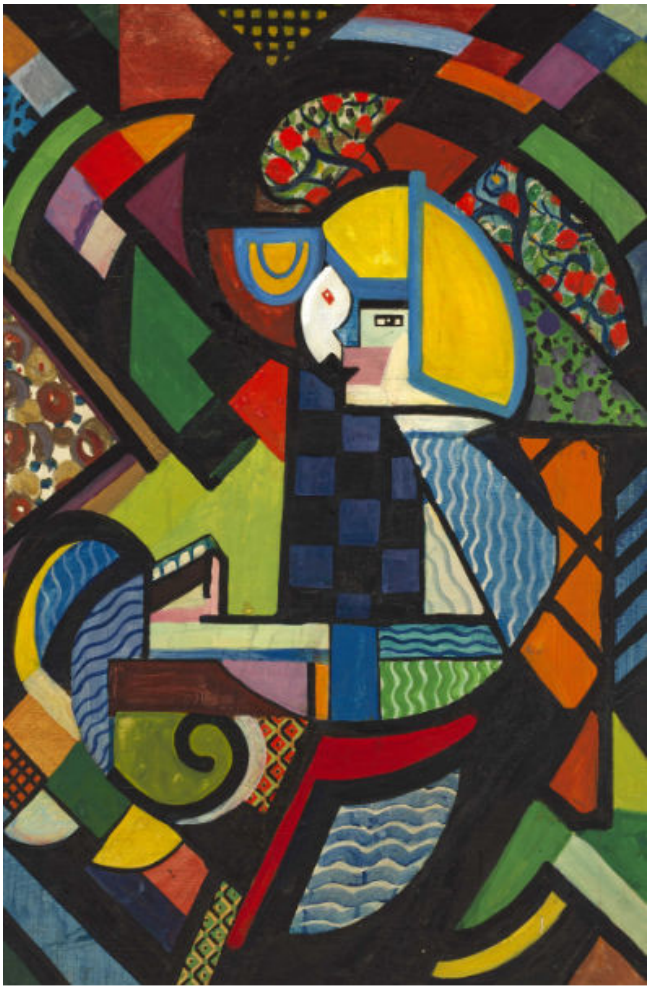
8) Orig. vs. a copy w/**white box in center: phash dist=0.140625**



9) Orig. vs. a **copy with 2 black boxes added: phash distance=0.171875**



10) Orig. vs a **copy cropped on all 4 sides: phash distance=0.1953125**

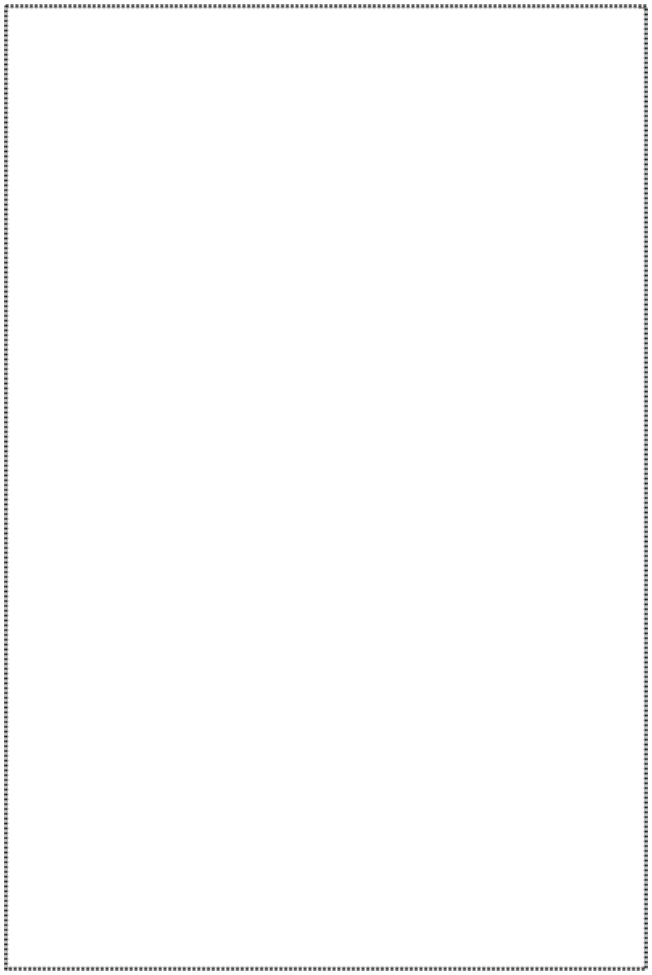


All of the above were found to be perceptually "similar" (perceptual hash distance <=0.2).

Some other changes we tried result in "NOT similar" perceptual hash distances.

11) Original image vs a copy **rotated 90 degrees to the right:** phash dist=**0.4765625 (exceeds our 0.2 threshold)**

12) Orig. vs a **white panel** (dots to show perimeter of "image") : **phash distance=0.53125 (exceeds threshold)**



13) Orig. vs. a **color inverted copy: phash distance=0.9921875 (exceeds threshold)**

Clearly our perceptual hash function CAN spot "similar" images while also detecting "material changes" to the base image.

Let's continue on by trying an actual screenshot-based example.

Part III: Screenshots of Gambling-Related Domain Names: A Case Study

"Dulce periculum"
[Danger is sweet]

"Cui amat periculum in illo peribit"
[Whoever loves danger will perish by it]

Gambling-Related Domains

Gambling is popular around the world, but online gambling is an area that's often heavily regulated (or banned entirely). Why is regulation required?

- Many people may buy a lottery ticket, play cards socially, or visit Las Vegas on vacation to gamble and see the sights, setting a reasonable budget that they can afford to lose and stopping if/when they hit that limit. For those people, gambling is a form of entertainment, and they can "take it or leave it." Others may be prone to problem gambling, compulsively losing money they may need to pay the rent or buy groceries[16]. Readily-accessible online casinos can facilitate those sort of self-destructive behaviors.

- Unlicensed and unregulated online gambling operations may be controlled by organized crime[17] or less-than-trustworthy parties. Some of those operators may cheat customers, run rigged games, abscond with customer deposits, or refuse to pay earned winnings. If/when that happens, what legal recourse would a defrauded party have against an anonymous online-but-offshore casino operator?

- The legal bricks-and-mortar gaming industry may[18], or may not[19], have a positive local economic impact. Online gambling may undercut and displace local gambling venues. If that happens, state and local tax revenues may be reduced[20], and economic development may be negatively impacted, killing local gambling-related jobs, reducing gambling-related travel/tourism to the area, and cutting public-benefit tax revenue income.

---

[16] If you think you or someone you know may have a gambling problem, consider contacting Gamblers Anonymous at https://www.gamblersanonymous.org/ga/ or a state-affiliated problem gambling support organization: https://www.ncpgambling.org/help-treatment/help-by-state/
[17] "Italian Mafia Bets on Illegal Online Gambling," https://www.occrp.org/en/daily/13985-italian-mafia-bets-on-illegal-online-gambling
[18] "National Economic Impact of the U.S. Gaming Industry," https://www.americangaming.org/resources/economic-impact-of-the-u-s-gaming-industry-2/
[19] "Casinos and Regional Economies: Has the Game Changed?" https://www.richmondfed.org/publications/research/economic_brief/2022/eb_22-28
[20] "Does Internet Gambling Strengthen the U.S. Economy? Don't Bet On It," https://www.repository.law.indiana.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1289&context=fclj at pp. 117

- Online gaming may represent a conduit for "laundering" money.[21]

- Minors may exploit the anonymity of online casinos to wager before they're of legal age to do so.[22]

Because gambling is subject to widespread regulation, search engines and social media platforms routinely limit both paid advertising and other promotional options for online gambling sites:

- Google's Advertising Policy on Gambling and Games[23]
- Bing's Gambling and Lotteries Policies[24]
- Meta's Online Gambling and Gaming Policy[25]
- Twitter's Gambling Content Policy[26]

The result of restrictions associated with those and similar policies is that casino site owners and other gambling operators may engage in atypical domain-related practices, including registering unusually-large numbers of domains. They may do so for a variety of purposes:

- **Some May Use Large Numbers of Domains in an Effort to Keep Any Individual Domain from "Running Too Hot"** – Law enforcement agencies don't have the resources to go after each and every offense they uncover. Agencies may have undisclosed offense "thresholds" that may need to be exceeded before criminal behavior gets scored as "bad enough to merit attention," or agencies may routinely prioritize the "biggest" or "most notorious" offenders first. That's one reason why smart criminals may divide their activity over dozens of seemingly-unrelated sites: by doing so, offenders may hope that they can appear to be "insignificant" and "unworthy" of law enforcement targeted action.

  How can LEOs overcome this approach? Simple: they need to "connect the dots" and recognize that there may be a set of sites under common ownership and control, and when those sites are taken together AS A GROUP, they may "add up" to an organization that (in aggregate) does merit careful scrutiny.

- **Affiliate Marketing** – Some affiliate marketing programs track affiliates by routing web visits through automatically-redirecting "front" domains – visits that come "through" one or more specific "front" domains get attributed to affiliate X, while visits that come "through" a different set of "front" domains get attributed to affiliate Y, etc.

---

[21] "Money laundering through the gambling industry," https://baselgovernance.org/sites/default/files/2022-09/QG28%20gambling.pdf

[22] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2533814/
[23] https://support.google.com/adspolicy/answer/6018017?hl=en
[24] https://about.ads.microsoft.com/en-us/policies/restricted-categories/gambling-and-lotteries
[25] https://transparency.fb.com/policies/ad-standards/content-specific-restrictions/gambling/
[26] https://business.twitter.com/en/help/ads-policies/ads-content-policies/gambling-content.html

There are many casino affiliate marketing programs that may use this or similar approaches. Sites listing a number of gambling-related affiliate programs can readily be found by simply making an online search for “affiliate programs gambling”.

- **Attempting to Overcome Domain-Based Blocklisting** – If a national government requires Internet Service Providers (ISPs) to block attempts to access a specific unlicensed online gambling-related domain, a casino operator may try to circumvent that filtering by offering other new (and temporarily unblocked) alternative domains, with all of those domains eventually channeling visitors to the same ultimate destination. Of course, when domain owners try doing this sort of "end run," governments may respond by updating their ISP-must-block lists to include the new filter-evasive domains, which may in turn trigger the casino operator to register another round of filter-evasive domains, etc., in what may ultimately become a pointless test of wills (and frustrating for ISPs).

  One company that tracks gambling-related blocklist entries, Vixio.com, asserts that there were "more than 110,000 blocklist entries in monitored countries as of December 2020."[27] There's no reason to believe the number of blocked gambling-related domains has diminished since that time.

- **Bad Actors May Register Large Numbers of Domains for Gambling-related Malware[28] or Phishing Attacks.[29]** In this scenario, bad actors may try to leverage popular interest in gambling sites to drop malware on visitors and/or to "socially engineer" visitors into disclosing credit card or other financial information.

  Registering large numbers of gambling-related domains increases the likelihood that Internet users may inadvertently end up visiting look-alike (and potentially dangerous) gambling-related sites, while also increasing the likelihood that antivirus or anti phishing vendors may overlook and fail to blocklist at least some of them.

- **Large Numbers of Domains May Also Be Registered for Ad-Revenue-Related Purposes, Domain Speculation or Other Mundane/Lawful Purposes.** If a domain speculator notices many people are interested in casino-related domains, they may speculatively buy casino-related domains for resale, or as a way of attracting eyeballs for PPV (pay-per-view) ads.

---

[27] https://vixio.com/gamblingcompliance/blocklist-monitoring-tracker/
[28] "Operation Earth Berberoka: An Analysis of a Multivector and Multiplatform APT Campaign Targeting Online Gambling Sites," https://documents.trendmicro.com/assets/white_papers/wp-operation-earth-berberoka.pdf
[29] "How Phishing Impacts the Online Gambling Industry," https://www.sportsbetting3.com/research/how-phishing-impacts-the-online-gambling-industry#incidents

So How Many Online Gambling-Related Domains Are Out There?

To the best of our knowledge, there's no single comprehensive list of gambling-related domains. While reviewing a copy of the Tranco Top domains list[30] (which now includes data from DomainTools[31]), we noticed an unusually-large number of casino-related domains.

Specifically, if we look at the full Tranco list for October 28th, 2022[32] (which had 7,240,113 total entries), we discover well over 50,000 likely gambling-related domains:

```
$ wget "https://tranco-list.eu/download/W9Y39/full"
$ awk -F',' '{print $2}' < full > tranco_W9Y39.txt
$ egrep '(casino|kasino|cazino|kazino|slots|poker|jackpot)'
tranco_W9Y39.txt > casino.txt
$ wc -l casino.txt
54460 casino.txt
```

That seemed like an awfully lot of gambling-related domains to us.

We also checked DNSDB Flexible Search[33] to identify gambling-related domains that have been seen by DNSDB sensors. In this case, we began by running Flexible Search queries for "A" records (IPv4 address records), looking for some (but obviously not all) gambling-related terms:

```
$ dnsdbflex --regex '(casino|kasino|cazino|kazino|slots|poker|jackpot)'
-t A -l0 -A7d -j > dnsdbflex-output.jsonl
$ wc -l dnsdbflex-output.jsonl
1000000 dnsdbflex-output.jsonl
```

Decoded, those commands ask for domains where:

- The RRname ("left hand side" of a DNS record) contains any one of various gambling keywords {casino, kasino, cazino, kazino, slots, poker, or jackpot},
- The record type is "A" (that's the dash tee space capital A option)
- The record was seen at least sometime within the last 7 days (dash capital A seven dee)
- “Dash ell zero" means give me up to a million of those hits in JSON Lines format (dash j).

Since we got a full million results from our first query, we did a second query (offset by our already-found-million-results) to see if we can get still further results – and we found some:

---

[30] https://tranco-list.eu/

[31] https://www.domaintools.com/resources/blog/mirror-mirror-on-the-wall-whos-the-fairest-website-of-them-all/

[32] https://tranco-list.eu/list/W9Y39/1000000

[33] Flexible Search allows us to use regular expressions to find matches in DNSDB. For details, see www.farsightsecurity.com/assets/media/download/DNSDB_Flexible_Search_Intro.pdf

```
$ dnsdbflex --regex '(casino|kasino|cazino|kazino|slots|poker|jackpot)'
-t A -l0 -O1000000 -A7d -j >> dnsdbflex-output.jsonl
$ wc -l dnsdbflex-output.jsonl
1172797 dnsdbflex-output.jsonl
```

Now let's run another query to check for "CNAME" records (domain nicknames pointing at other domain names), tacking them on to the same file of results:

```
$ dnsdbflex --regex '(casino|kasino|cazino|kazino|slots|poker|jackpot)'
-t CNAME -l0 -A7d -j >> dnsdbflex-output.jsonl
$ wc -l dnsdbflex-output.jsonl
1458826 dnsdbflex-output.jsonl
```

And finally, let's finish up by checking for any "AAAA" records (IPv6 address records):

```
$ dnsdbflex --regex '(casino|kasino|cazino|kazino|slots|poker|jackpot)'
-t AAAA -l0 -A7d -j >> dnsdbflex-output.jsonl
$ wc -l dnsdbflex-output.jsonl
1545958 dnsdbflex-output.jsonl
```

Now that we've successfully found over 1.5 million potential hits, let's: (a) condense those down to just base (registrable) domain names with jq[34] plus a small script shown in Appendix 1 called `2nd-level-dom-large`, (b) then we'll sort and deduplicate, and (c) lastly, we'll double check to ensure relevant keywords are still present in the base domain (in some cases, our keywords may have only been present in the now-discarded hostname part):

```
$ jq -r '.rrname' < dnsdbflex-output.jsonl | 2nd-level-dom-large | sort
-u | egrep '(casino|kasino|cazino|kazino|slots|poker|jackpot)' >
dnsdbflex-base-domains.txt
$ wc -l dnsdbflex-base-domains.txt
317720 dnsdbflex-base-domains.txt
```

We freely concede that some of those "hits" may be only peripherally-related to actual gambling, or may actually be for things like anti-gambling rehabilitation services, but we believe that most of those hits are genuinely true to their nominal purpose.

Where Are All These Gambling-Related Domains Hosted?

We can take the domains we've found in dnsdbflex and look them up in DNSDB using dnsdbq. We'll do that by editing our domain list to create lines that look like:

[34] https://stedolan.github.io/jq/

```
dnsdbq -r 0-casino.com/A -l0 -j -A7d -a > batch-output.txt
dnsdbq -r 0-casino.info/A -l0 -j -A7d -a >> batch-output.txt
dnsdbq -r 0-casinos.org/A -l0 -j -A7d -a >> batch-output.txt
dnsdbq -r 0-poker.biz/A -l0 -j -A7d -a >> batch-output.txt
[etc]
```

After running 317,720 queries of that sort, we were left with a 100MB output file of results. You might conceptually imagine output that simply looks like…

```
domainname, IPv4 address
```

but that's not actually what we got. For example:

- Our output is in JSON Lines[35] format, not plain text CSV format
- A single DNSDB "A" record or "AAAA" query might return multiple IPs
- A RRname might have multiple results (each of which might have multiple IPs)
- We might see "special" IP values, or
- There might be occasional problems resolving IPs to autonomous system numbers

Let's dig into that a bit more.

Simple Result: A sample domain with just **one result** (and with **one resource record** for that result) looks like the following when the JSON Lines is "pretty printed" with jq:

```
{
  "count": 76,                    <-- number of cache misses seen
  "time_first": 1645361436,       <-- time first seen in Un*x ticks
  "time_last": 1667047559,        <-- time last seen in Un*x ticks
  "rrname": "0-casino.com.",      <-- name of the domain
  "rrtype": "A",                  <-- resource record type
  "bailiwick": "0-casino.com.",   <-- where in the DNS "tree" this was seen
  "rdata": [                      <-- IPv4 address or address for this domain
      "99.83.248.67"
  ],
  "dnsdbq_rdata": {
      "99.83.248.67": {           <-- IP address for this ASN lookup
      "asinfo": {
      "as": [
      16509                       <-- current ASN info for that IP address
      ],
      "cidr": "99.83.240.0/20"    <-- CIDR block containing that IP address
      }
    }
  }
}
```

[35] https://jsonlines.org/

Results with Multiple RRs In A Single RRset: A single result may also be an RRset with ***more than one resource record***. In that case, the `rdata` array will have multiple IP addresses, and the `dnsdbq_rdata` AS information will have multiple entries, one for each IP address reported:

```
$ jq '.' batch-output.txt
[...]
{
  "count": 285,
  "time_first": 1648027701,
  "time_last": 1667557367,
  "bailiwick": "0-casinos.org.",

  "rdata": [
      "104.21.6.251",
      "172.67.135.134"
  ],
  "dnsdbq_rdata": {
      "104.21.6.251": {
      "asinfo": {
      "as": [
      13335
      ],
      "cidr": "104.21.0.0/20"
      }
      },
      "172.67.135.134": {
      "asinfo": {
      "as": [
      13335
      ],
      "cidr": "172.67.128.0/20"
      }
      }
  }
}
[...]
```

Multiple Results for a Single RRname Over Time: Other times, a single domain may have had multiple results returned from DNSDB. For example, since we track unique combinations of (RRname, RRtype, Bailiwick, Rdata and data source (sensors vs zone files)), if the domain's hosting moved from one IP address to another, we may see multiple results for that domain name.

For example (manually highlighting some values but NOT "pretty-printing" this example):

{"count":192,"time_first":**1653897820**,"time_last":1667290049,"rrname":"**00-casino.com.**"
, "rrtype":"A","bailiwick":"00-casino.com.","rdata":["**154.80.242.178**"],

"dnsdbq_rdata":{"154.80.242.178":{"asinfo":{"as":[**134175**],"cidr":"154.80.224.0/19"}}}
}

{"count":3,"time_first":**1667376452**,"time_last":1667549226,"rrname":"**00-casino.com.**",
"rrtype":"A","bailiwick":"00-casino.com.","rdata":["**154.93.184.156**"],
"dnsdbq_rdata":{"154.93.184.156":{"asinfo":{"as":[**134548**],"cidr":"154.93.184.0/22"}}}
}

Results That Resolve to a Special IP Address: We also wanted to be sure to call out the fact that some of the domains in our file appear to have been pointed at "special" IP values such as:

- `127.0.0.1` `<-- localhost`[36]
- `127.0.53.53` `<-- a special ICANN name collision signaling value`[37]
- `10.10.10.10` `<-- A commonly used RFC1918 address`[38]
- `0.0.0.0` `<-- "Unavailable address"`[39]

Results That Can't Be Mapped to a Corresponding ASN: Some domains may also not map to an ASN, either because the values are special IP addresses (as just mentioned), or because the IP addresses aren't in the current BGP routing table used by the Oregon Routeviews project.[40] Our plan for handling all of this is to:

- Remove any results that point at one of the special values just mentioned:

```
$ egrep -v "(127\.0\.0\.1|127\.0\.53\.53|10\.10\.10\.10|0\.0\.0\.0)"
batch-output.txt > batch-output-2.txt

$ wc -l batch-output.txt batch-output-2.txt
   368380 batch-output.txt
   365861 batch-output-2.txt
```

- Remove any results where an ASN lookup timed out or otherwise failed:

```
$ egrep -v "(\:$|Host name lookup failure)" batch-output-2.txt >
batch-output-3.txt

$ wc -l batch-output-2.txt batch-output-3.txt
   365861 batch-output-2.txt
   365168 batch-output-3.txt
```

- Keep the results where ASN information WAS returned:

```
$ grep asinfo batch-output-3.txt > batch-output-4.txt
```

[36] https://en.wikipedia.org/wiki/Localhost
[37] https://www.icann.org/en/announcements/details/icann-approves-name-collision-occurrence-management-framework--special-ip-address-12705353-alerts-system-administrators-of-potential-issue-1-8-2014-en
[38] https://datatracker.ietf.org/doc/html/rfc1918
[39] https://en.wikipedia.org/wiki/0.0.0.0
[40] dnsdbq uses https://www.routeviews.org/routeviews/ data for IP to ASN mapping by default

```
$ wc -l batch-output-3.txt batch-output-4.txt
   365168 batch-output-3.txt
   364644 batch-output-4.txt
```

- Condense the results into a more compact format and de-aggregate the RRsets:

```
$ jq -r '"\(.rrname), \(.time_last), \(.count), \(.rdata[]),
\(.dnsdbq_rdata[].asinfo.as[])"' < batch-output-4.txt | sort -u >
batch-output-5.txt

$ wc -l batch-output-4.txt batch-output-5.txt
   364644 batch-output-4.txt
   546548 batch-output-5.txt

$ more batch-output-5.txt
0-casino.com., 1667042552, 2, 104.247.82.50, 206834
0-casino.com., 1667047559, 76, 99.83.248.67, 16509
0-casino.info., 1667594504, 16000, 129.121.24.7, 62729
0-casinos.org., 1667050123, 54, 104.21.6.251, 13335
0-casinos.org., 1667050123, 54, 172.67.135.134, 13335
0-poker.biz., 1667557367, 285, 188.165.140.64, 16276
00-00midnightcasino.com., 1667372689, 232, 104.21.72.147, 13335
00-00midnightcasino.com., 1667372689, 232, 172.67.151.98, 13335
[etc]
```

- In the case of RRnames returning multiple results, take the most recently seen of those

```
$ sort -t, -k1,1 -k2,2nr batch-output-5.txt | sort -su -t, -k1,1 >
batch-output-6.txt

$ wc -l batch-output-5.txt batch-output-6.txt
  546548 batch-output-5.txt
  276400 batch-output-6.txt
```

- Now tally up results by ASN, returning ASNs with 250 or more gambling-related domains:

```
$ cut -d, -f5 batch-output-6.txt | sort -n | uniq -c | sort -nr >
top-asns.txt

$ wc -l top-asns.txt
2330 top-asns.txt

$ more top-asns.txt
 73350  13335     <-- AS13335      Cloudflare (73,350/276,400*100=26.5%)

 31950  16509     <-- AS16509      Amazon (31,950/276,400*100=11.5%)
```

21600	***396982***	***<-- AS396982***	***Google (21,600/276,400*100=7.8%)***
9044	***22612***	***<-- AS22612***	***Namecheap (9,044/276,400*100=3.2%)***
8877	***16276***	***<-- AS16276***	***OVH (8,877/276400*100=3.2%): 5 ASNs=52.2%***
6427	47846	<-- AS47846	SEDO
5487	15169	<-- AS15169	Google
5316	14061	<-- AS14061	Digital Ocean
3809	209242	<-- AS209242	Cloudflare
3570	61969	<-- AS61969	Team Internet (DE)
3542	204601	<-- AS204601	Zomro BV (NL)
3088	60819	<-- AS60819	Safenames Ltd (GB)
3087	24940	<-- AS24940	Hetzner (DE)
2651	8560	<-- AS8560	1&1 IONOS (SE)
2290	46606	<-- AS46606	Unified Layer
2033	197695	<-- AS197695	Reg.ru (RU)
1898	40034	<-- AS40034	Confluence Networks (VG)
1806	133618	<-- AS133618	Trellian Pty Ltd (AU)
1746	32244	<-- AS32244	Liquid Web LLC
1698	20857	<-- AS20857	Signet BV (NL)
1697	13008	<-- AS13008	Entain Services Austria GmbH (AT)
1662	58061	<-- AS58061	Scalaxy BV (NL)
1550	63949	<-- AS63949	Linode LLC
1516	31624	<-- AS31624	Verotel Intl BV (NL)
1507	19324	<-- AS19324	Dosarrest Internet Security Ltd
1438	19574	<-- AS19574	Corporation Service Co
1317	203	<-- AS203	Lumen (Centurylink/Level3)
1263	58182	<-- AS58182	Wix.com (IL)
1205	14618	<-- AS14618	Amazon
1131	19871	<-- AS19871	Network Solutions LLC
991	39570	<-- AS39570	Loopia AB (SE)

```
978  51747    <-- AS51747     Internet Vikings Intl AB (SE)
953  36351    <-- AS36351     Softlayer Tech Inc
872  47583    <-- AS47583     Hostinger Intl Ltd (CY)
861  60906    <-- AS60906     Playdom BV (CW)
779  54113    <-- AS54113     Fastly Inc.
769  19551    <-- AS19551     Incapsula Inc
745  20738    <-- AS20738     Host Europe GmbH (DE)
742  42708    <-- AS42708     Portlane.com/glesys.se (SE)
711  35041    <-- AS35041     Binero AB (SE)
702  200719   <-- AS200719    Miss Hosting AB (SE)
691  26347    <-- AS26347     New Dream Network LLC
679  48357    <-- AS48357     K4X (EE)
678  60781    <-- AS60781     LeaseWeb Netherlands BV (NL)
667  6724     <-- AS6724      Strato AG (DE)
665  3842     <-- AS3842      InMotion Hosting Inc
663  20473    <-- AS20473     The Constant Co LLC
643  48287    <-- AS48287     RU-Center (RU)
633  48635    <-- AS48635     CLDIN BV (NL)
624  398101   <-- AS398101    Godaddy.com LLC
598  29873    <-- AS29873     Newfold Digital Inc
555  43338    <-- AS43338     TSG Interactive Svcs Ltd (IM)
536  198610   <-- AS198610    Beget LLC (RU)
510  12996    <-- AS12996     Domeneshop AS (NO)
496  51468    <-- AS51468     One.com A/A (DK)
489  32787    <-- AS32787     Akamai
486  18779    <-- AS18779     EGI Hosting
482  54600    <-- AS54600     PEG Tech Inc
454  7506     <-- AS7506      GMO Internet Inc (JP)
443  32475    <-- AS32475     Singlehop LLC
```

```
440  31034    <-- AS31034     Aruba S.p.A. (IT)
433  59253    <-- AS59253     Leaseweb (SG)
433  399522   <-- AS399522    The Producers Inc
429  12552    <-- AS12552     GlobalConnect AB (SE)
427  14537    <-- AS14537     Continent 8 LLC
423  15598    <-- AS15598     IP Exchange GmbH (DE)
419  29169    <-- AS29169     Gandi SAS (FR)
415  131965   <-- AS131965    Xserver Inc (JP)
400  132203   <-- AS132203    Shenzhen Tencent (CN)
384  13768    <-- AS13768     Aptum Technologies (CA)
377  48854    <-- AS48854     team.blue Denmark A/S (DK)
359  31727    <-- AS31727     Node4 Limited (GB)
347  49981    <-- AS49981     WorldStream BV (NL)
346  40244    <-- AS40244     Turnkey Internet Inc
345  34788    <-- AS34788     Neue Medien Muennich GmbH (DE)
340  42745    <-- AS42745     Safe Value Ltd (SC)
338  206281   <-- AS206281    Stichting DIGI NL (NL)
334  9009     <-- AS9009      M247 Europe SRL (RO)
328  53755    <-- AS53755     Input Output Flood LLC
319  54489    <-- AS54489     CoreSpace Inc
314  12859    <-- AS12859     BIT BV (NL)
312  46844    <-- AS46844     Sharktech
308  50474    <-- AS50474     O2SWITCH SARL (FR)
304  38880    <-- AS38880     Micron21 Datacentre Pty Ltd (AU)
303  51167    <-- AS51167     Contabo GmbH (DE)
292  34922    <-- AS34922     NetNames (GB)
290  32748    <-- AS32748     Steadfast
290  134548   <-- AS134548    DXTL Tseung Kwan O Service (HK)
288  8075     <-- AS8075      Microsoft
```

```
288  28753    <-- AS28753     Leaseweb Deutschland GmbH (DE)

285  24611    <-- AS24611     Datacenter Luxembourg SA (LU)

275  15456    <-- AS15456     InterNetX GmbH (DE)

271  9123     <-- AS9123      TimeWeb Ltd. (RU)

267  15348    <-- AS15348     Tucows (CA)

259  20860    <-- AS20860     IOMart Cloud Services Ltd (GB)

254  29550    <-- AS29550     Simply Transit (GB)

251  8972     <-- AS8972      Host Europe GmbH (DE)

   [remaining ASNs have counts < 250]
```

Why might we want to know where domains are hosted? A few reasons might include:

- Some providers might be more willing to deal with problematic domains (if there are any) than other providers
- Some domains might be hosted by providers in locations where corruption is a problem, or where the authorities might have more pressing problems than taking action against problematic domains
- Breaking domains out by provider can sometimes be helpful when it comes to clustering badness, or documenting an intentional attempt at engineering takedown-resistance by diversifying a site's hosting to multiple providers.

Risk Scoring Our Gambling Domains

Speaking of ASN-based attributes, in an earlier research report, we discussed DomainTool risk scores and showed how risk scores could be compared on an ASN-by-ASN basis for select domains (our example for that earlier report focused on domains that begin with a digit).[41] See that report for a discussion of our approach and the code we used to produce "violin" risk plots.

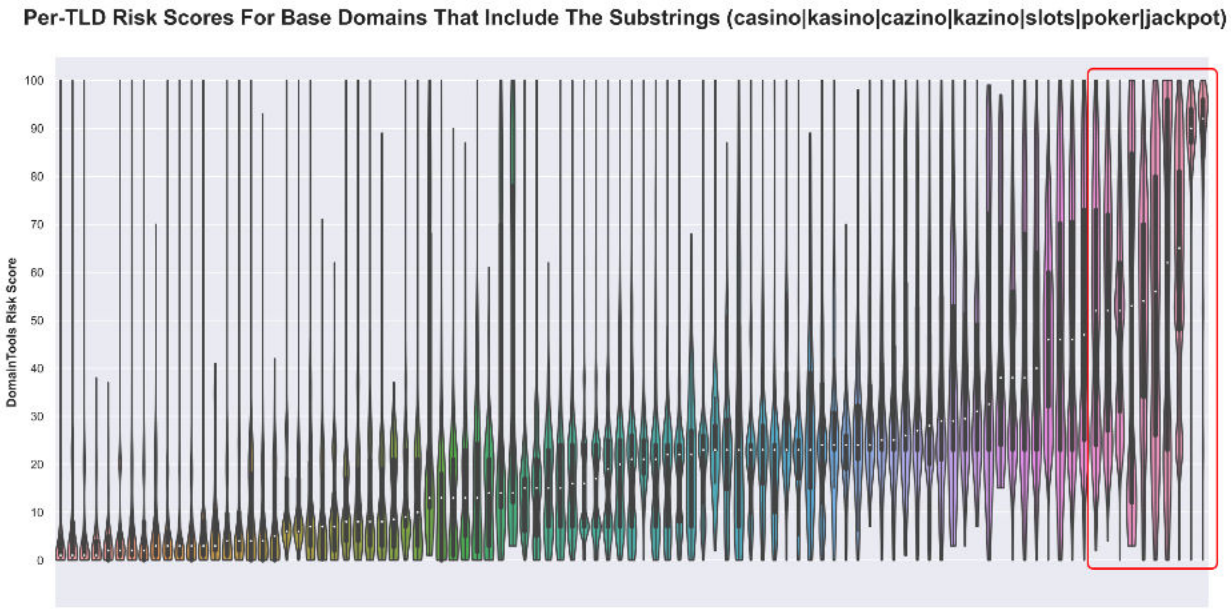
Today, we apply a similar approach to gambling-related domains.

We were curious: did some TLDs with gambling-related domains have a particularly high median DomainTools risk score? If so, was it the result of malware-related risk, phishing-related risk, spam-related risk, or simply the result of the current domain being close to infrastructure associated with other risky domains?

[41] www.domaintools.com/resources/white-papers/domains-that-begin-with-a-digit/

The first graph illustrates the distribution of risk scores across various top-level domains ("TLDs") we'd found with 100 or more gambling-related domains.[42] [Note that we do not expect you to be able to identify the various TLDs from the graph, this graph is just meant to give you a "shape of the graph" and a sense of the median risk score distribution.]

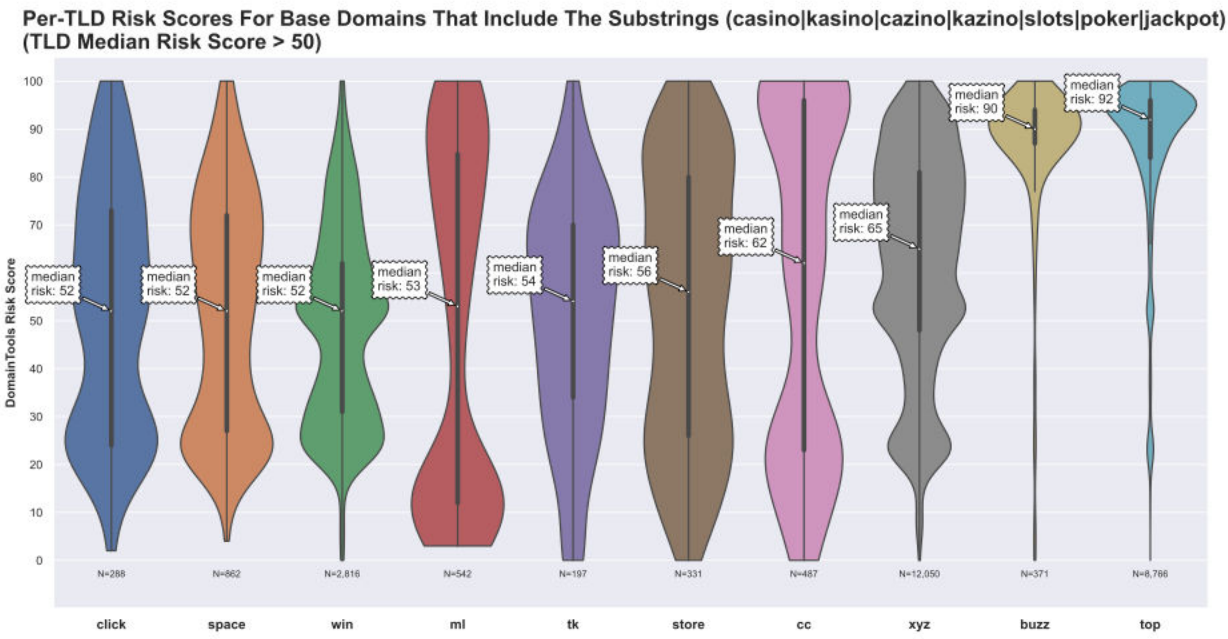
Median risk scores appear as a small white dot in the middle of each of these narrow "violins," with median values ranging from 1 (low risk) to 92 (high risk):



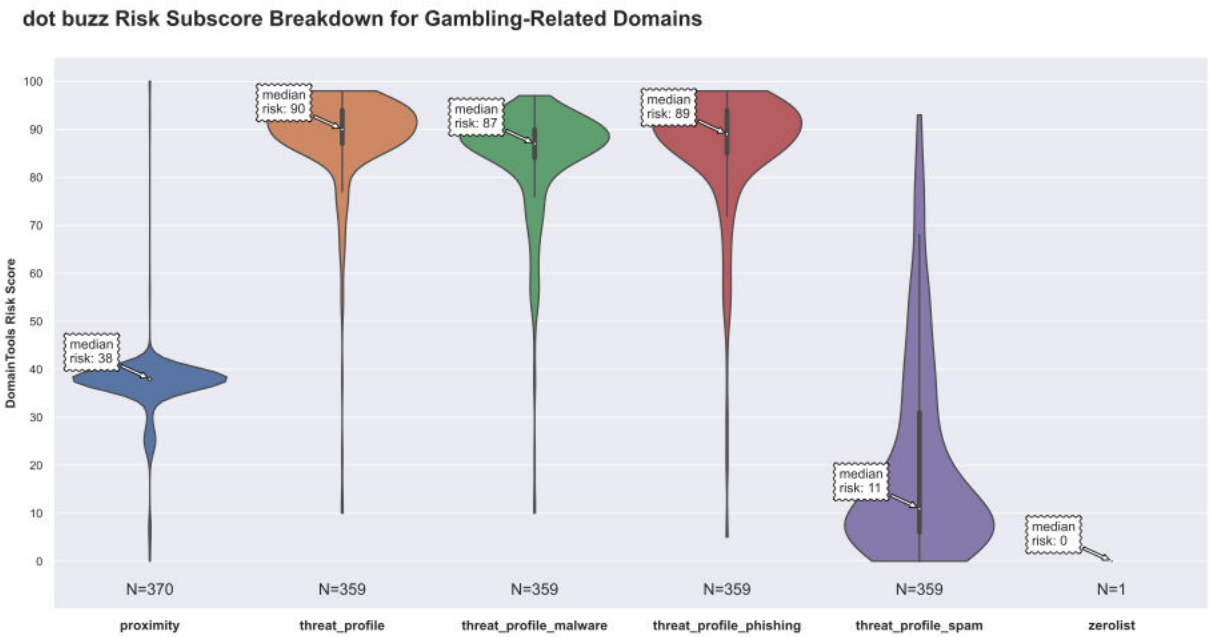
Obviously, most of those TLDs had median DomainTools risk scores well below even 50. Some TLDs, however, were higher.

For our 2nd plot, we wanted to drill in on JUST the TLDs that had a median DomainTools risk score for the TLD of more than 50. There were ten TLDs of that sort (highlighted with a thin red box in the above), and shown in more detail in this second graph:

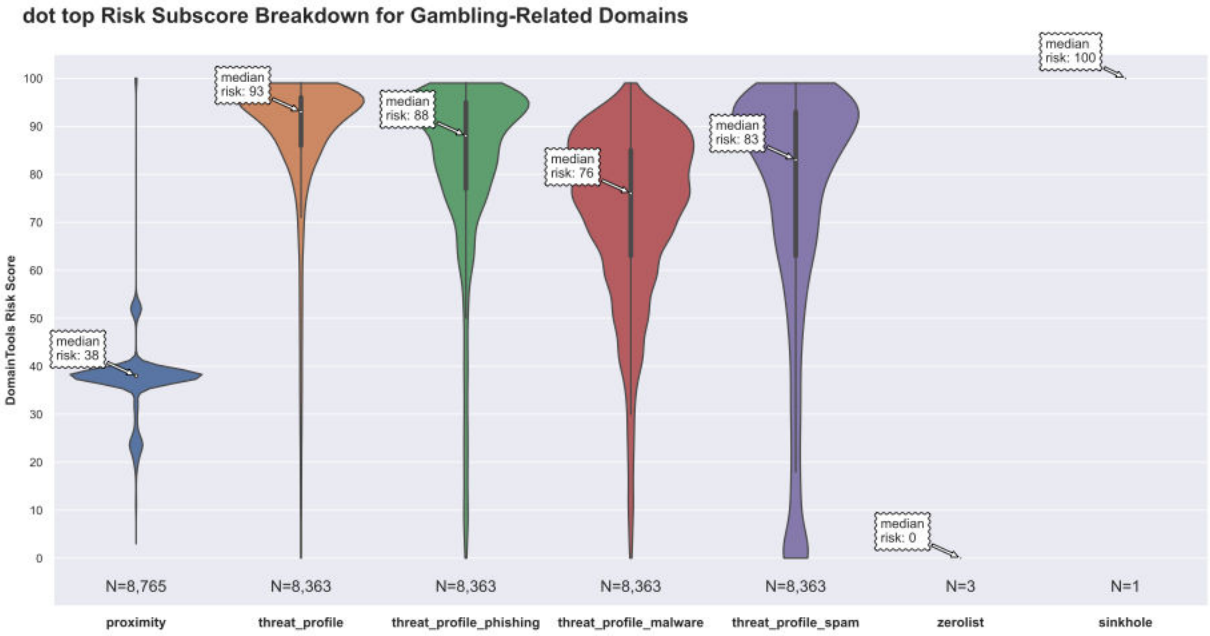
[42] For the purposes of this analysis, gambling-related domains were defined to be domains that included one of the substrings "casino," "kasino," "cazino," "kazino," "slots," "poker," or "jackpot" in the base (registerable) part of the domain name.



Two of those TLDs (those on the far right side of this 2nd graph) stand out for having exceptionally high median risk scores for gambling-related domains: those two TLDs are dot buzz and dot top. What risk subscore drove those elevated values for those two TLDs? Fortunately, we developed the ability to produce per-subscore violin plots in the process of researching our earlier report. For the dot buzz gambling-related domains, the risk appears to be driven by a mix of malware and phishing (spam-related risk and proximity subscores were low).



For dot top, phishing, spam and malware scores were all above the normal threshold of 70, while proximity was low.



Manually reviewing some domains from the two TLDs, we note that many of the higher-scoring domains were of the form *gambling_related_name-<random-looking two letter suffix>.tld*

Introductory Case Study: Some “Gambling”-Related Domains Routed by AS48357 (K4X)

The top 20 domains (and counts) associated with IPs announced by K4X (Estonia) are:

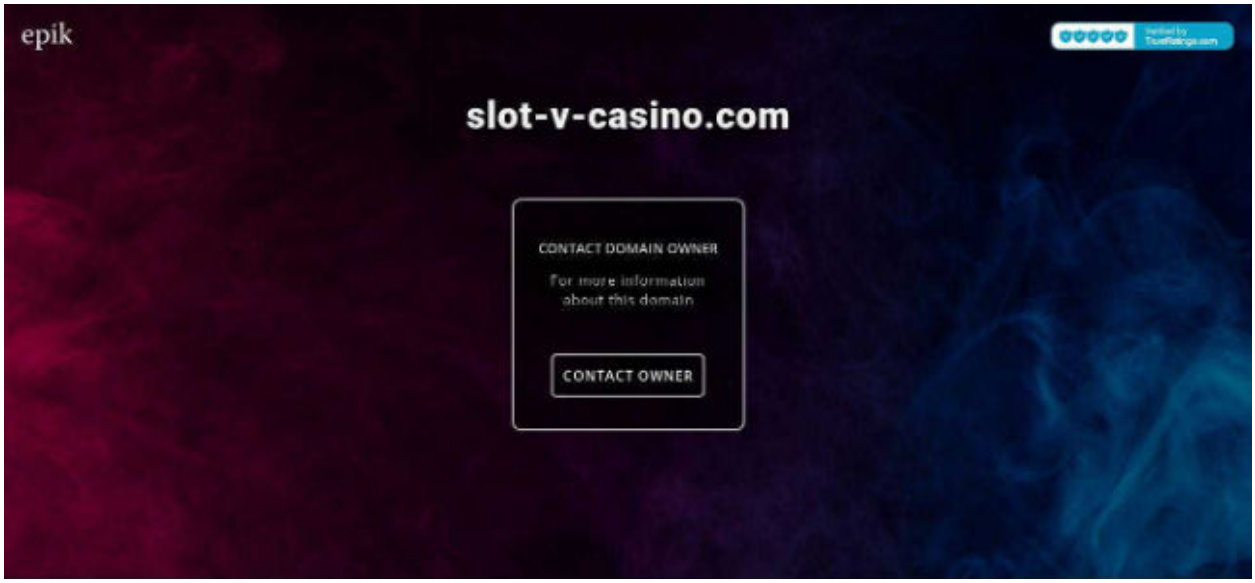
```
$ grep 48357 batch-output-6.txt | sort -k3,3nr | cut -d, -f 1,3 | head -20
> k4x.txt
joycasino-ym.top., 483579
joycasino-tvv.top., 483572
goldplazacasino.com., 331951
pokerstatic.com., 237172
slot-v-casino.com., 228602
casino-x-online.com., 226560
frank-casinos.com., 184766
vulcan-casino-online.co., 173148
astanacasino.com., 149227
faraoncazino.com., 148050
selectorcasino.net., 66419
casinoebet.com., 63199
```

```
grande-casino.xyz., 54270
absolutepoker.com., 2813
padtycasino.com., 2604
casino-midas.com., 1570
manodipoker.com., 1050
casinoforfree.com., 951
pokerfares.com., 948
3dslots.com., 936
```

We can use Iris Investigate to look at the most recent screenshot for each of those sites, saving the result locally by right clicking on the as-displayed image.

Once we've saved copies of all the relevant images, we can then try running some comparisons.

Eyeballing `slot-v-casino.com.jpg`, we can see that this is a domain that's available for sale. The top of that screenshot looks like:



Are there any other domains in the other 19 top domains from k4x that are similar to the screenshot for slot-v-casino.com? Let's check... we'll use a little script that looks like:

```
$ cat k4x3.txt
./compare_two_images.py slot-v-casino.com.jpg joycasino-ym.top.jpg
./compare_two_images.py slot-v-casino.com.jpg joycasino-tvv.top.jpg
./compare_two_images.py slot-v-casino.com.jpg goldplazacasino.com.jpg
./compare_two_images.py slot-v-casino.com.jpg pokerstatic.com.jpg
./compare_two_images.py slot-v-casino.com.jpg casino-x-online.com.jpg
./compare_two_images.py slot-v-casino.com.jpg frank-casinos.com.jpg
./compare_two_images.py slot-v-casino.com.jpg vulcan-casino-online.co.jpg
./compare_two_images.py slot-v-casino.com.jpg astanacasino.com.jpg
./compare_two_images.py slot-v-casino.com.jpg faraoncazino.com.jpg
```

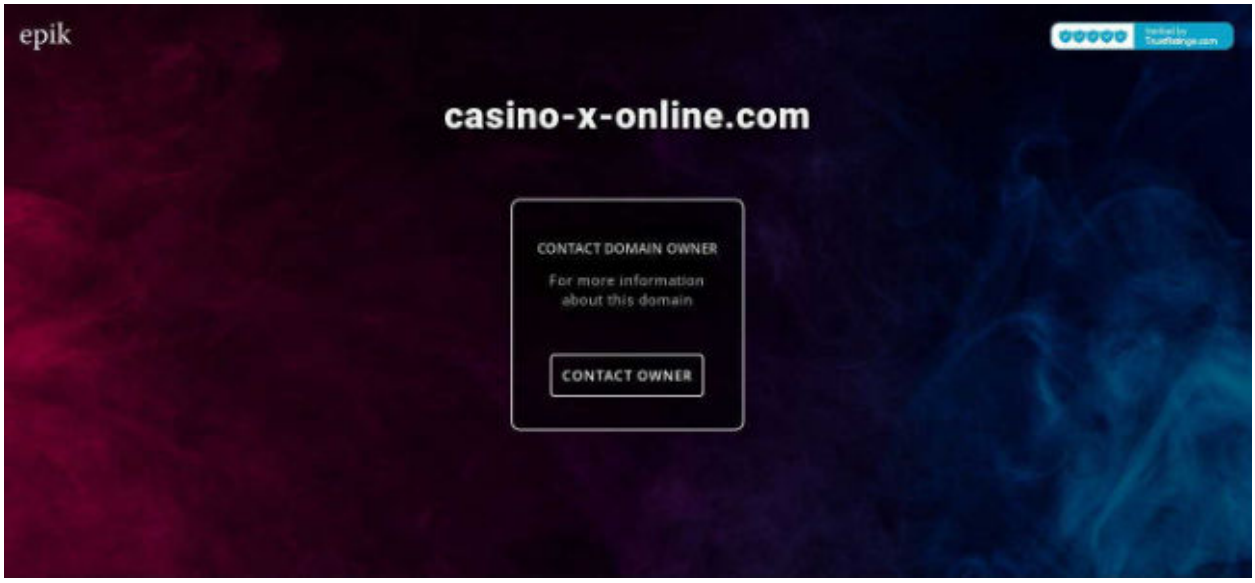
```
./compare_two_images.py slot-v-casino.com.jpg selectorcasino.net.jpg
./compare_two_images.py slot-v-casino.com.jpg casinoebet.com.jpg
./compare_two_images.py slot-v-casino.com.jpg grande-casino.xyz.jpg
./compare_two_images.py slot-v-casino.com.jpg absolutepoker.com.jpg
./compare_two_images.py slot-v-casino.com.jpg padtycasino.com.jpg
./compare_two_images.py slot-v-casino.com.jpg casino-midas.com.jpg
./compare_two_images.py slot-v-casino.com.jpg manodipoker.com.jpg
./compare_two_images.py slot-v-casino.com.jpg casinoforfree.com.jpg
./compare_two_images.py slot-v-casino.com.jpg pokerfares.com.jpg
./compare_two_images.py slot-v-casino.com.jpg 3dslots.com.jpg
```

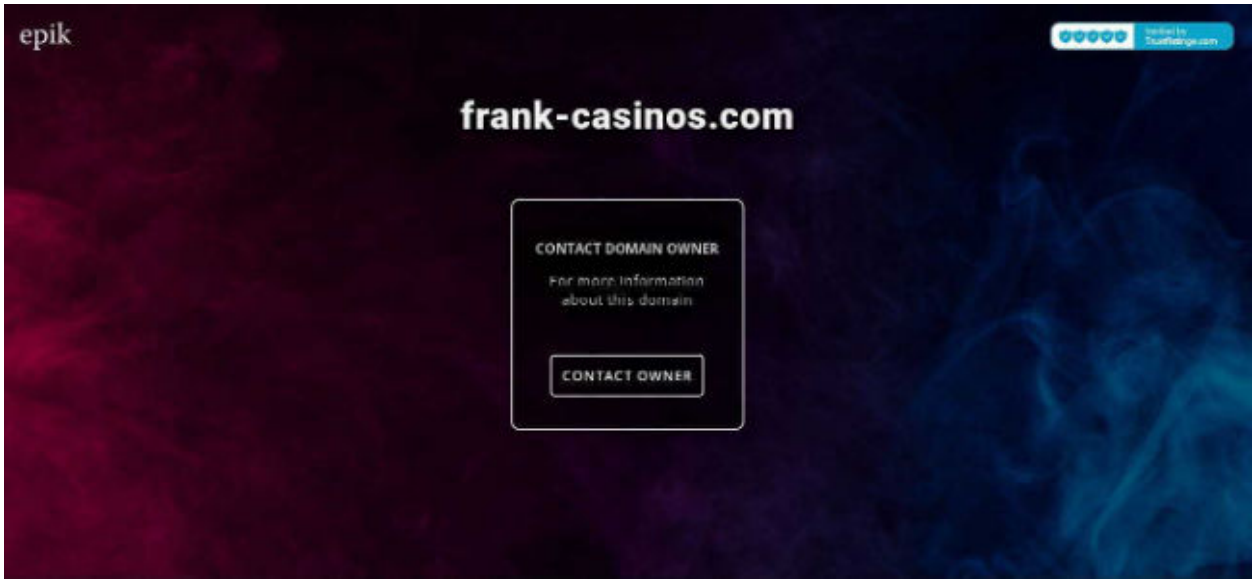
When we run that script, we see:

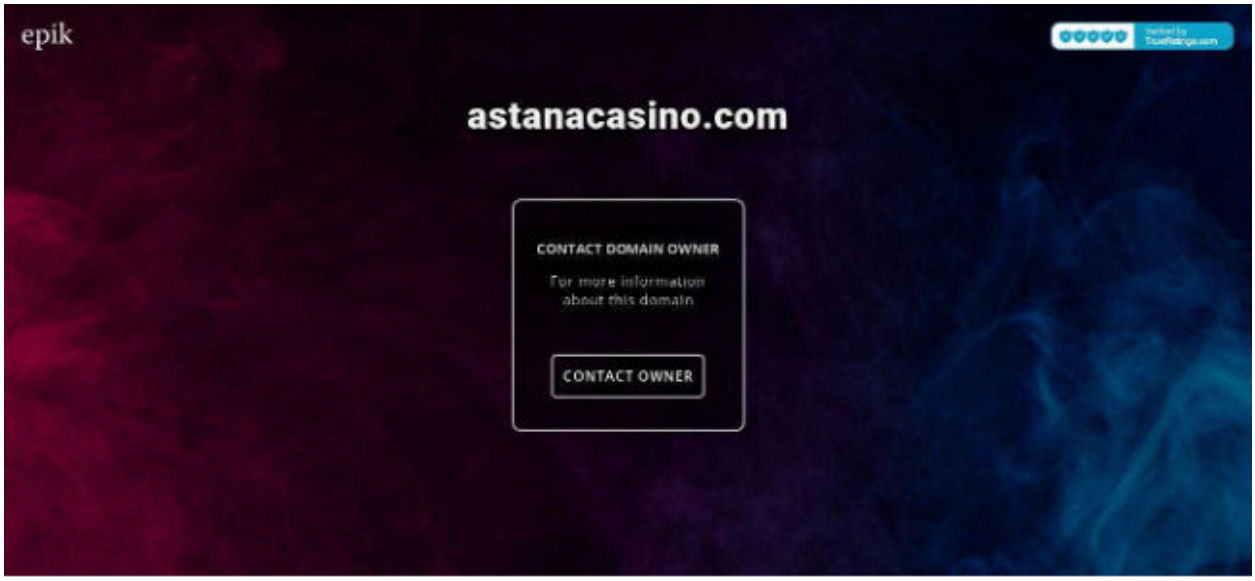
```
$ sh -x k4x3.txt
+ ./compare_two_images.py slot-v-casino.com.jpg joycasino-ym.top.jpg
Perceputal hash distance is: 0.515625
Threshold is: 0.2
+ ./compare_two_images.py slot-v-casino.com.jpg joycasino-tvv.top.jpg
Perceputal hash distance is: 0.515625
Threshold is: 0.2
+ ./compare_two_images.py slot-v-casino.com.jpg goldplazacasino.com.jpg
Perceputal hash distance is: 0.3828125
Threshold is: 0.2
+ ./compare_two_images.py slot-v-casino.com.jpg pokerstatic.com.jpg
Perceputal hash distance is: 0.1796875
Threshold is: 0.2
+ ./compare_two_images.py slot-v-casino.com.jpg casino-x-online.com.jpg
Perceputal hash distance is: 0.015625
Threshold is: 0.2
+ ./compare_two_images.py slot-v-casino.com.jpg frank-casinos.com.jpg
Perceputal hash distance is: 0.0234375
Threshold is: 0.2
+ ./compare_two_images.py slot-v-casino.com.jpg vulcan-casino-online.co.jpg
Perceputal hash distance is: 0.4921875
Threshold is: 0.2
+ ./compare_two_images.py slot-v-casino.com.jpg astanacasino.com.jpg
Perceputal hash distance is: 0.03125
Threshold is: 0.2
+ ./compare_two_images.py slot-v-casino.com.jpg faraoncazino.com.jpg
Perceputal hash distance is: 0.0234375
Threshold is: 0.2
+ ./compare_two_images.py slot-v-casino.com.jpg selectorcasino.net.jpg
Perceputal hash distance is: 0.0078125
Threshold is: 0.2
+ ./compare_two_images.py slot-v-casino.com.jpg casinoebet.com.jpg
Perceputal hash distance is: 0.4921875
Threshold is: 0.2
+ ./compare_two_images.py slot-v-casino.com.jpg grande-casino.xyz.jpg
Perceputal hash distance is: 0.453125
```

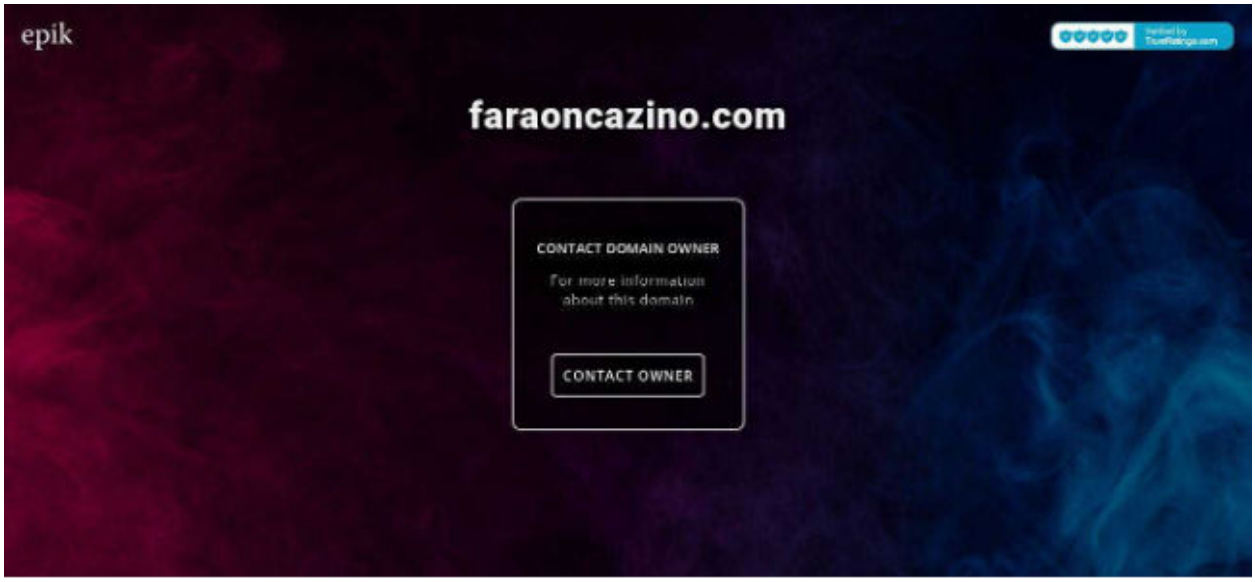
```
Threshold is: 0.2
+ ./compare_two_images.py slot-v-casino.com.jpg absolutepoker.com.jpg
Perceptual hash distance is: 0.5
Threshold is: 0.2
+ ./compare_two_images.py slot-v-casino.com.jpg padtycasino.com.jpg
Perceptual hash distance is: 0.5
Threshold is: 0.2
+ ./compare_two_images.py slot-v-casino.com.jpg casino-midas.com.jpg
Perceptual hash distance is: 0.0078125
Threshold is: 0.2
+ ./compare_two_images.py slot-v-casino.com.jpg manodipoker.com.jpg
Perceptual hash distance is: 0.390625
Threshold is: 0.2
+ ./compare_two_images.py slot-v-casino.com.jpg casinoforfree.com.jpg
Perceptual hash distance is: 0.3046875
Threshold is: 0.2
+ ./compare_two_images.py slot-v-casino.com.jpg pokerfares.com.jpg
Perceptual hash distance is: 0.3828125
Threshold is: 0.2
+ ./compare_two_images.py slot-v-casino.com.jpg 3dslots.com.jpg
Perceptual hash distance is: 0.53125
Threshold is: 0.2
```

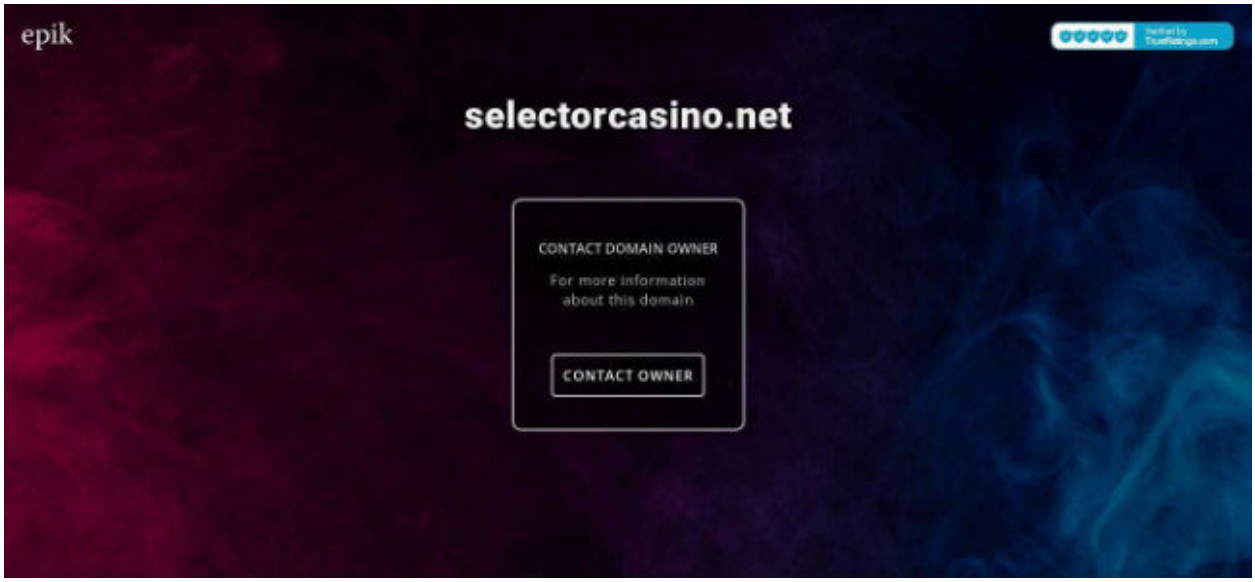
So, 7 of our 19 screenshots (plus our original reference domain) appears to form a cohesive set. Eyeballing those screenshots, they are indeed visually similar, confirming what our code identified automatically. Looking just at the top block for each of those, we see:

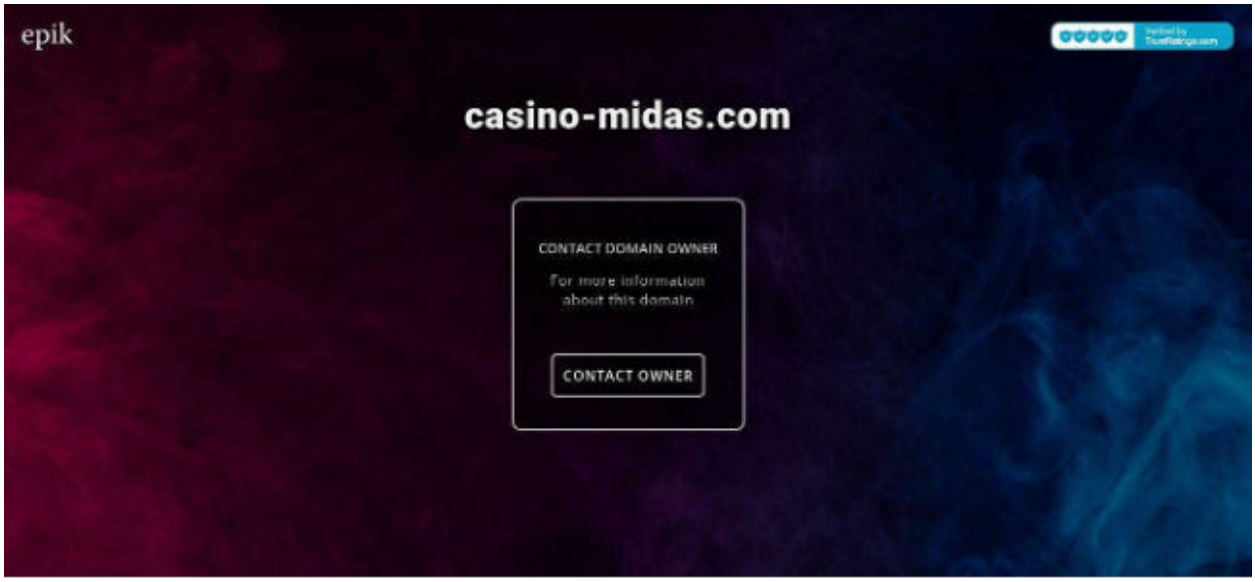

epik
casino-x-online.com
CONTACT DOMAIN OWNER
For more information about this domain
CONTACT OWNER


epik
frank-casinos.com
CONTACT DOMAIN OWNER
For more information about this domain
CONTACT OWNER


epik
astanacasino.com
CONTACT DOMAIN OWNER
For more information about this domain
CONTACT OWNER


epik
faraoncazino.com
CONTACT DOMAIN OWNER
For more information about this domain
CONTACT OWNER





You get the idea. We are obviously able to find similar screenshots using this approach.

A More In-Depth Case Study: “Triple 7” Gambling-Related Domains

While working with the list of domains we extracted from DNSDB Flexible Search, we noticed that there were over 3,600 unique base (registrable) domains containing the literal string "777," presumably because "seven is a lucky number."[43] Let's see if we can identify groups of similar "brands" from the triple seven screenshots associated with that domain set.

We began by manually reviewing and saving screenshots for those domains (3,085 are available).

[43] "Here’s Why 7 Is Considered a Lucky Number," https://www.rd.com/article/number-7/

The next step was to compute the perceptual hash of each of those screenshots. We did that with `phash_domains_current_dir.py`

That resulted in a series of perceptual hash values that looked like:

```
[snip]
0T4YuTvoB+Ca/cwPLKmx8C4HOp5l4cQHPB+l5mPALtk= original777casino.net.jpg
0U8QTK60LzGPoRVKlUoY5i6mDqcNkdjP0Pe41u+RL7E= casino777bonus.com.jpg
0US2yznNlLE5zWxaxPFPGmtaw/DNmXlKxmmR5TFJkeQ= 777-goldslots.co.jpg
0US2yznNlLE5zWxaxPFPGmtaw/DNmXlKxmmR5TFJkeQ= 777-goldslots.com.jpg
0US2yznNlLE5zWxaxPFPGmtaw/DNmXlKxmmR5TFJkeQ= 777goldslots.co.jpg
0US2yznNlLE5zWxaxPFPGmtaw/DNmXlKxmmR5TFJkeQ= azino777slots.club.jpg
0US2yznNlLE5zWxaxPFPGmtaw/DNmXlKxmmR5TFJkeQ= azino777slots.top.jpg
0US2yznNlLE5zWxaxPFPGmtaw/DNmXlKxmmR5TFJkeQ= azino777slots.xyz.jpg
0VuRKmqwKtUZudLYaohribjV1DaFetUqhHZv6m/pIpM= azino777kasino.bike.jpg
0VuRKmqwKtUZudLYaohribjV1DaFetUqlHZv6mvpIpM= azino777kasino.info.jpg
0VuRKmqwKtUZudLYaohribjV1DaFetUqlHZv6mvpIpM= azino777kasino.one.jpg
0VuRKmqwKtUZudLYaohribjV1DaFetUqlHZv6mvpIpM= azino777kasino.org.jpg
0fBZoWwPHLW5O0t6mkwNlszCNIWzknNys3Nkne2aErI= casino-777vulkan.com.jpg
0fBZoWwPHLW5O0t6mkwNlszCNIWzknNys3Nkne2aErI= cazino-777vulcan.com.jpg
0lLwcKU1JaUf706qSqpwqDjAJeUP7w+veOJyomayG+M= online-slots777.xyz.jpg
0s+8y70wbTTSz1LLTjy9Njkjxs8SzGUwbDkiB8G2g3A= 777casinoonline.com.jpg
0s+8y70wbTTSz1LLTjy9Njkjxs8SzGUwbDkiB8G2g3A= turkcecasino777.com.jpg
0sW+H70yQODSz0PPvTAtOMLPQs9kNgwQ0/CHvprNwM0= my777slots.club.jpg
0sW+H70yQODSz0PPvTAtOMLPQs9kNgwQ0/CHvprNwM0= pokerwin777.com.jpg
[snip]
```

We could (theoretically) have computed the Hamming distance between the perceptual hashes for each pair of domains. If we were to "brute force" this, that would imply potentially making 4,757,070 comparisons:

```
(3,085*(3085-1))/2=4,757,070
```

Why not a full 3,085**2=9,517,225 comparisons?

(1) The distance between perceptual hashes is commutative (A-->B returns the same distance as B-->A) so we only need to do half of all possible comparisons (that "gets us" the "divide by two" part of the formula).

(2) We also know that we don't need to compare the perceptual hash of a domain to itself – by definition that distance will always be zero (that allows us to subtract one from one of the two 3085 values).

Nonetheless, even with those efficiencies, doing nearly 4.8 million comparisons would still be a relatively poor choice algorithmically[44] and a lot of work. We could use any of a number of more scalable approaches, but that's beyond the scope of today's report. For today, we can conceptually "cheat" and take advantage of the fact that many screenshot images may have the exact same perceptual hash to reduce our problem's size. Sorting and counting common values, we see things like:

```
38 id1cqiJXdyijdVyiiNzYzCMzdzOjXVzOjNbZiJshZjM=
29 9iAmTNzOiZdzMyM5ZmysxtzmmxlxmSZnzGYzGTOZxmY=
21 9iAmTNzOiZdzMyM5Zmws5tzumxkxmSZnzGYzGTOZxmY=
20 7aXtpZJUEkpoW+2bLZkSpJtk6STkzRbZaTbJNrbSFtk=
16 wA+FCo0YzLPut8q3yrPagprSn9KZ0rjy+OLY4pjKkEo=
15 xLLUsjsRew/rSZ7hnmDKSMpexHaGNMZshl2YO5G/ka0=
15 wA+FC8+Syrfat57CndK58vjymMqQyo3aTJxGGVIdUE8=
15 riUbIb2GvJYzKWtttCXG1sCWzmfDa8Np0NzA2JGYm3k=
15 9igmTNzGiZdzMyM5ZmysxtzmmZlxmSZmzGYzGTOZzmY=
15 7ZSstpNiE2tlbeyNLM0TMpky7bLmxFLMbZHtm5JLEkw=
13 wA+FCo0YyLPqt+q32rOawp7SndK48rjiuOLY4pLKgco=
13 7aXtpZJUEkpoW+2bLZkS5Jtk6STkyRbZaTbJNrbSFtk=
13 0ECRjTsWOCczGGM4a3KS0nm8k+I9J7adPG2WR83HZbw=
12 8Dx4rFqlYjFHo0dDhVod1jlKHFwPy0fHTrhlXm/OBXo=
11 wPKYJJS0kJw+Hb4daR09HTRT4WY0Y3RjYWY0H2X/Rfc=
11 riUbIb2GvJZzKWtttCXG1sCWzmfDacNp0NzA2JGYm3k=
11 9iEmTNzOiZdzMyM5ZmwsxtzmmxlxmSZnzGYzGTOZxmY=
10 wVqenZ6NsjWwdeFhNDbh8D4+Ho9D2EPIQ/ger0H2QeI=
10 w/CSE7gfk0M7T9NJOwtzSTtJM+lK4ZbhluWG5YbllqE=
10 g/Q3ezwPXIw8D5Mzk/Ew6MD0TMvGtEcMLw5vDjsbkTs=
10 9igmTNzGiZdzMyM5ZmysxtzmmxlxmSZnzGYzGTOZxmY=
10 9iEmTNzOiZNzMyM5Zmws5tzumxkxmSZnzGYzGTOZxmY=
10 4UKSyZPLw8tjS2NLa07DSJPMxsyQ55LnmMeYz7jKxtg=
[etc]
```

That simple step took us down from 3,085 hashes to 2,248 hashes.

We decided to now check out what some of those top perceptual hash values looked like.

(1) Our most-commonly seen hash value was id1cqiJXdyijdVyiiNzYzCMzdzOjXVzOjNbZiJshZjM=

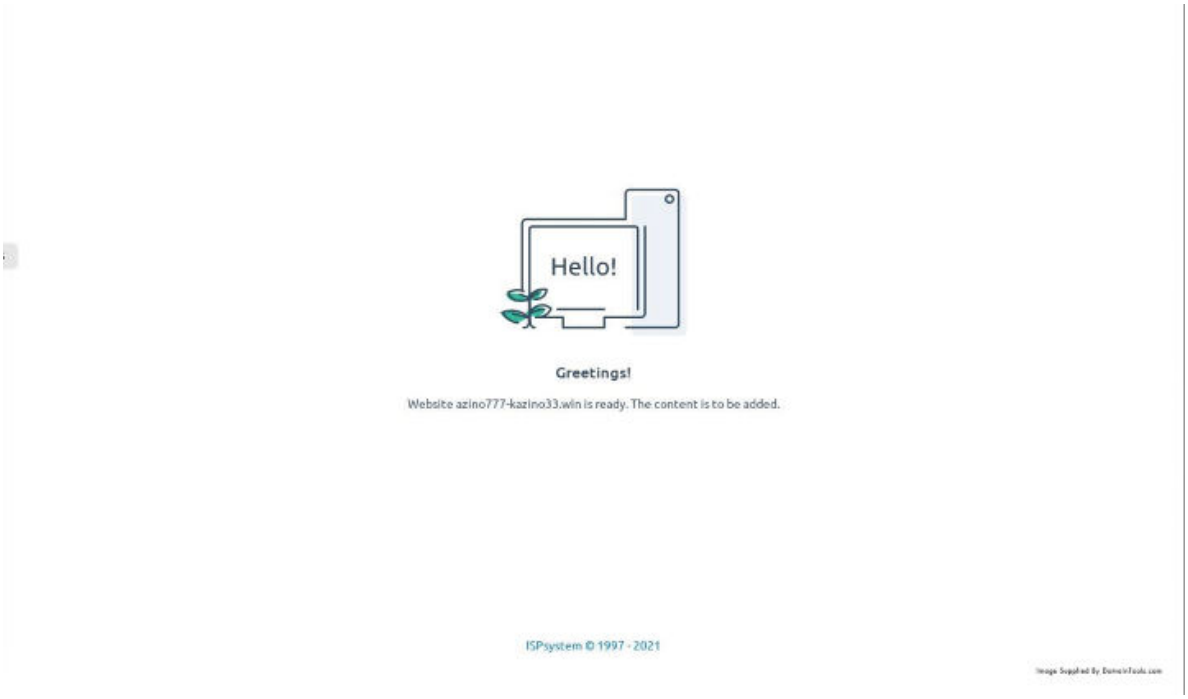
That hash is associated with 38 unique domains. A domain image with that perceptual hash (and shown in the screenshot below) is **777cazino--777.ru.jpg** (for this example, we're naming our screenshots with the domain name plus a .jpg suffix):

[44] https://adrianmejia.com/most-popular-algorithms-time-complexity-every-programmer-should-know-free-online-tutorial-course/

---

(2) 2nd most common hash value is 9iAmTNzOiZdzMyM5ZmysxtzmmxlxmSZnzGYzGTOZxmY=
That hash is associated with 29 unique domain images such as **azino777-kazino33.win.jpg**

That image looks like:



What's interesting is that there are **OTHER PERCEPTUAL HASHES** (and associated domain screenshot images) that **ALSO** look like the screenshot shown above, including:

21	9iAmTNzOiZdzMyM5Zmws5tzumxkxmSZnzGYzGTOZxmY=	such as azino777-casino-officialnyi18.win.jpg
15	9igmTNzGiZdzMyM5ZmysxtzmmZlxmSZmzGYzGTOZzmY=	such as casino-pinup-play777.win.jpg
11	9iEmTNzOiZdzMyM5ZmwsxtzmmxlxmSZnzGYzGTOZxmY=	such as azino777-casino-official2.xyz.jpg
10	9igmTNzGiZdzMyM5ZmysxtzmmxlxmSZnzGYzGTOZxmY=	such as azino777-kazino101.ru.jpg
10	9iEmTNzOiZNzMyM5Zmws5tzumxkxmSZnzGYzGTOZxmY=	such as azino777-casino-officials295.win.jpg
8	9iAmTdzOiZdzMyM5ZmwsxtzmmxlxmSZnzGYzGTOZxmY=	such as joycasino-official-site777.win.jpg
7	9iAmTNzOiZdzMyM5Zmws5tzmmxlxmSZnzGYzGTOZxmY=	such as casinopinup-officialsites777.win.jpg

That's the 111 unique screenshot images that look like the example screenshot shown.

It's one thing to talk about those images "looking similar," but what do we see if we find the actual Hamming distance between those perceptual images? And are the differences less than our threshold value of < 0.2?

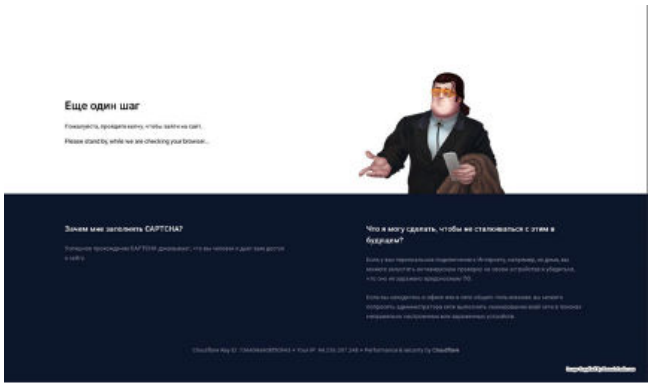
azino777-kazino33.win.jpg	azino777-casino-officialnyi18.win.jpg	0.015625
azino777-kazino33.win.jpg	casino-pinup-play777.win.jpg	0.0234375
azino777-kazino33.win.jpg	azino777-casino-official2.xyz.jpg	0.0078125
azino777-kazino33.win.jpg	azino777-kazino101.ru.jpg	0.0078125
azino777-kazino33.win.jpg	azino777-casino-officials295.win.jpg	0.0234375
azino777-kazino33.win.jpg	joycasino-official-site777.win.jpg	0.0078125
azino777-kazino33.win.jpg	casinopinup-officialsites777.win.jpg	0.0078125
azino777-casino-officialnyi18.win.jpg	casino-pinup-play777.win.jpg	0.0390625
azino777-casino-officialnyi18.win.jpg	azino777-casino-official2.xyz.jpg	0.015625
azino777-casino-officialnyi18.win.jpg	azino777-kazino101.ru.jpg	0.0234375
azino777-casino-officialnyi18.win.jpg	azino777-casino-officials295.win.jpg	0.0078125
azino777-casino-officialnyi18.win.jpg	joycasino-official-site777.win.jpg	0.015625
azino777-casino-officialnyi18.win.jpg	casinopinup-officialsites777.win.jpg	0.0078125
casino-pinup-play777.win.jpg	azino777-casino-official2.xyz.jpg	0.03125
casino-pinup-play777.win.jpg	azino777-kazino101.ru.jpg	0.015625
casino-pinup-play777.win.jpg	azino777-casino-officials295.win.jpg	0.046875

casino-pinup-play777.win.jpg	joycasino-official-site777.win.jpg	0.03125
casino-pinup-play777.win.jpg	casinopinup-officialsites777.win.jpg	0.03125
azino777-casino-official2.xyz.jpg	azino777-kazino101.ru.jpg	0.015625
azino777-casino-official2.xyz.jpg	azino777-casino-officials295.win.jpg	0.015625
azino777-casino-official2.xyz.jpg	joycasino-official-site777.win.jpg	0.0078125
azino777-casino-official2.xyz.jpg	casinopinup-officialsites777.win.jpg	0.0078125
azino777-kazino101.ru.jpg	azino777-casino-officials295.win.jpg	0.03125
azino777-kazino101.ru.jpg	joycasino-official-site777.win.jpg	0.015625
azino777-kazino101.ru.jpg	casinopinup-officialsites777.win.jpg	0.015625
azino777-casino-officials295.win.jpg	joycasino-official-site777.win.jpg	0.0234375
azino777-casino-officials295.win.jpg	casinopinup-officialsites777.win.jpg	0.015625
joycasino-official-site777.win.jpg	casinopinup-officialsites777.win.jpg	0.0078125

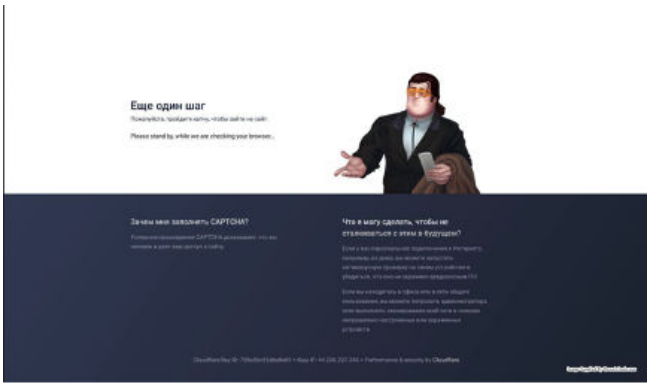
---

(3) Third most popular individual hash? 7aXtpZJUEkpoW+2bLZkSpJtk6STkzRbZaTbJNrbSFtk= as associated with 777-vulcan-casino.net.jpg and 19 other domains. However, as we looked through screenshots, we also noted three "brother" screenshot images:

777-vulcan-casino.net.jpg

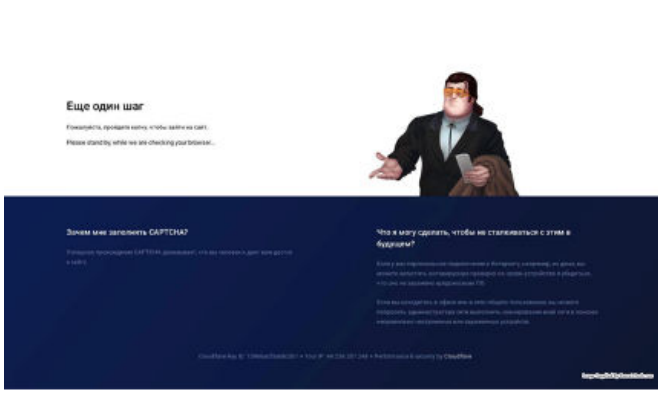


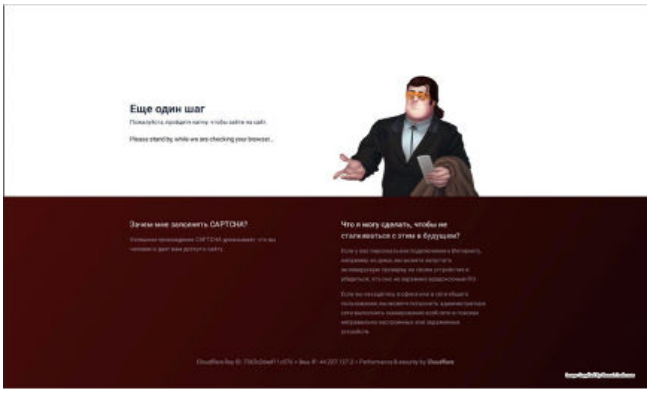
azino777casino.best.jpg



777-free-slots.com.jpg

maxbetcasino777.com.jpg





Those sure *look* similar, don't they? In fact, those images DO differ in small but noticeable ways (hint: look at the background color and the width of the text "below the guy" in each image):

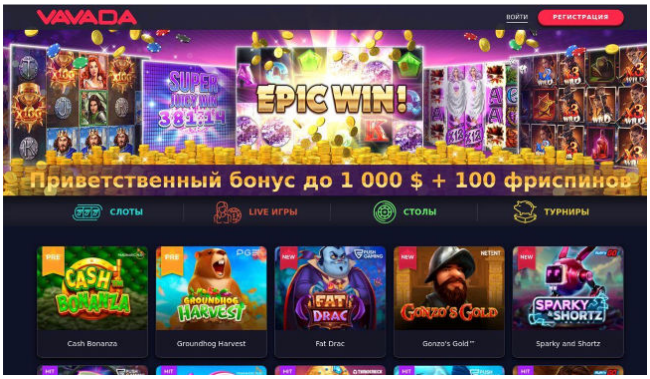
20	7aXtpZJUEkpoW+2bLZkSpJtk6STkzRbZaTbJNrbSFtk=	such as 777-vulcan-casino.net.jpg
15	7ZSstpNiE2tlbeyNLM0TMpky7bLmxFLMbZHtm5JLEkw=	such as azino777casino.best.jpg
13	7aXtpZJUEkpoW+2bLZkS5Jtk6STkyRbZaTbJNrbSFtk=	such as 777-free-slots.com.jpg
6	7ZTstpNiE2tlbeyNLN0TMpky7bLmxFLMKZHtm5JJEmw=	such as maxbetcasino777.com.jpg

Checking the distance between the perceptual hashes of those screenshots we can see that some pairs are NOT considered similar – we've rendered those in red in the table below:

777-vulcan-casino.net.jpg	azino777casino.best.jpg	**0.3046875**
777-vulcan-casino.net.jpg	777-free-slots.com.jpg	**0.0078125**
777-vulcan-casino.net.jpg	maxbetcasino777.com.jpg	**0.3046875**
azino777casino.best.jpg	777-free-slots.com.jpg	**0.3125**
azino777casino.best.jpg	maxbetcasino777.com.jpg	**0.0234375**
777-free-slots.com.jpg	maxbetcasino777.com.jpg	**0.3125**

---

(4) Fourth most popular hash: wA+FCo0YzLPut8q3yrPagprSn9KZ0rjy+OLY4pjKkEo= as associated with vavada-casino777p.ru.jpg



16 wA+FCo0YzLPut8q3yrPagprSn9KZ0rjy+OLY4pjKkEo=	such as vavada-casino777p.ru.jpg
13 wA+FCo0YyLPqt+q32rOawp7SndK48rjiuOLY4pLKgco=	such as casino-vavada777.ru.jpg
9 wA+FCo0IyJfqt8q3yreaop7SmNKY1rjwuOL44tjKnco=	such as casinosvavada777.ru.jpg

vavada-casino777p.ru.jpg	casino-vavada777.ru.jpg	0.0703125
vavada-casino777p.ru.jpg	casinosvavada777.ru.jpg	0.0859375
casino-vavada777.ru.jpg	casinosvavada777.ru.jpg	0.0859375

Limitations To This Approach

While we've successfully demonstrated that we can identify related domains based on their screenshots, there are limitations we've not discussed so far:

- Some sites may intentionally (or inadvertently) prevent the successful capture of screenshots. *If domains can't be screenshot, there's obviously nothing available for comparison* (or what is available may not be meaningful for subject-specific analyses).

- When screenshots ***can*** be successfully captured, they may be relatively large. For example, our study of the "triple 7" screenshots (3,085 images) entailed 2,273,329,012 octets-worth of downloads. That's an average screenshot size of ~737KB per screenshot. *It would be more efficient if users could simply download a small precomputed standardized perceptual hash if that's all that's needed for image comparison purposes.*

- When sites are able to be successfully captured, *some sites may have intentionally-frequently-changing home pages* (perhaps featuring frequently-updated "fresh news itemettes"). This can result in a need to frequently refetch/refresh existing screenshots.

- *Some likely-related images may not be mechanically identified as similar by perceptual hash methods even when human analysts would readily spot the images as being "related."* An example of this would include the 777-vulcan-casino.net / azino777casino.best / 777-free-slots.com / maxbetcasino777.com example shown above. This may be a reflection that we should be using an alternative approximate hash, or multiple mechanisms (rather than just one, as we did here).

- *We've not discussed full at-scale database solutions for working with millions or hundreds of millions of images.* There are many potential approaches to this sort of

thing, including Burkhard-Keller trees[45], Ball trees (aka VP trees),[46] multi-index hashing,[47] and other approaches, but that's beyond the scope of this proof-of-concept.

In spite of these shortcomings, we hope that you've nonetheless found this to be an intriguing exploration of screenshot-driven programmatic capabilities.

[45] http://blog.notdot.net/2007/4/Damn-Cool-Algorithms-Part-1-BK-Trees

[46] https://pyimagesearch.com/2019/08/26/building-an-image-hashing-search-engine-with-vp-trees-and-opencv/

[47] http://www.cs.toronto.edu/~norouzi/research/papers/multi_index_hashing.pdf

Conclusion

"Memores acti prudentes futuri"
[Mindful of what has been done, aware of what will be]

This report has covered a lot of ground. Just to briefly recap some of the highlights by way of conclusion, among other things, we've:

- Shown the latent value that exists in DomainTools collection of domain screenshots
- Explained exact and approximate image hashing approaches, and demonstrated some examples of what does and doesn't thwart approximate image comparisons
- Discovered gambling-related domain names, assessed the location of those domains, and the risk associated with gambling domains from some TLDs
- Applied perceptual hashing to gambling related domains to identify sets of similar domains
- Discussed limitations to the demonstrated approach, and highlighted options for some potential future work
- Provided code for the examples used in the report

If you have any questions about any of this, we welcome your feedback:
https://www.domaintools.com/contact

Appendices

Appendix 1: 2nd-level-dom-large

```
#!/usr/bin/perl
use strict;
use warnings;
use IO::Socket::SSL::PublicSuffix;

my $pslfile = '/usr/local/share/public_suffix_list.dat';
my $ps = IO::Socket::SSL::PublicSuffix->from_file($pslfile);

while (my $line = <STDIN>) {
      chomp($line);
      my $root_domain = $ps->public_suffix($line,1);
      printf( "%s\n", $root_domain );
}


# Note: public_suffix_list.dat comes from https://publicsuffix.org/
```

Appendix 2: phash_domains_current_dir.py (compute phash for an image in the current dir)

```
#!/usr/local/bin/python3
""" phash list of base domains """
# python3 libraries
import io
import os
import sys
from pathlib import Path
import requests
from PIL import Image
# https://github.com/thorn-oss/perception/tree/master
from perception import hashers

def get_domains():
      """ Get at least one base domain to phash from sys.stdin pipe """
      if not os.isatty(sys.stdin.fileno()):
      domains_to_phash=sys.stdin.readlines()
      domains_to_phash_count=len(domains_to_phash)
      if domains_to_phash_count < 1:
            raise ValueError("Pipe in at least one base domain")
      else:
      raise ValueError("Pipe in at least one base domains")
```

```
        print("Base domains to phash: "+str(domains_to_phash_count))
        return domains_to_phash

def retrv_scrnshot_and_phash(dom):
        """ get the screenshot image and perceptual hash it """
        hasher=hashers.PHash(hash_size=16)
        dom=dom.rstrip()
        with open(dom,'rb') as f:
        bytes=f.read()
        try:
        image=Image.open(io.BytesIO(bytes))
        # https://github.com/thorn-oss/perception
        my_hash=hasher.compute(image)
        print(my_hash,dom)
        except:
        # sys.stderr.write(str(bytes))
         # print()
        pass

        # save the perceptual hash to disk
        return

def main():
        """ Process the supplied base domains """
        # loop over the base domains
        domains_to_phash=get_domains()

        for dom in domains_to_phash:
        dom=dom.rstrip()
        test_phash=retrv_scrnshot_and_phash(dom)

        print(dom,test_phash)

if __name__ == "__main__":
main()
```

Appendix 3: compare_two_images.py -- compare two domains from the same directory

```
$ cat compare_two_images.py
#!/usr/local/bin/python3
""" Compare two images using percentual hashes """

import sys
from PIL import Image

# https://github.com/thorn-oss/perception/tree/master
```

```
from perception import hashers

if len(sys.argv) != 3:
      raise ValueError("Error: must supply two images as arguments")

image1 = str.rstrip(sys.argv[1])
image2 = str.rstrip(sys.argv[2])

img1 = Image.open(image1)
img2 = Image.open(image2)

# Perceptual hash static definitions -- https://github.com/thorn-oss/perception
hasher = hashers.PHash(hash_size=16)
my_hash1 = hasher.compute(img1)
my_hash2 = hasher.compute(img2)

# recommended threshold for PHash (hash_size=16), "expected false positive rate
of <1%"
threshold = 0.2

distance=hasher.compute_distance(my_hash1, my_hash2)
print("Perceputal hash distance is: "+str(distance))
print("Threshold is: "+str(threshold))
```

"Giant Steps," John Coltrane (1959)
https://www.youtube.com/watch?v=30FTr6G53VU
6.8 million views

#13 on the Jazz24.org Jazz 100