Understanding and Mitigating Malware Assignment

Submitted for Guillermo Suarez de Tangil and Hector Menendez

August 13, 2024

RAGeQUIT

A Chrome extension, propagated via an LLM-driven social media campaign, that will do RAG in the browser by making API calls to OpenAI. It will also serve up a link to the user that will install a cryptominer. The campaign draws inspiration from the techniques of BlackCat[9], WARMCOOKIE[10], and DodgeBox[1].

Propagation

Our malware operation begins with a campaign to propagate a Chrome extension via social media. This will be supported and enhanced via LLMs. The strategy breaks down as follows:

1. Create botfarm. This will break down as follows:
    - Leveraging automation generate email addresses on platforms including Yahoo, Google, Microsoft and Hotmail. Automate as much of the process as possible. Inject a human into the loop to defeat captchas, this will limit the number of accounts that can be created per hour, however a few days work should produce a sufficient volume of accounts. The target no. of accounts to open should be 10,000. This figure shall have some random noise injected into it to prevent it being such a neat round number.
    - Use the email accounts generated, create social media profiles. The profiles should be created in a staggered fashion over the course of 6 months to prevent them being detected via clustering methodologies. Facebook, Twitter, TikTok and Instagram are the desired target platforms. Bots will not consistently have all 4. They shall have a randomly selected sample between 1 and 4. This should result in a final tally of 25,000 social media accounts, plus/minus some noise to avoid round numbers.
    - User profiles will be generated via a combination of foundation model-based vision applications, and large language model generated content. On-prem models should ideally be used for this. Meta's Llama 3.1 8B appeals the most. As a small parameter model it will be the most straightforward and cheap to run, and verifiable metrics exist to attest to its performance in the task we've given it. Namely that it is capable of generating convincing fake user content, demonstrated by its convincing performance in the SpearPhishing component of Meta's CyberSecEval3 Benchmark[4]. While this model does notionally ship with controls in the form of LlamaGuard[3], these controls ought not be a limitation since on the face of it we are only trying to get the LLM to generate engaging content, something they will be expected to do under legitimate circumstances.
2. 40% of bot accounts, all necessarily non-Google based emails, are to install the Chrome extension. This is part of an attempt to build trust in the extension. They will also leave complimentary, albeit appropriately balanced, reviews.

3. Social media campaign promoting Chrome Extension. Bots will make an effort to push the extension via the most direct means available. Messaging DMs preferred, forum posts will also be pursued as an option.

4. End goal is to achieve Chrome extension downloads. Target is 20,000 real installs.

Payload Delivery

Our payload will be delivered in multiple stages:

1. A Chrome extension has been chosen as the first stage delivery vector for our campaign. It is a trusted platform, yet in spite of seemingly sincere efforts by Google to implement robust review processes[5], there remains anecdotal evidence that these controls have not been effective, and that Chrome remains a viable vector for payload delivery[12] (citation is of an extension that as of 13/08/24 was live on the Chrome Webstore and was flagrantly malware, updated as recently as 29/07/24).

2. Once the extension is installed it will, as advertised, make API calls to GPT-4o-mini, via a custom GPT. This particular model choice has been made from a cost[8], reliability, and quality standpoint. From the perspective of the Chrome extension very little, if anything, suspicious will be going on. Indeed, to save development cost and time, we intend to flagrantly rip off an existing application that does something very similar, albeit with minimal modification to avoid it being too obvious to reviewers[7]. The major difference will be that our app will communicate with a remote custom GPT rather than a local model. We have taken this approach since we think it is unlikely we will be able to convince users to run a local model, and we want to keep the installation process easy to convert as many installs as possible. We are gambling that review processes at OpenAI are even weaker than they are at Google, and that a custom GPT that communicates with an external web server and forwards requests unaudited by OpenAI will likely be approved. To mitigate spiralling costs from API usage the extension will begin to throttle user token requests exponentially after 2 weeks, until the application becomes completely unusable. The manner in which this throttling is executed will hopefully mean that a poor reviews can be attributed to users devices malfunctioning rather than the extension. In any case this should be more than enough time for the deployment of the agent.

3. The custom GPT will push URLs into the dialogue box that the extension is using, these links will claim to be pages hosting HTML summaries of the conversation so far, and indeed they will be.

4. The HTML pages will also contain Javascript, which will silently invoke WScript.

5. WScript will invoke the Background Intelligent Transfer Service (BITS)[16] to download the RAGeQUIT malware application.

The second half of this chain of payload delivery is heavily inspired by WARMCOOKIE[10].

Privilege Escalation and Persistence

The malware draws inspiration from BlackCat/ALPHV[9] on privilege escalation in Windows. To enable the application to persist effectively it ideally should first gain system level privileges. To do this it will try to leverage the CMSTPLUA COM interface. This will enable it to spawn a process, namely our miner application, with elevated privileges. We would like our application to be able to survive a shutdown event, and even a user uninstalling the extension. To achieve this, once the elevated process is created, a scheduled task checking for the existence of the RAGeQUIT malware is created. If the task does not detect the malware it makes an attempt to once again use BITS to reinstall it.

Cryptominer

The main application shall be a Monero miner. The miner will make use of a few stealth properties. It will take instructions from the C2 component re. where it should focus its efforts in terms of hash checking. It will work as part of a mining pool consisting of other compromised devices. Our modus operandi shall be low-and-slow mining to maximise profits over a 12-24 month horizon. We hope to avoid detection for the duration of the campaign. The miner shall leverage the RandomX[15] mining algorithm, as is required by the Monero coin. Due to its design as a CPU-optimised democratised coin, this gives the devices we intend to infect a competitive advantage versus if we instead opted to mine a more conventional proof-of-work coin such as Bitcoin, the mining market for which is dominated by ASICs and GPUs optimised for the task. On the other hand it strikes a balance as being a coin that has a very high market capitalisation and is actively traded. This minimises the risk of the coin going to zero during our campaign, and our efforts all being for nought.

C2

C2 is necessary for coordinating agents in the mining pool, and for exfiltrating valid hashes when they are found. There are two major elements we will consider regarding C2. How traffic can be concealed, and how it can be secured. For concealment we will leverage the techniques of Domain Fronting, Domain Generation Algorithms, and obfuscating C2 traffic within a field masquerading as a cookie in the data section of HTTP packets. To secure traffic the malware will ship with a public key, that will be used to asymmetrically encrypt valid hashes once they are found.

Domain Fronting

Domain fronting has been a common technique for concealing C2 traffic in network logs for many years. It's tried and tested. Research from 2017[11] exits to suggest that it is absolutely viable as a means of evading network layer detections, and effectively obfuscating a true C2 domain. We will be imitating the technique documented here[14], leveraging the Fastly CDN for domain fronting, and disguising our traffic as calls to the PyPi repository to download Python packages. This traffic will potentially look strange to an analyst threat hunting within a corporate network, however quite likely it will seem sufficiently benign as to avoid deeper inspection.

DGA

With the C2 approach we are taking DGA remains necessary since the CDN provider can block redirects to our C2 domain, with this becoming the equivalent of a takedown. We will derive new domains from top posts on x.com (formerly Twitter). We shall take the MD5 hash of the text of the top post at a predefined time, and take the first 8 characters, then append the letters, '-cdn.com'. CDN domains notoriously look very similar to DGA domains, since in a sense they are, and are accordingly responsible for no shortage of false positives for detections looking for DGA. An analyst doing deep packet inspection on the HTTP traffic who opts not to do a deeper investigation may write this off as just standard CDN traffic being routed in a strange but plausible way. These domains will be impossible to register ahead of time, so the availability of the C2 domain is a concern. However it does mean that if the DGA is discovered and reversed it is impossible for them to be reliably pre-emptively registered/taken down.

Concealing C2 Traffic

One last time, we shall again draw inspiration from WARMCOOKIE[10]. The means by which this malware campaign concealed its C2 traffic is ingenious, and we would be remiss not to offer it the ultimate form of flattery and imitate it. Traffic will be concealed in the cookie of HTTP requests. The first 4 bytes shall be the key, all remaining bytes in the cookie shall be the fragmented data of the message, encrypted via a stream cipher. To distinguish ourselves slightly from WARMCOOKIE we shall instead make use of AES in CFB mode, which turns AES into a stream cipher rather than a block cipher. This is in case any corporate networks into which RAGeQUIT is deployed are making use of smarter deep packet inspection based detections that are now explicitly looking for WARMCOOKIE C2.

C2 Authentication

We will make use of public key cryptography to secure communications between the agent and the operator. The malware will ship with the public part of a key that can be used to asymmetrically encrypt a valid hash once one is found. We want to ensure that the hash reaches the legitimate operator, so we shall implement a 2-way handshake protocol.

1. The agent encrypts the valid hash using the public key, and sends it to the domain it currently has established contact with via the C2 channel. Until it receives a response it continues to send the discovered hash with exponential backoff. If a new C2 domain is registered it immediately retries.
2. The controller receives and decrypts the hash using the corresponding private key. They then take the SHA256 hash of the valid hash, and send that back to the agent via the C2 channel.
3. The agent receives the doubly hashed value, and verifies that it corresponds to the hash that they transmitted. This serves as proof of receipt by the legitimate operator. Once verified the agent discards the discovered hash, and ceases attempts to re-transmit.

While the option of providing similar cryptographic guarantees going the other way exists, ie. commands from controller to agent, we have opted not to implement this, in the interest of time and simplicity. The only command the controller will need to issue will be ones to coordinate the agents and what hashes they have been allocated. Other than this there is actually very little by way of command or control. With the authenticity of the traffic in that direction being of relatively low stakes it seems like a good rationalisation of engineering time not to bother.

Stealth

There shall be 3 stealth components to the malware. The whole profitability of the campaign hinges on a wide reach low-and-slow approach. It needs to not be discovered for as long as conceivably possible. One of the most likely manners in which we expect the campaign to be discovered is if the malware is deployed into a corporate network and spotted by threat researchers for a larger enterprise. For this reason the program must implement basic defence evasion tactics and must be subtle enough to avoid the attention of most modern EDR systems if possible.

1. Make use of the latest in EDR evasion in order to mitigate the event that we happen to compromise a device inside a corporate network that has an EDR agent running on it. There has been significant work on this in recent years that should enable effective defence evasion. Specifically, for our process we shall attempt to unhook `ntdll.dll`, the library mapping syscalls to programs, the hooking of which modern EDRs rely on in order to monitor syscalls. We will do this by leveraging Call Stack Spoofing, in the style leveraged by DodgeBox[1], which manipulates the call stack so that trusted processes appear to be the ones doing the DLL unhooking.
2. Delay deployment of the cryptominer. We want to decorrelate the installation of the malicious tooling and the deployment of the miner to avoid user suspicion in case there is a notable performance degradation. The miner should not operate until 10 days after deployment.

3. RandomX ships with a fast mode and a light mode[15]. Run the miner hot when the user is not directly interacting with their device. The process will look for mouse events, when there hasn't been one for 60 seconds RAGeQUIT will begin mining in fast mode. This should immediately downgrade to light in the event that there is another mouse event. The miner should also downgrade to light mode in the event the program `TaskManager.exe` is running.

Money

Revenue will be a projection with significant uncertainty. It will be a factor of the following items:

- Number of real user installs. We are targeting 20,000-50,000
- The hardware specification of the machines we infect, and accordingly the hashes/second they are capable of. We expect ∼1000 hashes/second on average. We think this figure could realistically be +/-20% in reality.
- A high proportion of the users who install running Windows. We expect 80-95% of users to be running Windows, with the majority of those remaining being Mac users, and a small sliver being Linux users.
- The price of Monero holding steady. As of 13/08/24 is stands at ∼150$. A glance at its price over the past year suggests it can reasonably be expected to fluctuate +/-20%
- How long the campaign lasts before it is taken down or meaningfully mitigated. We hope 12-24 months.

A lower bound estimate therefore gives us:

- 20,000 user installs
- 20,000 * 0.8 as a proportion of infected devices using Windows = 16,000
- 16,000 * 800 hashes/second = 12,800,000 total hashes/second across all agents
- 12,800,000 h/s translates to 0.089825 XMR/hour [17] assuming normal wattage (1.35kW)
- 0.089825 * 24 * 365 * ($150 * 0.8) = $94,400

An upper bound estimate gives us:

- 50,000 user installs
- 50,000 * 0.95 as a proportion of infected devices using Windows = 47,500
- 47,500 * 1200 hashes/second = 57,000,000 total hashes/second across all agents
- 57,000,000 h/s translates to 0.391322 XMR/hour [17] assuming normal wattage (1.35kW)
- 0.391322 * 24 * 365 * 2 * ($150 * 1.2) = $1,230,000

So we have a projected range of revenues that range by over a full order of magnitude. Now for the cost. The only significant ones to contend with are developer time, and GPT-4o-mini api requests. We will reasonably assume that local inference costs to run the content generation for the botfarm will number in the low thousands and can be disregarded. At $0.150 / 1M input tokens for GPT-4o-mini, we estimate 1 million tokens will account for the needs of a single user for 2 weeks. Meaning 20,000 real users will equate to $3000USD, and 50,000 real users will equate to $7500USD. With this estimate in hand we regard our costs as so low that we think this campaign will struggle not to make a profit provided estimates are reasonable. The only real question is whether it will be worth the developers time. This remains to be seen.

Mitigating RAGeQUIT

RAGeQUIT has the potential to fail in many ways. We will attempt to exhaustively enumerate them.

Imitation

The malware has been developed on a budget, such that it could conceivably be developed by one, relatively unskilled developer. To enable this it lifts from and copies from existing work in 4 notable places:

- The Chrome extension, lifted from the GitHub repo of mrdiamonddirt[7]
- The payload delivery mechanism and C2 channel are lifted from WARMCOOKIE[10]
- The privilege escalation vector is taken from BlackCat/ALPHV[9]
- The EDR defence evasion strategy is stolen from DodgeBox[1]

Behavioural detections in EDR systems are built for and distributed in response to the techniques leveraged in known malware campaigns. For instance, while it may have been easy to lift the privilege escalation methodology from BlackCat, abusing COM interfaces is a well known vector for privilege escalation, and would quite probably be detected as detections for similar campaigns are rolled out. Case in point, see this detection from Elastic seeking DLL sideloading via COM abuse for UAC bypass[2]. The same is true for the use of the DodgeBox methodology for EDR bypass. While impressive, and potentially very effective in the short run, it seems like a matter of time before detections for that kind of EDR bypass are devised, and RAGeQUIT will be wrongfooted when they are distributed widely.

Reach

Possibly the greatest risk to the RAGeQUIT campaign is that of a lack of reach. If it just doesn't reach enough users it will never come close to generating the amount of money needed to have made the campaign worthwhile. If the social media campaign induces fewer than 5000 real installs it seems unlikely to be worth continuing to bother to register the C2 domains.

C2 Beaconing

Even with significant effort having been made to obscure our C2 traffic, if it happens to travel over a corporate network there is every chance it could be noticed. This was notably the means by which Operation Triangulation was discovered by Kaspersky SOC Analysts, noting C2 traffic over the corporate WiFi network[6]. Leveraging cluster based anomaly detection, like that offered by the Elastic SIEM platform or the Darktrace network switch monitoring tool could easily highlight the anomalous nature of the traffic emanating from compromised hosts to the CDN domain.

CDN Provider and Law Enforcement

If engineers at Fastly become aware of the campaign they will be in a position to practically immediately take down C2 comms, by refusing to forward packets by what they have determined to be the next domain by reverse engineering the DGA of the actual Malware. If Fastly required encouragement, they are headquartered in the United States, and could conceivably be compelled to co-operate with a takedown operation led by a three letter agency. Along this line another proposal worth mooting would be to make CDN providers legally accountable if their networks host C2 traffic that they wilfully fail to detect. This suggestion has pitfalls and may be too onerous a burden on the providers, it may also prove too complex to enforce, with providers and the government haggling in court for potentially years over whether the provider took reasonable measures to detect something they ultimately missed.

Review processes

The Chrome Webstore[5] and OpenAI GPT Marketplace purportedly have review processes for their marketplaces. How they could step up their game would be to make the extensions ecosystem entirely hosted on their platforms (if there are additional web services running), and make the whole thing open source. Achknowledge that they have done too poor a job of auditing these applications historically, and let people decide for themselves whether an app is safe.

Cryptomining Fingerprinting

A fairly obvious detection to write at this stage might be an attempt at statistical pattern recognition applied to memory and CPU profiling. The behaviour and activity of CPU and memory when calculating thousands of hashes per second will be unmistakable, with the only plausible other similarly resource intensive activities being password hash cracking, ML model training, and ML model inference. With this knowledge in hand processes that ought to be performing these particular processes could be whitelisted, and any process whose memory and CPU profile appears to similar to that of one calculating lots of hashes should be flagged.

Domain Ownership Transparency

A significant part of campaigns such as this hinge on domains being able to be registered quickly and (in effect) anonymously. Laws could be drafted to oblige people to link their identities, as defined by government issued ID, more concretely to their ownership of a domain. In tandem, people could be held potentially criminally liable if through sheer negligence they allow their domain to be used for C2 traffic. Finally there could be legally mandated delays in the registration times for domains, to mitigate the kind of just-in-time C2 registration that was in use in this campaign.

We feel compelled to state that while these would all quite probably be somewhat effective in mitigating malware campaigns, they all represent onerous burdens on individuals and businesses, and hand significant power to national governments to administer and control the internet. A task which most who use the internet would probably agree they are poorly suited to, due to obvious conflicts of interest with regards controlling their populations.

Botfarm Takedown

A significant unknown in this campaign is whether the content generated by LLMs will be detectable, and therefore the botfarm will be taken down before the campaign can really get started. Tang et al. seem to think that methodologies exist, but that a steady progression towards LLM-generated content being entirely undetectable is inevitable[13]. The open question is whether our chosen model: Llama 3.1 8B, is already there. If not, then it is incumbent upon social media platform operators to leverage the detection methodologies cited by Tang et al., namely deep learning and basic perplexity scoring, to systematically root out generated content on social media, particularly when it appears to be pushing some kind of product.

Conclusion

RAGeQUIT as a campaign is deeply flawed. It is chop shop job comprised of a half dozen other pieces of code and malware with no clear sense of identity. Its heavy reliance on the work of others will likely be its eventual downfall when it is inevitably deployed into a corporate network and spotted via behavioural detection doing something a different piece of malware was seen doing. The question is whether a slap-dash job of this kind can still turn a decent profit. We think yes, but it may be very limited in how lucrative it can be, and the operator may well be better off just becoming a well paid security engineer.

References

[1] Y. H. Chang and S. Singh. Dodgebox - call stack spoofing. https://www.zscaler.com/blogs/security-research/dodgebox-deep-dive-updated-arsenal-apt41-part-1.

[2] Elastic. Com interface uac bypass detection. https://github.com/elastic/detection-rules/blob/e607d521b8c8db81fa88698e77a9a6a95e5cab44/rules/windows/privilege_escalation_uac_bypass_dll_sideloading.toml#L17.

[3] H. I. et al. Llamaguard. https://ai.meta.com/research/publications/llama-guard-llm-based-input-output-safeguard-for-human-ai-conversations/.

[4] S. W. et al. Cyberseceval3. https://arxiv.org/pdf/2408.01605v1.

[5] Google. Chrome webstore review process. https://developer.chrome.com/docs/webstore/review-process/.

[6] Kaspersky. Connecting the dots: Kaspersky reveals in-depth insights into operation triangulation. https://www.kaspersky.com/about/press-releases/2023_connecting-the-dots-kaspersky-reveals-in-depth-insights-into-operation-triangulation.

[7] mrdiamonddirt. Local llama chrome extension. https://github.com/mrdiamonddirt/local-llama-chrome-extension.

[8] OpenAI. Openai pricing. https://openai.com/api/pricing/.

[9] S. Scorecard. Blackcat/alphv deep dive. https://securityscorecard.com/research/deep-dive-into-alphv-blackcat-ransomware/.

[10] D. Stepanic. Warmcookie, 2024. https://www.elastic.co/security-labs/dipping-into-danger?trk=feed_main-feed-card_feed-article-content.

[11] K. Subramani, R. Perdisci, and P. Skafidas. Measuring cdns susceptible to domain fronting. https://arxiv.org/pdf/2310.17851.

[12] tackker. Keyboard history recorder. https://chromewebstore.google.com/detail/keyboard-history-recorder/igbodamhgjohafcenbcljfegbipdfjpk.

[13] R. Tang, Y.-N. Chuang, and X. Hu. The science of detecting llm generate text. https://cacm.acm.org/research/the-science-of-detecting-llm-generated-text/.

[14] TeamT5. Pypi domain fronting. https://teamt5.org/en/posts/hiding-in-plain-sight-obscuring-c2s-by-abusing-cdn-services/.

[15] tevador. Randomx. https://github.com/tevador/RandomX.

[16] D. Via and S. Runnels. Bits abuse. https://cloud.google.com/blog/topics/threat-intelligence/attacker-use-of-windows-background-intelligent-transfer-service.

[17] WhatToMine. Xmr - whattomine. https://whattomine.com/coins/101-xmr-cryptonight?hr=12800&p=1350.0&fee=0.0&cost=0.1&cost_currency=USD&hcost=0.0&span_br=&span_d=24&commit=Calculate.