STONERHINO C2

NEXT-GENERATION MOBILE PHONE COMPATIBLE COMMAND-AND-CONTROL (C2) POST-EXPLOITATION FRAMEWORK

Chang Tan

UNIVERSITY OF NEVADA, LAS VEGAS | 4505 S. MARYLAND PKWY

Contents

Introduction

STONERHINO is a Next-Generation Post-Exploitation and Command and Control (C2) Framework. It implements multiple technologies not found in the "gold standards" of red-teaming tools as vaguely covered in the background section, and brought into further detail in the Prototype, Implementation, Theoretical and Business Justifications section.

STONERHINO is probably the first C2 Framework of its kind with a programming/build/implementation process that enhances the security of trade secrets and tradecraft by terminating the attribution link with invasive technologies such as OpenTelemetry in higher-level programming languages such as Golang1, C#, and Rustlang. The entire framework, from post-exploitation implant/agent, Teamserver, to the client is written in the following languages...

1. C/C++ for implants that are cross-platform and can target Windows, Linux, MacOS, as well as embedded Internet-of-Things (IoT) Devices
2. C++ for Teamserver
3. Exported NodeJS packages (that are not documented) that serve as a wrapper around the Open-Source Google Chrome V8 JavaScript Engine, known as "C++ Addons".
4. REACT Web GUI Front-End for Security

Background

Command and Control Frameworks are an emerging and rapidly proliferating market utilized by legitimate penetration testers/red-teamers, as well as adversaries. Currently the market is fixated on bypasses of Endpoint Detection and Response (EDR) or Extended Detection and Response (XDR). This is a continuous cat-and-mouse game, but we personally feel that the developers/maintainers of post-exploitation framework has failed to foresee the future of the world wide web.

Alpha tests of components of STONERHINO, written by the author, Chang Tan, has proven to be more effective at evading AV/EDR/XDR Solutions than commercial or open-source products (Tan, IS 330 Project Mixed Boolean Payload versus Meterpreter, 2023). Furthermore, the author is trained to evade Endpoint Detection and Response as well as Extended Detection and Response using his training in programming using the Windows API from both the Mosse Institute of Australia and Sektor7 Research Institute, which is proven, in another YouTube Video (Tan, IS330 Project Windows Evasion Malware, 2023).

Currently, STONERHINO can evade the majority of AV/EDR/XDR Solutions, and can run raw shellcode as an encrypted blob through API-hooking when relying on shellcode from other C2 Frameworks. The author, Chang Tan, also is looking to implement the anti-reverse

---

[1] While the NSA has recommended memory-safe languages to replace more "unsafe" languages, the Google Golang Forced Opt-In Telemetry proposal on GitHub threatens to turn many developers away from the language. Golang was notoriously used as the Command and Control of the Mirai Botnet, as well as the Teamserver implementation in Brute Ratel and Havoc C2

engineering and analysis features that are meant for STONERHINO Implants. While employees from companies like Mandiant have trouble identifying STONERHINO Implants and continues to evade modern detection solutions to this day, the author has managed to prove that the implants are indeed malware by using Dynamic Binary Instrumentation tools such as Intel PIN (Tan, IS330 Catching STONERHINO Payloads with Intel PIN (TinyTracer), 2023).

Certain Web 3.0 Technologies such as the Interplanetary File System do have a use case, as well as an abuse case, as indicated by Cisco Talos Group, when malware and phishing campaigns began appearing around November 2022. Furthermore, as the Ukraine War continues to escalate, adversarial nations have begun "Balkanizing" their own IP address ranges, effectively "unplugging" their sovereign networks from the "internet", in what is known as, "The Splinternet". This is well documented in Russia, China, Iran, and North Korea.

STONERHINO attempts to address all of these issues and trends by having cutting edge modules that are prepared for the future threat landscape as a adversary simulation framework. Not only has certain proof-of-concepts have been available, but the ability to "breach" the "splinternet" without relying on Invisible Internet Project (I2P) or The Onion Router (TOR), has existed for at least half a decade. This includes, but not limited to, malicious JavaScript Service Workers that can persist inside of a target's browser and function as a network proxy to reach the formerly "unreachable" IP address ranges by impersonating the machine of the target who registered the Service Worker. Such technology exists since 2019.

Furthermore, STONERHINO as a Command and Control Framework is fixated on improving tradecraft of penetration testers/red-teamers by using Natural Language Processing Capabilities through a mobile web GUI web application for operators to direct commands with "fake phone calls". Thus, it reduces the attribution to the stereotypical "hacker in a hoodie with a laptop". Commands to poll the remote agent/implant, load modules or download additional stages or cleanly exit itself can be implemented with voice commands and parsed with machine learning technologies. STONERHINO comes preconfigured with its own default commands to be combined in false phone conversations, and is a optional feature that can be activated after requesting the operator to record their own voice for specific "activation or command keywords".

Another feature of STONERHINO, not found in other Post-Exploitation Frameworks is the proliferation of "facial recognition" bypasses. In 2020-2021, the Wall Street Journal has reported that the latest target for hackers is bypassing facial recognition ID, such as Apple's Face ID. It is a rapidly growing arms race, and Apple has proven resilient in defeating such attacks. Android on the other hand, has questionable protection against "face-hacking", and this technique will be actively researched by our dev teams to add as an experimental module (Olson, 2021).

Storyboarding Process

Empathy Map

Brian Halbach is a 37-year old Professional Red Teamer/Security Consultant with InGuardians Red Team from Washington D.C. He has been on multiple podcasts, including DarkNet Diaries hosted by Jack Rhysider in Episode 95 "Brian and Jon's Big Adventure".
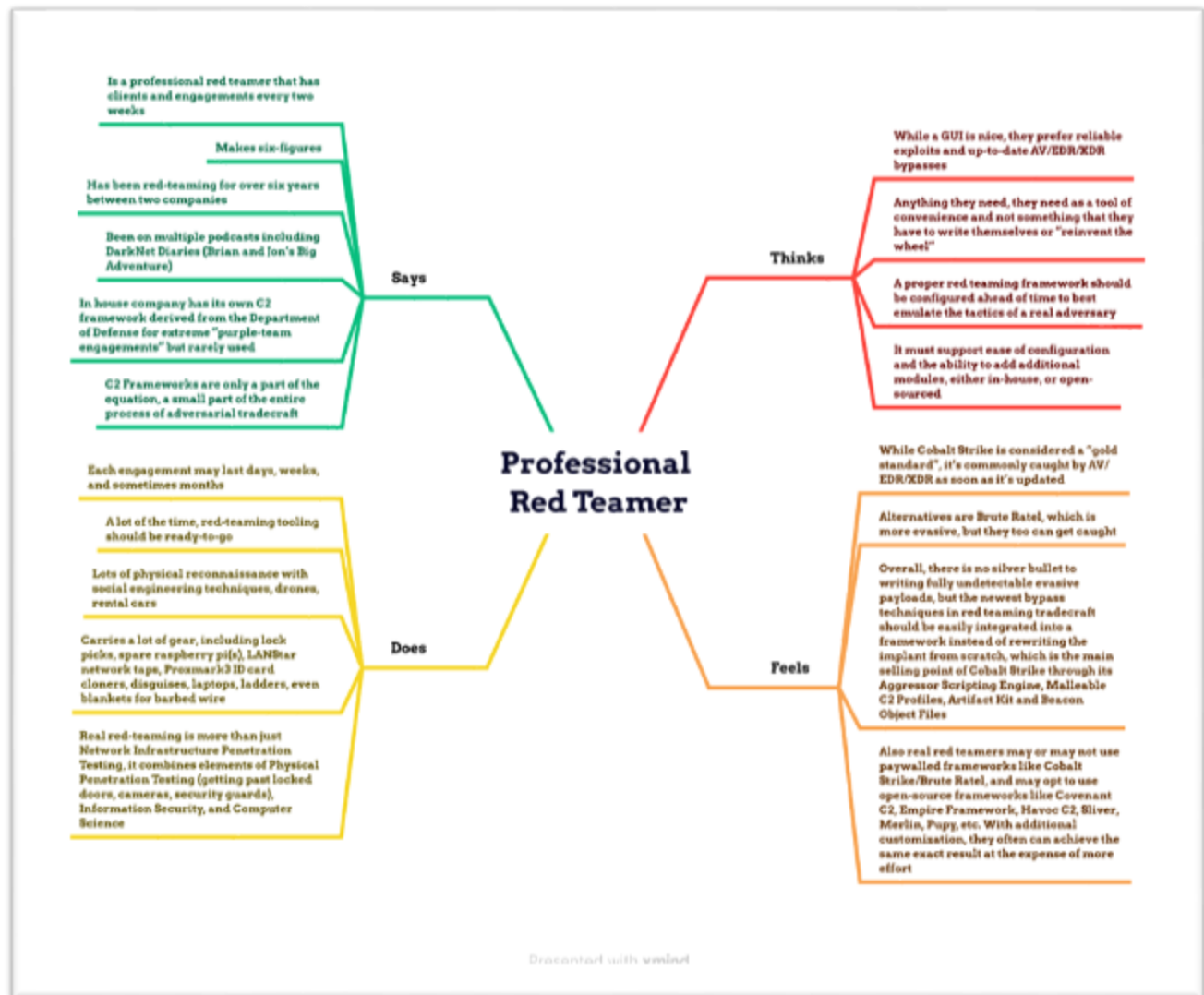
Brian has engagements every few weeks, usually lasting around two weeks and flies all across the United States to engage in them. Engagements may last days, weeks, and in some rarer cases, months. Brian says that C2 Frameworks are only a part of the equation of the entire process of adversarial tradecraft.

He expects a lot of red-teaming tooling to be ready-to-go out of the box. As a member of the "Red Team", he combines physical reconnaissance with social-engineering techniques, drones, rental cars and disguises. He carries with him a lot of commercial and custom gear, including lock-picking tools, spare raspberry pi(s) with custom offensive security software

installed inside of them, LANStar Network Taps, Modified Proxmark3 ID Card Cloners with amplified antennas and batteries, laptops, ladders, and even wool blankets for barbed wire.

While he feels that a Graphical User Interface (GUI) is nice, he feels that reliable exploits and up-to-date AV/EDR/XDR bypasses are a priority. Anything they need, they need it as a tool of convenience without having to "reinvent the wheel". A proper red-teaming framework must be preconfigured to emulate a actual adversary. Overall there is no silver-bullet to writing undetectable payloads but the latest bypass techniques for AMSI (Anti-Malware Scan Interface), Defender or Custom AV Products, EDR/XDR must be modular and not have to be reintegrated and rewritten from scratch.

Creating Personas

User Persona Name

Devices Used

- Lockpicks, ladders, wool blankets, Rental cars, drones, Faraday bags, Highly modified Raspberry Pis, Disguises
- $12,000 Dell Precision Mobile Workstation Laptops
- LANStar Network Taps & Proxmark3 ID Badge Scanners with extra lithium ion batteries and amplified antennas

Goals

-
- Easy to configure Command & Control (C2) spun up in the cloud
- Reliability of exploit success and Evasive implants
- High availability of the C2 Cloud
- Innovative new techniques in a ever changing threat landscape (considering the emergence and adoption of Web 3.0 Technologies and how it is already been used by actual adversaries, and that these tactics, techniques, and procedures must be emulated)
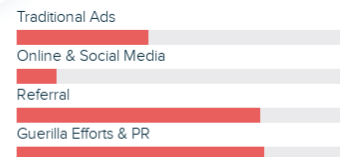
Frustrations

-
- A lot of default payloads get caught and require constant modification
- Currently, there are coworkers in his company that works on these modifications, such as AMSI bypasses and MOTW bypasses
- Adversarial tradecraft is constantly evolving, especially at the rates that defenders are patching or releasing new security updates
- To write truly evasive payloads, extensive obfuscation must be used on the Artifact Kit Templates, which in turn increases entropy which must be reduced as well as a legitimate code signing certificate to bypass Potentially Unwanted App Checks ("SmartScreen"), SSL/TLS traffic should be modified with a Malleable C2 Profile, and all payloads must be extendable/integrated with red teaming standard tools like Mimikatz, Rubeus, GhostPack Binaries and open source toolkits such as sshuttle, rpivot, and chisel for pivoting as well as LOLBin techniques like netcat/socat relays and netsh portproxy relays

"I am a experienced security consultant with a focus on network, application, wifi, and physical penetration testing."

Age: **37**
Work: **Senior Security Consultant**
Family: **In a relationship**
Location: **Minneapolis, MN**
Company: **InGuardians Red Team**

Personality

Introvert	Extrovert
Thinking	Feeling
Sensing	Intuition
Judging	Perceiving

Motivation

- Incentive
- Fear
- Growth
- Power
- Social

Preferred Channels

- Traditional Ads
- Online & Social Media
- Referral
- Guerilla Efforts & PR

Technical Experience

- Former Cisco Network Engineer
- Does not require certifications, and was taught tradecraft by his first company, Red Team Security (the same company that had a engagement with a nuclear power plant on YouTube)
- Well versed in information technology, learning software engineering
- Taught in stealth, lockpicking/physical security
- Frequent player of HackTheBox APT Labs and RastaLabs

Red-Teaming Experience

- He worked alongside former US Army Airborne Operators in his first company, Red Team Security
- Featured in DarkNet Diaries, interviewed by Jack Rhysider in Episode 95: Jon & Brian's Big Adventure

DO NEXT 👉 **Customer Journey Map**

See related templates

Creating Scenarios and Storyboards

Seamless Spin-Up to the Cloud

• Using AWS CloudFormation's "Infrastructure as Code" XML Format, new C2's can be auto provisioned as a single command

Easily Extendable with other Red-Teaming Tools with a Scripting Engine

• Similar to Cobalt Strike's Aggressor Scripts, the STONERHINO Framework can easily be extended. STONERHINO Scripts support..

• Python
• JavaScript
• Java
• Ruby
• Bash
• Powershell

Creating Scenarios and Storyboards

Adapted to the Modern Adversarial Environment

• Operators will no longer be attributed to the "hacker-in-the-hoodie" stereotype. They dress now in casual clothing to suits. Commands will be given through the Natural Language Processing Engine through fake phone calls via a teleconference web app.

Prototype

1. STONERHINO Command and Control Login Screen from Web GUI	2. Example Web GUI after login, including the "Voice Command Mode"	3. The Voice Command Mode uses Natural Language Processing via a fake phone call through a audio teleconference web application to send commands to implants and the Teamserver

Implementation Plan

Framework Design

The implementation of the entire C2 Framework is very similar to our market competitors such as Cobalt Strike (Java + C), BruteRatel C4 (C/C++ with Qt Framework), or Havoc C2 (C/C++ and Golang).

However, because of the debate over Google Golang's implementation of mandatory opt-in telemetry and privacy concerns, we have switched to using a combination of…

1. C/C++ for the implants
2. C++ and NodeJS for the Back-End Teamserver
3. REACT Front-End For The Browser-Based Graphical User Interface

Many of the higher-level features can be "abstracted" from C/C++ using existing NodeJS NPM Modules, by building a wrapper around the C++ Teamserver code using node-gyp (or generate-your-own project).

We have reverse-engineered features of Cobalt Strike from a cracked copy release (4.4), and quickly learned that HelpSystems follows the same method of using abstracted C/C++ code by wrapping it with a JNI Wrapper in Java. Specifically, one of the modules known as "CovertVPN", which clones the hardware MAC address of the implant's (beacon) target and requires a privileged user-vector (Administrator and up) to work.

Feature: GOLIATHSCORPION: Anti-Reverse Engineering

GOLIATHSCORPION is a retaliatory, anti-reverse engineering framework to make it more difficult for analysts to document the implant. A working proof-of-concept is already compiled targeting generic Linux Distributions such as Ubuntu-based Malware Analysis Virtual Machines such as RemNUX from the SANS Institute (Tan, Analyst-Punish, 2023). As of right now…

1. It is capable of targeting and frustrating all Debian-based Linux Distributions used by analysts
2. It can cause kernel panics (hard crashes), disable kernel-logging, and punish analysts by deleting the analyst's root directory depending on the vector they are analyzing the malware with (superuser or regular)

A Windows version of GOLIATHSCORPION is in the works, it's main feature is the ability to cause intentional BSOD (Blue Screens of Death) using a undocumented NT Kernel API Feature known as "NtRaiseHardError" from NTDLL.DLL (NTAPI Undocumented Functions , 2023). And proof-of-concept code was released on GitHub in 2016, making this a ideal method for punishing analysts utilizing Windows-based Analyst Virtual Machines such as FlareVM and CommandoVM (jpiechowka, 2016).

Feature: Rogue-Byte Interweave Obfuscation

Even if GOLIATHSCORPION is compromised or patched-out using static analysis tools, analysts also have to contend with the "Rogue-Byte Interweave" method of malware obfuscation.

The idea is that before dynamic analysis can begin, analysts must patch out the anti-debugging and analysis tricks that threat actors are using. This is a common technique to frustrate junior analysts and is commonly found in-the-wild in real malware samples.

Inserting "rogue bytes", that is, invalid assembly instructions, can break static analysis tools such as IDA or GHIDRA, and other disassemblers/decompilers while allowing the malware to execute normally. For our proof-of-concept, our position-independent shellcode is "interweaved" with short-jump instructions (a short JMP takes two operands, the JMP and the distance in bytes, up to 127 bytes up and down the stack) that are thousands of JMPs long.

So adding 1,000 short JMPs to allow the shellcode to remain executable, jumping between padded rogue-bytes of up to 126 bytes means a maximum jump distance of 127 bytes + 2 for the JMP instruction and another byte for the distance. Note that I said "maximum", that means we can randomize the JMPs (positive JMP) first down the stack, and then a random JMP back up the stack, with random rogue bytes padded between each JMP. Shellcode allocated to the heap must perform a negative JMP to move up the heap, and then begin the interweave.

Padding rogue-bytes with the interweave attack 1,000 times means $(127+2)*(10^3)$ bytes, or almost 126 kilobytes of garbage code that the analyst must patch out iteratively before they discover the anti-debugging tricks, and, of course, the GOLIATHSCORPION weapons.

By combining the Rogue-Byte Interweave Attack with Tampering Detection and Anti-Debugging Tricks, you successfully concealed the GOLIATHSCORPION sub-framework from being easily detected, meaning each variant of the STONERHINO C2 Implants are increasingly more difficult to reverse.

Feature: Stack-Machine Obfuscation *(Virtual Machine Obfuscation)*
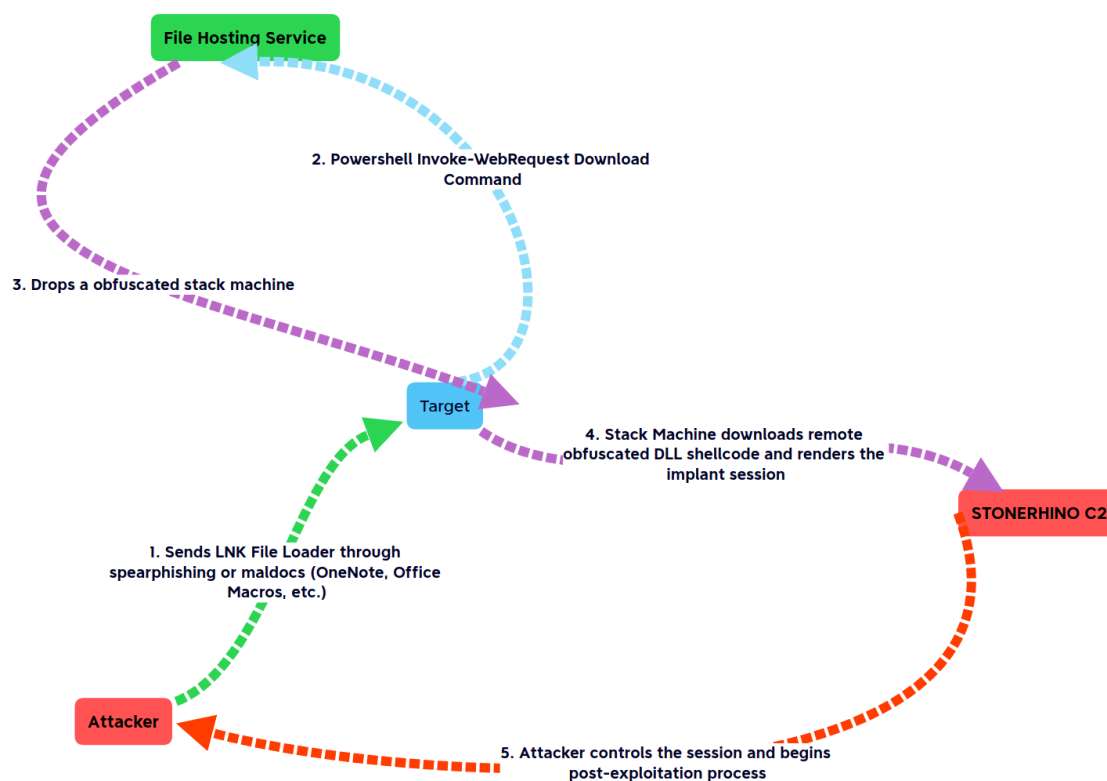
1. LNK file loader runs a powershell command via Invoke Web-Request to download a stack machine
2. The stack machine checks for sandboxes before running virtualized instructions
3. Virtualized instructions call methods from the WINHTTPAPI or Powershell commands to download a encrypted and base85 encoded shellcode file
4. The stack machine exits virtualization mode and allocates the decoded and decrypted shellcode onto its own shellcode runner and runs it
5. The shellcode returns a implant session and with additional commands, persists itself as a user land rootkit

LNK Loader:

      A reversed initial stage loader from Bumblebee that uses the Invoke-WebRequest Powershell Command to remotely download the stack machine without Mark-of-the-Web, a Alternate Data Stream Marker that indicates that the payload was remotely downloaded from a public IP address (MalwareBazaar Database, 2023).

Stack Machine:

It first checks for sandbox environments, by pretending to print strings while checking conditions before it escapes the sandbox and enters virtualization mode (MCD - MCSI Certified Code Deobfuscation Specialist, 2023). TrustedSec has a very commonly used list of Windows API functions that can be used to detect executable sandboxes, specifically targeting Microsoft Hyper-V based sandboxes for Defender, including but not limited to, counting the amount of threads in the sandbox, checking against RAM allocations of the sandbox, inspecting newly created files, checking uptime of the sandbox environment, and checking whether or not the sandbox is domain-joined (Lakhan, 2018).

Once the payload escapes the sandbox, it virtualizes all arithmetic and bitwise operations to deobfuscate the PowerShell download command via a COM object from a remote server and reads it into a buffer. Alternatively, the download command can be used by directly calling the WinHTTP API in C++ (Using the WinHTTP C/C++ API, 2021). And then decodes it from base85, then decrypts it before allocating it the stack/heap (jhackz, 2020). The stack machine uses a custom self-instrumentation method to detect dynamic analysis tools, debuggers, and tampering as suggested by the authors of Surreptitious Software from the University of Arizona (Collberg, 2009).

Shellcode Runner:

Standard shellcode runner that decrypts and runs the full implant, by either, allocating it to its own stack/heap, or unhooking the EDR/XDR hooks from a fresh copy of ntdll.dll in order to inject into the process (migration) by Windows API-hooking (RED TEAM OPERATOR: Malware Development, 2020).

This solves multiple problems

1. Keeps the shellcode away from disk.
2. Lowers entropy, a key factor in suspicious payloads.
3. The second stage stack machine is relatively benign and small in size and really doesn't do anything suspicious until it confirms it escapes the sandboxing environment.
4. Self-instrumentation would help cover all areas of "attack" from the analyst outside of side-channel attacks, while being compatible with all platforms and operating systems (so it's not confined to targeting specific DBI tools or debuggers like WinDBG, SoftICE, OllyDbg, x32/x64dbg)

This method would not be suspicious at all, and additional modifications of the final stage DLL shellcode can be modified at will to add/remove features.

Since the three components, the LNK file, stack machine, and remote DLL shellcode are three separate components, each stage can be modified with different loaders, implants, and infection chains to make maintaining the C2 framework consistent and fast to adapt against defenders.

Feature: CORSAIR: Splinternet/Intranet Attacks

CORSAIR is a feature of STONERHINO that allows it to perform "splinternet attacks", or attacking isolated intranets. It's based off the Shadowworkers C2 Framework to create obfuscated malicious JavaScript service-workers that lives in the browser, known as the "Man-in-the-Browser Attack" (libnex, 2023).

By compromising a target with network access to an isolated network, the service-worker functions as a underprivileged proxy for attackers to access other parts of the internal network or intranet. Such features have existed before in other penetration testing frameworks such as Beef-XSS (Browser Exploitation Framework), but the Man-in-the-Browser implants did not persist when the target closes the webpage tab or closes their browsers. Thanks to malicious service-workers, this capability to persist is possible as long as the target leaves the browser on. The tab does not need to remain open, only the infected browser has to be running.

CORSAIR is designed to be delivered using malicious webservers utilizing HTTPS (Web 2.0) as well as abusing emerging Web 3.0 Technologies, specifically the IPFS (InterPlanetary File System) and Nostr.io Protocols (Wandersleb, 2023). IPFS has native integration with NodeJS with it's own npm package to spin up your own IPFS Node and requires either a Web 2.0 to 3.0 Gateway, or as more browsers support it, Native IPFS Support, which no longer requires a IPFS Gateway to serve payloads.

Feature: Voice Activated Commands

One of the most hyped features of STONERHINO C2 as shown in the storyboard is Voice-Activated Commands utilizing Natural Language Processing. At a push of a button from the Web GUI of the Teamserver, a fake phone conversation is started with pre-programmed voice commands. Attackers now are disassociated with the stereotypical "hacker-in-a-hoodie" stereotype, and can send commands to their Teamserver implants with a fake phone conversation.

These voice-commands can easily be configured, and Voice Activated Commands is a optional feature that must be switched on, so that the operator's voice can be matched to each command word or phrase. Instead of typing commands behind a laptop or cell phone, the operator literally "talks to the wall", or more specifically, the Teamserver, to update polling of implants, load malicious DLLs, exfiltrate data, or terminate them, if needed.

Feature: FACEDANCER: Facial Recognition Hacking

According to this Wall Street Journal Article "Faces Are The Next Target For Fraudsters", published on July 7th, 2021, hacking Face ID technologies is a hot topic. Our team is still working on this, but generally speaking, Android Facial Recognition is not as secure as Apple's implementation (Olson, 2021).

We hope to introduce what we call, FACEDANCER, in a later release of STONERHINO C2.

Justification for the Innovation

We justify these innovations within two categories, theoretical and business-oriented.

Theoretical Justifications

In true adversarial tradecraft by nation-state threat actors, innovation is measured by how it is implemented, such as virtual-machine (stack-machine) obfuscated malware and how it is delivered through a attack chain (i.e. Bumblebee, Royal Ransomware, SmokeLoader). Furthermore, we did not come up with the abuse of Web 3.0 Technologies to deliver malware and phishing campaigns. This was already documented by true threat actors in the wild by Cisco Talos Group (Brumaghin, 2022).

In other words, a lot of academic "theory", has readily been adopted by true threat actors, whether they be nation-state or financially motivated. We need to catch up to emulate actual threat actors.

Business Justifications

By implementing these technologies, we produce a Command-and-Control Post-Exploitation Framework (C2) that would represent the new Gold Standard in an ever increasingly competitive market between Open-Source (Havoc C2, Covenant, or Sliver) and Proprietary (Cobalt Strike or Brute Ratel). Many of these features, particularly the ones that make static and dynamic analysis more difficult and the transformation of adversarial tradecraft (Natural Language Processing Capabilities) have not been implemented in current releases yet.

We hope to redefine the adversarial simulation landscape with STONERHINO C2.

Conclusion

STONERHINO C2 is a Next Generation Adversarial Framework that takes existing and emerging concepts, and dramatically improves upon the design, while adding multiple new innovations that are seen by actual threat actors in-the-wild. The information we received from actual red teamers contributed a lot to the design of STONERHINO.

Note that the mentioned innovations (GOLIATHSCORPION, CORSAIR, Rogue-Byte Interweaves, and Natural Language Processing) are only the proposed and partially implemented ideas that we provided. We may choose to expand further into other ideas and concepts to continually implement what was formerly theory, into actual useable tools to keep up with true threat actors.

STONERHINO C2 strives to be the next "Gold Standard" of Adversarial Frameworks by implementing not just what is cited in academia, but by emulating the latest Tactics, Techniques, and Procedures (TTPs) of actual adversaries. We hope that this proposal and partially implemented design will be a yardstick for our future releases and our competitors combined.

Works Cited

Alcorn, W. (2023, April 17). *Browser Exploitation Framework*. Retrieved from Github: https://github.com/beefproject/beef

Brumaghin, E. (2022). Threat Spotlight: Cyber Criminal Adoption of IPFS for Phishing, Malware Campaigns. *Talos Intelligence Threat Spotlight*.

Collberg, C. (2009). *Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection: Obfuscation, Watermarking, and Tamperproofing for Software Protection* . Tucson, AZ: Addison-Wesley Software Security.

jhackz. (2020, April 16). *RTO-Implant*. Retrieved from Github: https://github.com/jhackz/RTO-Implant/blob/master/RTO%20Implant/z85.c

jpiechowka. (2016, April 3). *Quick-BSOD*. Retrieved from Github: https://github.com/jpiechowka/quick-bsod/blob/master/src/QuickBSOD.cpp

Lakhan, H. (2018, June 19). *Enumerating Anti-Sandboxing Techniques*. Retrieved from TrustedSec: https://www.trustedsec.com/blog/enumerating-anti-sandboxing-techniques/

libnex, c. a. (2023, April 17). *Shadowworkers*. Retrieved from Github: https://github.com/shadow-workers/shadow-workers

MalwareBazaar Database. (2023, May 6). Retrieved from Malware Bazaar Abuse.ch: https://bazaar.abuse.ch/sample/ac8e67644d7b6b6f0bd78522a3568c98fe386a23542f73a2ec1a3cff4f433684/

MCD - MCSI Certified Code Deobfuscation Specialist. (2023, May 5). Retrieved from Mosse Institute: https://www.mosse-institute.com/certifications/mcd-certified-code-deobfuscation-specialist.html

NTAPI Undocumented Functions . (2023, April 17). Retrieved from undocumented.ntinternals.net: http://undocumented.ntinternals.net/index.html?page=UserMode%2FUndocumented%20Functions%2FError%2FNtRaiseException.html

Olson, P. (2021, July 7). Faces Are The Next Target For Fraudsters. *Wall Street Journal*, p. 5.

RED TEAM OPERATOR: Malware Development. (2020, August 20). Retrieved from Sektor 7 Institute: https://institute.sektor7.net/

Tan, C. (2023, February 20). *Analyst-Punish*. Retrieved from Github: https://github.com/tanc7/analyst-punish

Tan, C. (2023, May 3). *IS 330 Project Mixed Boolean Payload versus Meterpreter*. Retrieved from YouTube: https://youtu.be/iMunfdFvoM8

Tan, C. (2023, May 5). *IS330 Catching STONERHINO Payloads with Intel PIN (TinyTracer)*. Retrieved from YouTube: https://youtu.be/wE3PlMCs_70

Tan, C. (2023, May 5). *IS330 Project Windows Evasion Malware*. Retrieved from YouTube: https://youtu.be/pQWpWvqhiB8

Using the WinHTTP C/C++ API. (2021, January 7). Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/windows/win32/winhttp/using-the-winhttp-c-c---api

Wandersleb, L. (2023, April 17). *Nostr-Protocol*. Retrieved from Github: https://github.com/nostr-protocol