Apr 01 2024

NETMANAGEIT

Intelligence Report

Stories from the SoC Part 1: IDAT Loader to BruteRatel

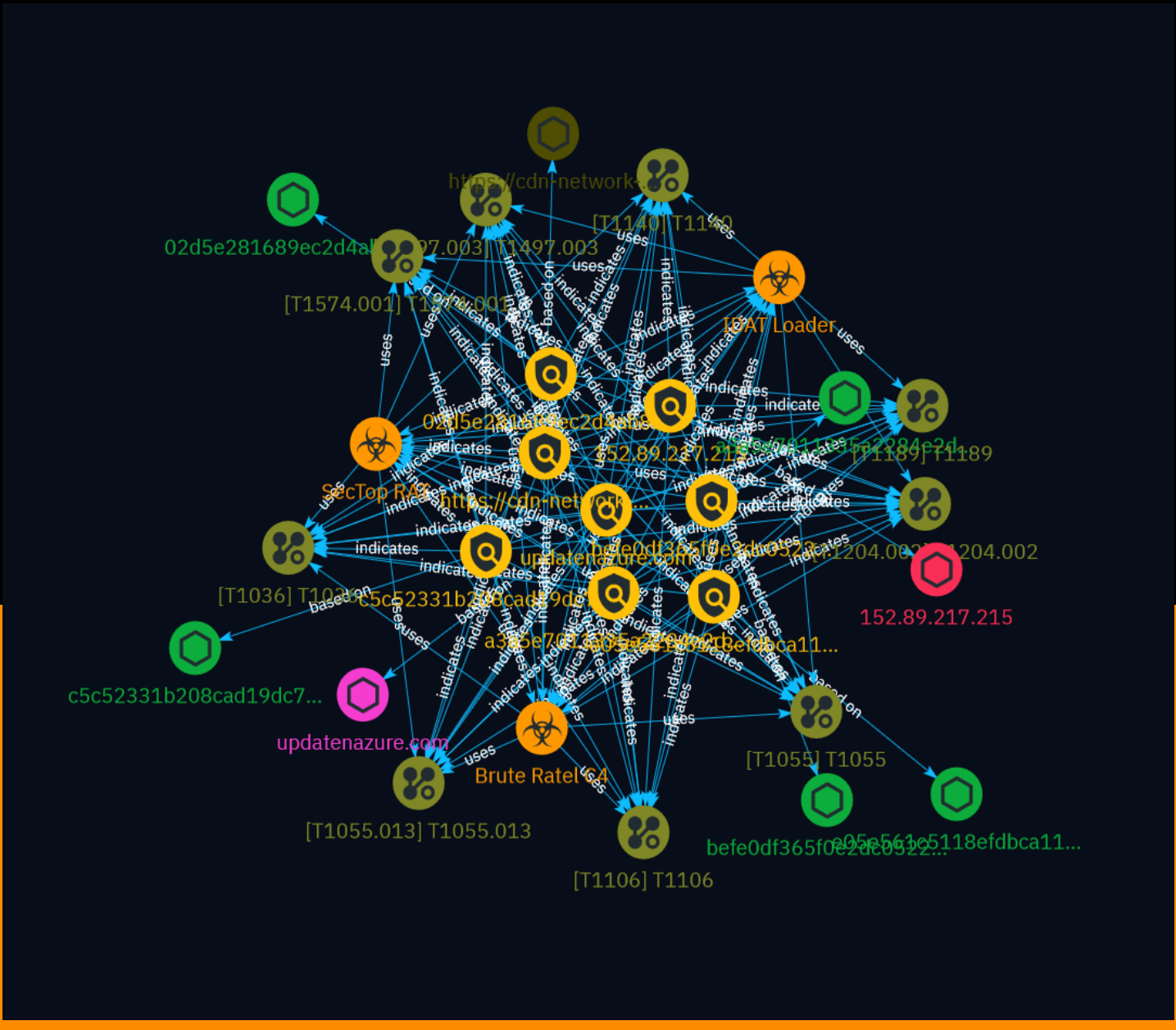

Table of contents

Overview

Entities

Observables

External References

Overview

Description

This report provides an analysis of a recent malware campaign that begins with a drive-by download of a Rust binary, which then loads the IDAT malware loader. The IDAT loader injects the SecTop RAT, followed by deployment of the Brute Ratel C4 framework for command and control. Technical details are provided on the tactics, techniques and procedures used at each stage of the attack.

Confidence

This value represents the confidence in the correctness of the data contained within this report.

100 / 100

Content

N/A

Indicator

Name

updatenazure.com

Description

- **Unsafe:** False - **Server:** N/A - **Domain Rank:** 0 - **DNS Valid:** True - **Parking:** False - **Spamming:** False - **Malware:** False - **Phishing:** False - **Suspicious:** True - **Adult:** False - **Category:** N/A - **Domain Age:** {'human': '2 months ago', 'timestamp': 1705661589, 'iso': '2024-01-19T05:53:09-05:00'} - **IPQS: Domain:** updatenazure.com - **IPQS: IP Address:** 205.234.231.176

Pattern Type

stix

Pattern

[domain-name:value = 'updatenazure.com']

Name

https://cdn-network-services-001.com/update/minor/1/release.json

Description

- **Unsafe:** False - **Server:** N/A - **Domain Rank:** 0 - **DNS Valid:** True - **Parking:** False - **Spamming:** False - **Malware:** False - **Phishing:** False - **Suspicious:** True - **Adult:** False - **Category:** N/A - **Domain Age:** {'human': '3

months ago', 'timestamp': 1705321708, 'iso': '2024-01-15T07:28:28-05:00'} - **IPQS: Domain:** cdn-network-services-001.com - **IPQS: IP Address:** 193.233.132.129

Pattern Type

stix

Pattern

[url:value = 'https://cdn-network-services-001.com/update/minor/1/release.json']

Name

152.89.217.215

Description

- **Zip Code:** N/A - **ISP:** LLC Smart Ape - **ASN:** 56694 - **Organization:** LLC Smart Ape - **Is Crawler:** False - **Timezone:** Europe/Moscow - **Mobile:** False - **Host:** data.amtp.cloud - **Proxy:** True - **VPN:** True - **TOR:** False - **Active VPN:** False - **Active TOR:** False - **Recent Abuse:** True - **Bot Status:** False - **Connection Type:** Premium required. - **Abuse Velocity:** Premium required. - **Country Code:** RU - **Region:** Moskva - **City:** Moscow - **Latitude:** 55.7522583 - **Longitude:** 37.61547089

Pattern Type

stix

Pattern

[ipv4-addr:value = '152.89.217.215']

Name

e05e561c5118efdbca113ca231c527b62e59a4bffae3bd374f7b4fcdd10e7d90

Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = 'e05e561c5118efdbca113ca231c527b62e59a4bffae3bd374f7b4fcdd10e7d90']
Name
c5c52331b208cad19dc710786e26ac55090ffca937410d76c53569d731f0bb92
Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = 'c5c52331b208cad19dc710786e26ac55090ffca937410d76c53569d731f0bb92']
Name
befe0df365f0e2dc05225470e45fdf03609f098a526d617c478b81ac6bb9147f
Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = 'befe0df365f0e2dc05225470e45fdf03609f098a526d617c478b81ac6bb9147f']

Name
a3a5e7011335a2284e2d4f73fd464ff129f0c9276878a054c1932bc50608584b
Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = 'a3a5e7011335a2284e2d4f73fd464ff129f0c9276878a054c1932bc50608584b']
Name
02d5e281689ec2d4ab8ac19c93321a09113e5d8fa39380a7021580ea1887b7a5
Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = '02d5e281689ec2d4ab8ac19c93321a09113e5d8fa39380a7021580ea1887b7a5']

Malware

Name
SecTop RAT
Name
IDAT Loader
Name
Brute Ratel C4

Attack-Pattern

Name
T1189
ID
T1189
Description
Adversaries may gain access to a system through a user visiting a website over the normal course of browsing. With this technique, the user's web browser is typically targeted for exploitation, but adversaries may also use compromised websites for non-exploitation behavior such as acquiring [Application Access Token](https://attack.mitre.org/techniques/T1550/001). Multiple ways of delivering exploit code to a browser exist (i.e., [Drive-by Target](https://attack.mitre.org/techniques/T1608/004)), including: * A legitimate website is compromised where adversaries have injected some form of malicious code such as JavaScript, iFrames, and cross-site scripting * Script files served to a legitimate website from a publicly writeable cloud storage bucket are modified by an adversary * Malicious ads are paid for and served through legitimate ad providers (i.e., [Malvertising](https://attack.mitre.org/techniques/T1583/008)) * Built-in web application interfaces are leveraged for the insertion of any other kind of object that can be used to display web content or contain a script that executes on the visiting client (e.g. forum posts, comments, and other user controllable web content). Often the website used by an adversary is one visited by a specific community, such as government, a particular industry, or region, where the goal is to compromise a specific user or set of users based on a shared interest. This kind of targeted campaign is often referred to a strategic web compromise or watering hole attack. There are several known examples of this occurring.(Citation: Shadowserver Strategic Web Compromise) Typical drive-by compromise process: 1. A user visits a website that is used to host the adversary controlled content. 2. Scripts automatically execute, typically searching versions of the browser and plugins for a potentially vulnerable

version. * The user may be required to assist in this process by enabling scripting or active website components and ignoring warning dialog boxes. 3. Upon finding a vulnerable version, exploit code is delivered to the browser. 4. If exploitation is successful, then it will give the adversary code execution on the user's system unless other protections are in place. * In some cases a second visit to the website after the initial scan is required before exploit code is delivered. Unlike [Exploit Public-Facing Application](https://attack.mitre.org/techniques/T1190), the focus of this technique is to exploit software on a client endpoint upon visiting a website. This will commonly give an adversary access to systems on the internal network instead of external systems that may be in a DMZ. Adversaries may also use compromised websites to deliver a user to a malicious application designed to [Steal Application Access Token](https://attack.mitre.org/techniques/T1528)s, like OAuth tokens, to gain access to protected applications and information. These malicious applications have been delivered through popups on legitimate websites.(Citation: Volexity OceanLotus Nov 2017)

Name

T1055.013

ID

T1055.013

Description

Adversaries may inject malicious code into process via process doppelgänging in order to evade process-based defenses as well as possibly elevate privileges. Process doppelgänging is a method of executing arbitrary code in the address space of a separate live process. Windows Transactional NTFS (TxF) was introduced in Vista as a method to perform safe file operations. (Citation: Microsoft TxF) To ensure data integrity, TxF enables only one transacted handle to write to a file at a given time. Until the write handle transaction is terminated, all other handles are isolated from the writer and may only read the committed version of the file that existed at the time the handle was opened. (Citation: Microsoft Basic TxF Concepts) To avoid corruption, TxF performs an automatic rollback if the system or application fails during a write transaction. (Citation: Microsoft Where to use TxF) Although deprecated, the TxF application programming interface (API) is still enabled as of Windows 10. (Citation: BlackHat Process Doppelgänging Dec 2017) Adversaries may abuse TxF to a perform a file-less variation of [Process Injection](https://attack.mitre.org/techniques/T1055). Similar to [Process Hollowing](https://attack.mitre.org/techniques/T1055/012), process doppelgänging involves replacing the memory of a legitimate process, enabling the veiled execution of malicious code that may evade

defenses and detection. Process doppelgänging's use of TxF also avoids the use of highly-monitored API functions such as `NtUnmapViewOfSection`, `VirtualProtectEx`, and `SetThreadContext`. (Citation: BlackHat Process Doppelgänging Dec 2017) Process Doppelgänging is implemented in 4 steps (Citation: BlackHat Process Doppelgänging Dec 2017): * Transact – Create a TxF transaction using a legitimate executable then overwrite the file with malicious code. These changes will be isolated and only visible within the context of the transaction. * Load – Create a shared section of memory and load the malicious executable. * Rollback – Undo changes to original executable, effectively removing malicious code from the file system. * Animate – Create a process from the tainted section of memory and initiate execution. This behavior will likely not result in elevated privileges since the injected process was spawned from (and thus inherits the security context) of the injecting process. However, execution via process doppelgänging may evade detection from security products since the execution is masked under a legitimate process.

Name

T1497.003

ID

T1497.003

Description

Adversaries may employ various time-based methods to detect and avoid virtualization and analysis environments. This may include enumerating time-based properties, such as uptime or the system clock, as well as the use of timers or other triggers to avoid a virtual machine environment (VME) or sandbox, specifically those that are automated or only operate for a limited amount of time. Adversaries may employ various time-based evasions, such as delaying malware functionality upon initial execution using programmatic sleep commands or native system scheduling functionality (ex: [Scheduled Task/Job](https://attack.mitre.org/techniques/T1053)). Delays may also be based on waiting for specific victim conditions to be met (ex: system time, events, etc.) or employ scheduled [Multi-Stage Channels](https://attack.mitre.org/techniques/T1104) to avoid analysis and scrutiny.(Citation: Deloitte Environment Awareness) Benign commands or other operations may also be used to delay malware execution. Loops or otherwise needless repetitions of commands, such as [Ping](https://attack.mitre.org/software/S0097)s, may be used to delay malware execution and potentially exceed time thresholds of automated analysis environments.(Citation: Revil Independence Day)(Citation: Netskope Nitol) Another variation, commonly referred to as API hammering, involves making various

calls to [Native API](https://attack.mitre.org/techniques/T1106) functions in order to delay execution (while also potentially overloading analysis environments with junk data). (Citation: Joe Sec Nymaim)(Citation: Joe Sec Trickbot) Adversaries may also use time as a metric to detect sandboxes and analysis environments, particularly those that attempt to manipulate time mechanisms to simulate longer elapses of time. For example, an adversary may be able to identify a sandbox accelerating time by sampling and calculating the expected value for an environment's timestamp before and after execution of a sleep function.(Citation: ISACA Malware Tricks)

Name

T1055

ID

T1055

Description

Adversaries may inject code into processes in order to evade process-based defenses as well as possibly elevate privileges. Process injection is a method of executing arbitrary code in the address space of a separate live process. Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via process injection may also evade detection from security products since the execution is masked under a legitimate process. There are many different ways to inject code into a process, many of which abuse legitimate functionalities. These implementations exist for every major OS but are typically platform specific. More sophisticated samples may perform multiple process injections to segment modules and further evade detection, utilizing named pipes or other inter-process communication (IPC) mechanisms as a communication channel.

Name

T1036

ID

T1036

Description

Adversaries may attempt to manipulate features of their artifacts to make them appear legitimate or benign to users and/or security tools. Masquerading occurs when the name or location of an object, legitimate or malicious, is manipulated or abused for the sake of evading defenses and observation. This may include manipulating file metadata, tricking users into misidentifying the file type, and giving legitimate task or service names. Renaming abusable system utilities to evade security monitoring is also a form of [Masquerading](https://attack.mitre.org/techniques/T1036).(Citation: LOLBAS Main Site) Masquerading may also include the use of [Proxy](https://attack.mitre.org/techniques/T1090) or VPNs to disguise IP addresses, which can allow adversaries to blend in with normal network traffic and bypass conditional access policies or anti-abuse protections.

Name

T1140

ID

T1140

Description

Adversaries may use [Obfuscated Files or Information](https://attack.mitre.org/techniques/T1027) to hide artifacts of an intrusion from analysis. They may require separate mechanisms to decode or deobfuscate that information depending on how they intend to use it. Methods for doing that include built-in functionality of malware or by using utilities present on the system. One such example is the use of [certutil](https://attack.mitre.org/software/S0160) to decode a remote access tool portable executable file that has been hidden inside a certificate file.(Citation: Malwarebytes Targeted Attack against Saudi Arabia) Another example is using the Windows `copy /b` command to reassemble binary fragments into a malicious payload.(Citation: Carbon Black Obfuscation Sept 2016) Sometimes a user's action may be required to open it for deobfuscation or decryption as part of [User Execution](https://attack.mitre.org/techniques/T1204). The user may also be required to input a password to open a password protected compressed/encrypted file that was provided by the adversary. (Citation: Volexity PowerDuke November 2016)

Name

T1106

ID

T1106

Description

Adversaries may interact with the native OS application programming interface (API) to execute behaviors. Native APIs provide a controlled means of calling low-level OS services within the kernel, such as those involving hardware/devices, memory, and processes. (Citation: NT API Windows)(Citation: Linux Kernel API) These native APIs are leveraged by the OS during system boot (when other system components are not yet initialized) as well as carrying out tasks and requests during routine operations. Adversaries may abuse these OS API functions as a means of executing behaviors. Similar to [Command and Scripting Interpreter](https://attack.mitre.org/techniques/T1059), the native API and its hierarchy of interfaces provide mechanisms to interact with and utilize various components of a victimized system. Native API functions (such as `NtCreateProcess`) may be directed invoked via system calls / syscalls, but these features are also often exposed to user-mode applications via interfaces and libraries.(Citation: OutFlank System Calls)(Citation: CyberBit System Calls)(Citation: MDSec System Calls) For example, functions such as the Windows API `CreateProcess()` or GNU `fork()` will allow programs and scripts to start other processes.(Citation: Microsoft CreateProcess)(Citation: GNU Fork) This may allow API callers to execute a binary, run a CLI command, load modules, etc. as thousands of similar API functions exist for various system operations.(Citation: Microsoft Win32)(Citation: LIBC)(Citation: GLIBC) Higher level software frameworks, such as Microsoft .NET and macOS Cocoa, are also available to interact with native APIs. These frameworks typically provide language wrappers/abstractions to API functionalities and are designed for ease-of-use/portability of code.(Citation: Microsoft NET)(Citation: Apple Core Services)(Citation: MACOS Cocoa)(Citation: macOS Foundation) Adversaries may use assembly to directly or in-directly invoke syscalls in an attempt to subvert defensive sensors and detection signatures such as user mode API-hooks.(Citation: Redops Syscalls) Adversaries may also attempt to tamper with sensors and defensive tools associated with API monitoring, such as unhooking monitored functions via [Disable or Modify Tools](https://attack.mitre.org/techniques/T1562/001).

Name

T1574.001

ID

T1574.001

Description

Adversaries may execute their own malicious payloads by hijacking the search order used to load DLLs. Windows systems use a common method to look for required DLLs to load into a program. (Citation: Microsoft Dynamic Link Library Search Order)(Citation: FireEye Hijacking July 2010) Hijacking DLL loads may be for the purpose of establishing persistence as well as elevating privileges and/or evading restrictions on file execution. There are many ways an adversary can hijack DLL loads. Adversaries may plant trojan dynamic-link library files (DLLs) in a directory that will be searched before the location of a legitimate library that will be requested by a program, causing Windows to load their malicious library when it is called for by the victim program. Adversaries may also perform DLL preloading, also called binary planting attacks, (Citation: OWASP Binary Planting) by placing a malicious DLL with the same name as an ambiguously specified DLL in a location that Windows searches before the legitimate DLL. Often this location is the current working directory of the program.(Citation: FireEye fxsst June 2011) Remote DLL preloading attacks occur when a program sets its current directory to a remote location such as a Web share before loading a DLL. (Citation: Microsoft Security Advisory 2269637) Adversaries may also directly modify the search order via DLL redirection, which after being enabled (in the Registry and creation of a redirection file) may cause a program to load a different DLL.(Citation: Microsoft Dynamic-Link Library Redirection)(Citation: Microsoft Manifests) (Citation: FireEye DLL Search Order Hijacking) If a search order-vulnerable program is configured to run at a higher privilege level, then the adversary-controlled DLL that is loaded will also be executed at the higher level. In this case, the technique could be used for privilege escalation from user to administrator or SYSTEM or from administrator to SYSTEM, depending on the program. Programs that fall victim to path hijacking may appear to behave normally because malicious DLLs may be configured to also load the legitimate DLLs they were meant to replace.

Name

T1204.002

ID

T1204.002

Description

An adversary may rely upon a user opening a malicious file in order to gain execution. Users may be subjected to social engineering to get them to open a file that will lead to code execution. This user action will typically be observed as follow-on behavior from [Spearphishing Attachment](https://attack.mitre.org/techniques/T1566/001). Adversaries may use several types of files that require a user to execute them, including .doc, .pdf, .xls, .rtf, .scr, .exe, .lnk, .pif, and .cpl. Adversaries may employ various forms of [Masquerading](https://attack.mitre.org/techniques/T1036) and [Obfuscated Files or Information](https://attack.mitre.org/techniques/T1027) to increase the likelihood that a user will open and successfully execute a malicious file. These methods may include using a familiar naming convention and/or password protecting the file and supplying instructions to a user on how to open it.(Citation: Password Protected Word Docs) While [Malicious File](https://attack.mitre.org/techniques/T1204/002) frequently occurs shortly after Initial Access it may occur at other phases of an intrusion, such as when an adversary places a file in a shared directory or on a user's desktop hoping that a user will click on it. This activity may also be seen shortly after [Internal Spearphishing](https://attack.mitre.org/techniques/T1534).

Domain-Name

Value
updatenazure.com

Url

Value
https://cdn-network-services-001.com/update/minor/1/release.json

IPv4-Addr

Value
152.89.217.215

StixFile

Value
e05e561c5118efdbca113ca231c527b62e59a4bffae3bd374f7b4fcdd10e7d90
c5c52331b208cad19dc710786e26ac55090ffca937410d76c53569d731f0bb92
a3a5e7011335a2284e2d4f73fd464ff129f0c9276878a054c1932bc50608584b
befe0df365f0e2dc05225470e45fdf03609f098a526d617c478b81ac6bb9147f
02d5e281689ec2d4ab8ac19c93321a09113e5d8fa39380a7021580ea1887b7a5

External References

- https://www.rapid7.com/blog/post/2024/03/28/stories-from-the-soc-part-1-idat-loader-to-bruteratel/
- https://otx.alienvault.com/pulse/660a7ce1e59c73f5e5e2ef0a