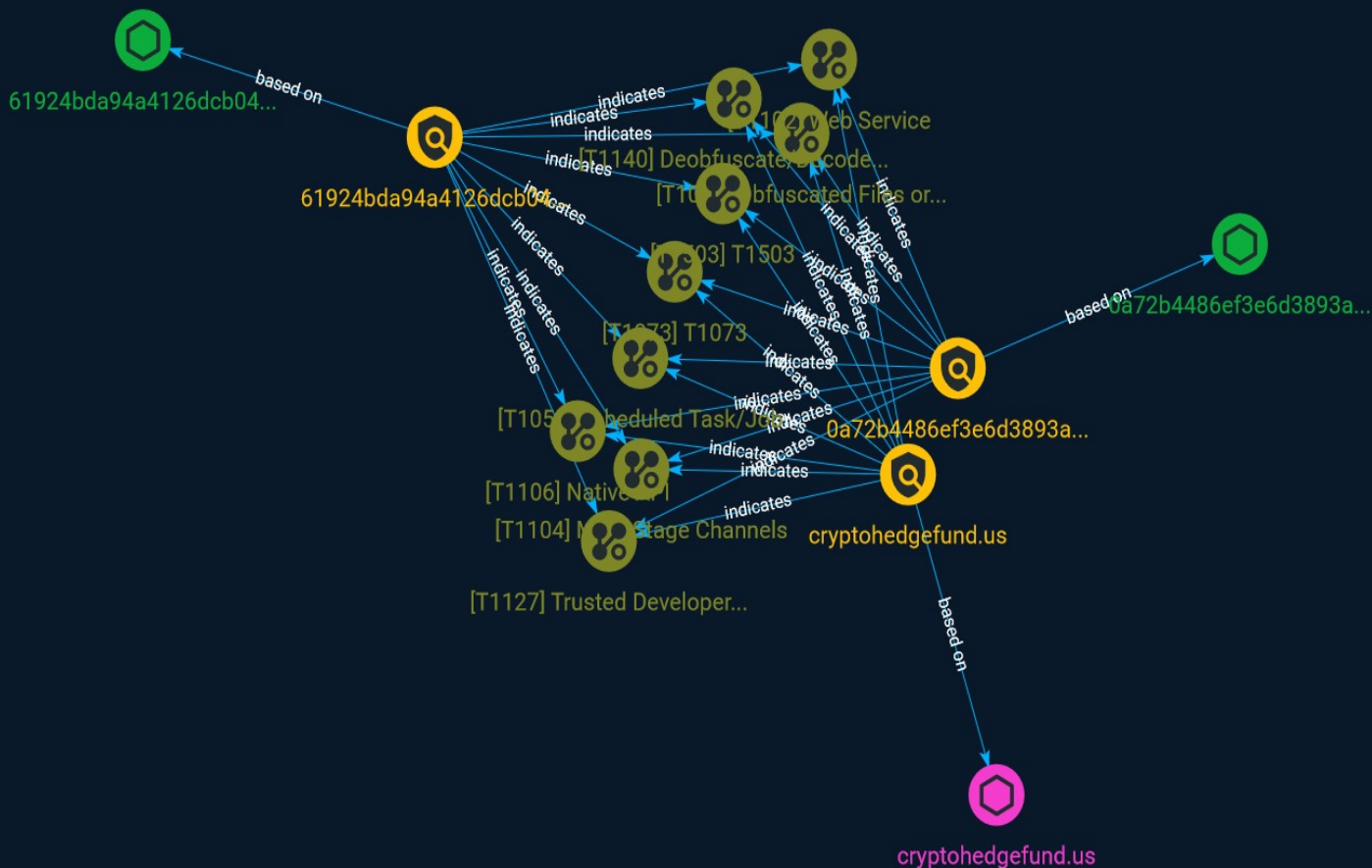




NETMANAGEIT

# Intelligence Report

## Sneaky DoubleFinger loads GreetingGhoul targeting your cryptocurrency



# Table of contents

---

## Overview

---

● Description	3
● Confidence	3

---

---

## Entities

---

● Attack-Pattern	4
● Indicator	10

---

---

## Observables

---

● Domain-Name	12
● StixFile	13

---

---

## External References

---

● External References	14
-----------------------	----

---

# Overview

## Description

Stealing cryptocurrencies is nothing new. For example, the Mt. Gox exchange was robbed of many bitcoins back in the beginning of 2010s. Attackers such as those behind the Coinvault ransomware were after your Bitcoin wallets, too. Since then, stealing cryptocurrencies has continued to occupy cybercriminals.

## Confidence

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

15 / 100

# Attack-Pattern

**Name**

Scheduled Task/Job

**ID**

T1053

**Description**

Adversaries may abuse task scheduling functionality to facilitate initial or recurring execution of malicious code. Utilities exist within all major operating systems to schedule programs or scripts to be executed at a specified date and time. A task can also be scheduled on a remote system, provided the proper authentication is met (ex: RPC and file and printer sharing in Windows environments). Scheduling a task on a remote system typically may require being a member of an admin or otherwise privileged group on the remote system.(Citation: TechNet Task Scheduler Security) Adversaries may use task scheduling to execute programs at system startup or on a scheduled basis for persistence. These mechanisms can also be abused to run a process under the context of a specified account (such as one with elevated permissions/privileges). Similar to [System Binary Proxy Execution](<https://attack.mitre.org/techniques/T1218>), adversaries have also abused task scheduling to potentially mask one-time execution under a trusted system process. (Citation: ProofPoint Serpent)

**Name**

Native API

**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. 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 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. While invoking API functions, adversaries may also attempt to bypass defensive tools (ex: unhooking monitored functions via [Disable or Modify Tools](<https://attack.mitre.org/techniques/T1562/001>)).

**Name**

Obfuscated Files or Information

**ID**

T1027

**Description**

Adversaries may attempt to make an executable or file difficult to discover or analyze by encrypting, encoding, or otherwise obfuscating its contents on the system or in transit. This is common behavior that can be used across different platforms and the network to evade defenses. Payloads may be compressed, archived, or encrypted in order to avoid detection. These payloads may be used during Initial Access or later to mitigate detection. Sometimes a user's action may be required to open and [Deobfuscate/Decode Files or Information](<https://attack.mitre.org/techniques/T1140>) for [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) Adversaries may also use compressed or archived scripts, such as JavaScript. Portions of files can also be encoded to hide the plain-text strings that would otherwise help defenders with discovery. (Citation: Linux/Cdorked.A We Live Security Analysis) Payloads may also be split into separate, seemingly benign files that only reveal malicious functionality when reassembled. (Citation: Carbon Black Obfuscation Sept 2016) Adversaries may also abuse [Command Obfuscation](<https://attack.mitre.org/techniques/T1027/010>) to obscure commands executed from payloads or directly via [Command and Scripting Interpreter](<https://attack.mitre.org/techniques/T1059>). Environment variables, aliases, characters, and other platform/language specific semantics can be used to evade signature based detections and application control mechanisms. (Citation: FireEye Obfuscation June 2017) (Citation: FireEye Revoke-Obfuscation July 2017)(Citation: PaloAlto EncodedCommand March 2017)

**Name**

Multi-Stage Channels

**ID**

T1104

**Description**

Adversaries may create multiple stages for command and control that are employed under different conditions or for certain functions. Use of multiple stages may obfuscate the command and control channel to make detection more difficult. Remote access tools will call back to the first-stage command and control server for instructions. The first stage may have automated capabilities to collect basic host information, update tools, and upload additional files. A second remote access tool (RAT) could be uploaded at that point to redirect the host to the second-stage command and control server. The second stage will likely be more fully featured and allow the adversary to interact with the system

through a reverse shell and additional RAT features. The different stages will likely be hosted separately with no overlapping infrastructure. The loader may also have backup first-stage callbacks or [Fallback Channels](<https://attack.mitre.org/techniques/T1008>) in case the original first-stage communication path is discovered and blocked.

**Name**

T1073

**ID**

T1073

**Name**

Web Service

**ID**

T1102

**Description**

Adversaries may use an existing, legitimate external Web service as a means for relaying data to/from a compromised system. Popular websites and social media acting as a mechanism for C2 may give a significant amount of cover due to the likelihood that hosts within a network are already communicating with them prior to a compromise. Using common services, such as those offered by Google or Twitter, makes it easier for adversaries to hide in expected noise. Web service providers commonly use SSL/TLS encryption, giving adversaries an added level of protection. Use of Web services may also protect back-end C2 infrastructure from discovery through malware binary analysis while also enabling operational resiliency (since this infrastructure may be dynamically changed).

**Name**

Trusted Developer Utilities Proxy Execution

**ID**

T1127

**Description**

Adversaries may take advantage of trusted developer utilities to proxy execution of malicious payloads. There are many utilities used for software development related tasks that can be used to execute code in various forms to assist in development, debugging, and reverse engineering.(Citation: engima0x3 DNX Bypass)(Citation: engima0x3 RCSI Bypass)(Citation: Exploit Monday WinDbg)(Citation: LOLBAS Tracker) These utilities may often be signed with legitimate certificates that allow them to execute on a system and proxy execution of malicious code through a trusted process that effectively bypasses application control solutions.

**Name**

Deobfuscate/Decode Files or Information

**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**

T1503

**ID**

T1503

# Indicator

**Name**

cryptohedgefund.us

**Pattern Type**

stix

**Pattern**

[domain-name:value = 'cryptohedgefund.us']

**Name**

61924bda94a4126dcb04e4ab481c3193a3de8eb589dcb20e739c8fde2bf7b006

**Description**

SHA256 of 16203abd150a709c0629a366393994ea

**Pattern Type**

stix

**Pattern**

[file:hashes!'SHA-256' =  
'61924bda94a4126dcb04e4ab481c3193a3de8eb589dcb20e739c8fde2bf7b006']

**Name**

0a72b4486ef3e6d3893ad3169cfbf6556550899e1dbc92afae264fa45cfb7ce8

**Description**

SHA256 of a9a5f529bf530d0425e6f04cbe508f1e

**Pattern Type**

stix

**Pattern**

[file:hashes!'SHA-256' =  
'0a72b4486ef3e6d3893ad3169cfbf6556550899e1dbc92afae264fa45cfb7ce8']

# Domain-Name

## Value

cryptohedgefund.us

# StixFile

## Value

61924bda94a4126dcb04e4ab481c3193a3de8eb589dcb20e739c8fde2bf7b006

0a72b4486ef3e6d3893ad3169cfbf6556550899e1dbc92afae264fa45cfb7ce8

# External References

- 
- <https://securelist.com/doublefinger-loader-delivering-greetingghoul-cryptocurrency-stealer/109982/>
- 
- <https://otx.alienvault.com/pulse/648779ab5ab20b509301154f>