Computop AI Protect

Über Computop AI Protect (unterstützt von Nethone)

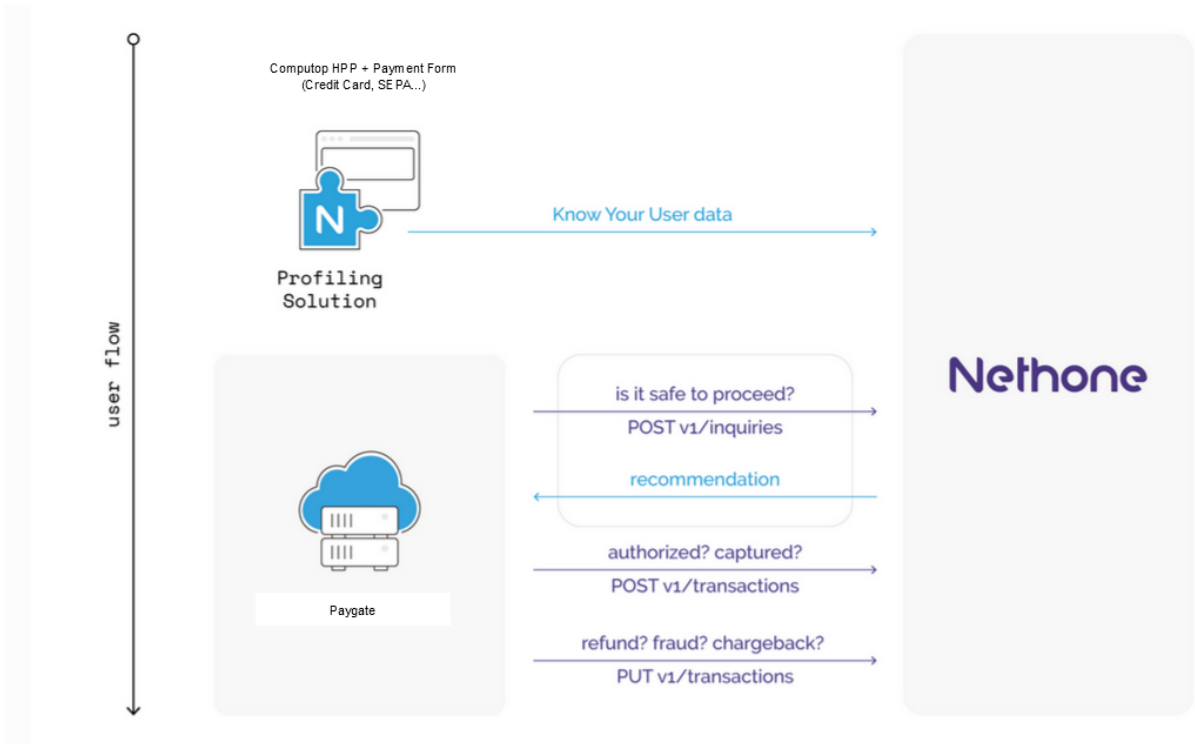
Computop AI Protect ist eine KI-basierte Betrugserkennungslösung mit Profiling und maschinellen Lernmodellen, die dabei hilft, Rückbuchungsraten und das Betrugsrisiko zu senken. Das System arbeitet daran, den Verkehr auf der Website zu profilieren, betrügerisches Verhalten zu erkennen und Einblicke in die Geschäftsintelligenz zu liefern. Es verarbeitet die bereitgestellten Daten und muss im richtigen Moment abgefragt werden, um das Betrugsrisiko einzuschätzen oder im Moment der Abfrage Einblicke zu liefern.

Logo	Computop the payment people | Nethone
Info	Mit der Betrugspräventionslösung Computop AI Protect können Sie jeden einzelnen Benutzer überprüfen, um alle riskanten Benutzer zu stoppen, ohne die guten Benutzer zu beeinträchtigen. Maximieren Sie passiv und in Echtzeit die Akzeptanzrate und reduzieren Sie Ihre Betrugs-/Rückbuchungsquote mit den genauesten Lösungen zur Erkennung von Finanztransaktionsbetrug.
Type	Risikomanagement

KI-Schutzlösung

Die Lösung funktioniert in zwei Schritten

- Profiling-Lösung – zum Sammeln von Benutzerdaten
- Anfrage – um Empfehlungen zur Transaktion abzugeben



Diese Betrugsprüfung/Transaktionsüberprüfung kann für verschiedene Zahlungsmethoden verwendet werden. Wir haben uns auf Kreditkarten-, SEPA- und Paypal-Transaktionen konzentriert, aber sie kann erweitert werden.

Das empfohlene Ergebnis der Betrugsprüfung von AI Protect lautet:

- Akzeptieren – Transaktion gilt als nicht betrügerisch
- Ablehnen – Transaktion gilt als betrügerisch
- Überprüfen – Transaktion abgelehnt. Der Händler hat die Möglichkeit, die Anfrage im Panel zu überprüfen und bei Bedarf die erforderlichen Änderungen an den Regeln vorzunehmen.

Implementierung

Computop hat AI Protect auf unseren gehosteten Zahlungsseiten paymentpage.aspx, payssl.aspx und paysdd.aspx implementiert.

AI Protect (Profiling + Anfrage) HPP

Das Profiler-Modul ist als Option zur Verwendung einer Vorautorisierungsfunktion über unsere gehostete Zahlungsseite (payssl.aspx) oder Zahlungsformulare paymentpage.aspx und paysdd.aspx implementiert.

Der praktische Weg besteht darin, dass wir vor der Autorisierung die vom Händler in der Anfrage gesendeten Daten verwenden und eine Betrugsprüfung durchführen können, bevor die Autorisierung erfolgt.

Die verwendbare Logik ist:

- wenn der Betrugsprüfungsstatus akzeptiert ist = Paygate löst die Autorisierung aus
- wenn der Betrugsprüfungsstatus abgelehnt ist, findet die Autorisierung nie statt, der Kunde kann eine andere Zahlungsmethode verwenden

AI Protect (Profiling + Anfrage) Server-zu-Server und Paypal

Der Profiler wurde implementiert, um Betrugsprüfungen per Server-zu-Server für die Zahlungsmethoden Kreditkarte, PayPal und EDD durchführen zu können.

Der Händler muss allerdings eine Profiling-Lösung auf seiner Webseite installieren.

Profiling-Implementierung

Folgende drei Schritte sind zum Profiling nötig

1. Javascript laden
2. Profiling starten
3. Profiling abschließen

JavaScript-Datei

Diese JavaScript-Datei enthält eine Profilierungslogik. Die statische JavaScript-Datei sollte auf der Webseite mit einer URL in der Form https://`domain_name`/s/`merchant_id`/`script_name`.js eingebunden werden.

Hinweis*: Das Profiling-Skript wird während des Onboardings von Nethone bereitgestellt..

Argument name	Details
`domain_name`	**string (required)** — domain used by Nethone profiler
`merchant_id`	**string (required)** — merchant identifier provided during the integration process (contains only numbers), maxlen *19 chars*
`script_name`	**string (required)** — script name provided by Nethone during integration; maxlen *60 chars* has to start with 'd' character

Please use the `'crossorigin="use-credentials"'` property when attaching the JavaScript and add `'async'` property.

Skripteinbindung

Der folgende Codeausschnitt muss auf der Seite eingefügt werden, auf der der Händler das Profiling durchführen möchte. Normalerweise ist dies die Zahlungsseite.

```
<!-- javascript inserted at the bottom of body; merchant_id may be hardcoded or inserted by backend,
crossorigin="use-credentials" property is mandatory-->
<script type="text/javascript" id="SCRIPT_TAG_ID" crossorigin="use-credentials" src="https://provided_name.
nethone.io/s/${merchant_id}/dNfsXe.js" async></script>
```

Skriptinitialisierungsfunktion

Die Skriptinitialisierungsfunktion (`dftp.init`) muss ausgeführt werden, um die Browserdaten des Benutzers zu erfassen. Sie sollte aufgerufen werden, wenn das Formular geladen wurde. Die Funktion erfordert ein Argument `options`, ein JavaScript-Objekt mit mehreren Eigenschaften.

Mögliche Eigenschaften sind:

`attemptReference`	**String (erforderlich)** – eindeutige ID, die beim Anzeigen des Formulars generiert wird. Sie darf **kein** Präfix `mznx-` enthalten maxlen *128 Zeichen*
`sensitiveFields`	**Liste (bedingt erforderlich)** – eine Liste von Zeichenfolgen mit IDs sensibler Felder. Aus den für diese Felder erfassten Verhaltensdaten werden alle vertraulichen Informationen entfernt. Normalerweise sind dies die *Kreditkartennummer* ('ccn') und das Feld *dreistellige CVV* ('cvv').
`allowedFields`	**Liste (bedingt erforderlich)** – eine Liste von Zeichenfolgen mit Whitelist-Feld-IDs. Für diese Felder werden vollständige Verhaltensdaten erfasst, ohne dass Informationen entfernt werden. Normalerweise sind dies z.B. die Felder *Name* oder *Adresse*.
`secretFields`	**List (optional)** – eine Liste von Zeichenfolgen, die IDs geheimer Felder enthalten. Für diese Felder werden keine Verhaltensdaten erfasst. Normalerweise wäre dies das Feld *Passwort*.

Parameter attemptReference

`attemptReference` wird verwendet, um die durchgeführte Abfrage mit den Daten abzugleichen, die das Profiling-Skript über einen Benutzer erfasst hat. Es muss sich um einen eindeutigen Wert handeln, der bei jeder Formularansicht generiert wird. Sie sollten `attemptReference` niemals wiederverwenden. Bei Single Page Applications dürfen Sie die Profiling-Lösung nicht neu initialisieren.

Parameter für Verhaltensdaten

Die Handhabung der vom Profiling-Skript erfassten Verhaltensdaten wird durch die Parameter `sensitiveFields`, `allowedFields` und `secretFields` gesteuert. Die Übergabe von `sensitiveFields` und `allowedFields` steuert, wie Verhaltensdaten aus Feldern behandelt werden, die nicht anderweitig angegeben sind. Die möglichen Handhabungsmodi für nicht angegebene Felder sind Standard und Alternative.

Modus Standard:

Dies ist der empfohlene Modus. In diesem Modus werden für Felder, die nicht anderweitig angegeben sind, vollständige Verhaltensdaten erfasst. Die Übergabe von nur `sensititiveFields` aktiviert diese Modus, `secretFields` können zusätzlich übergeben werden.

Beispielnutzung:

```
dftp.init({
    attemptReference: '8b7115e0-49d2-438b-b88f-b265e44b156f',
    sensitiveFields: ['number', 'cvc']
});
// or
dftp.init({
    attemptReference: '8b7115e0-49d2-438b-b88f-b265e44b156f',
    sensitiveFields: ['number', 'cvc'],
    secretFields: ['password']
});
```

Zusätzliche Überlegungen

Vom Profiling-Skript erfasste Verhaltensdaten werden nach Formularfeld-IDs gruppiert. Es wird dringend empfohlen, dass alle Formularfelder HTML-IDs haben, die im Laufe der Zeit stabil bleiben.

⚠️ Warnung **sensible/erlaubte/geheime Felder MÜSSEN HTML IDs HABEN.**

Wenn eines der folgende Dinge zutrifft:

- Felder haben keine IDs im HTML-Code
- HTML-Klasseneigenschaft wird anstelle von IDs übergeben
- HTML-Namenseigenschaft wird anstelle von IDs übergeben

können ALLE sensiblen/geheimen Schlüsselereignisse (zum Beispiel mit enthaltenen Kartendaten) an den Profiler gesendet werden.

Profiling-Abschluss

Um sicherzustellen, dass alle erforderlichen Profiling-Daten erfasst wurden, wird empfohlen, die Funktion `dftp.profileCompleted` zu verwenden. Sie gibt ein *Promise*-Objekt zurück, das aufgelöst wird, wenn wir die Verarbeitung abgeschlossen haben und eine Abfrage sicher durchgeführt werden kann.

```
Sample usage with modern JavaScript:

try {
    await dftp.profileCompleted();
} catch (err) {
    console.error("Profiling failed with err: " + err);
}
doWorkAfterProfiling();


Sample usage with Promises:

dftp.profileCompleted().catch(err => console.error("Profiling failed with err: " + err)).finally
(doWorkAfterProfiling);


doWorkAfterProfiling can be a function used to submit form data and perform the inquiry
```

Profiling-Skript & Seitenlebenszyklus

Das Profiling-Skript **muss** nur einmal pro Seitenlebenszyklus geladen und initialisiert werden. Sie können das Vorhandensein des Objekts `dftp` prüfen , um ein erneutes Laden des Skripts zu vermeiden. Wenn der Benutzer wiederholt Transaktionen durchführt, ohne die Seite neu zu laden, sollten Sie bei einer Anfrage weiterhin dieselbe `attemptReference` verwenden.

Beispiel einer Zahlungsseitenvorlage und Implementierung eines Profiling-Skripts

```
<html>
<head>
</head>

<body>
    <!-- payment form -->
    <form id="payment-form">
        <label>
            Name:<br>
            <input type="text" name="name" id="name" class="form-element">
        </label>
        <br>
        <label>
            Credit card number:<br>
            <input type="text" name="ccn" id="ccn" class="form-element">
        </label>
        <br>
        <label>
            Expiration date:<br>
            <input type="text" name="expiration" id="expiration" class="form-element">
        </label>
        <br>
        <label>
            CVV:<br>
            <input type="text" name="cvv" id="cvv" class="form-element">
        </label>
    </form>
    <!-- payment form end-->

    <!-- button which is used for sending data, calls dftp.profileCompleted inside sendForm wrapper-->
    <button id="send" onclick="sendForm()">Pay</button>

    <script>
        function validateFormAndSend(){
            // isValid is the merchants function used for card data validation
            if (isValid()) {
                // merchant function used for sending data to server
                sentPaymentData();
            } else {
                // merchant function used for displaying card data errors on the form
                displayValidationErrors();
            }
        }

        // function for handling case when script from profiler URL cannot be loaded and dftp object does
not exist
        function sendForm() {
            if (window.dftp){
                dftp.profileCompleted().catch(err => console.error("Profiling failed with err: " + err)).
finally(validateFormAndSend);
            }
            else {
                validateFormAndSend();
            }
        }
    </script>

    <!-- javascript inserted at the bottom of body; merchant_id may be hardcoded or inserted by backend,
crossorigin="use-credentials" property is mandatory-->
    <script type="text/javascript" id="SCRIPT_TAG_ID" crossorigin="use-credentials" src="
https://provided_name.nethone.io/s/${merchant_id}/dNfsXe.js" async></script>
```

```
    <!-- javascript initializing profiling -->
    <script>
      var scriptID = "SCRIPT_TAG_ID";       // ID of <script> tag where our script is being loaded
      var options = {
          attemptReference: "${attempt_reference}",   // inserted by the backend
          sensitiveFields: ["ccn", "cvv"];            // list of sensitive fields
      };

      if (window.dftp) {
          dftp.init(options);
      } else {
          var el = document.getElementById(scriptID);
          el.addEventListener("load", function () {
              dftp.init(options);
          });
      }
    </script>

</body>
</html>
```

Unterstützung

Das hängt vollständig von der verwendeten Zahlungsart ab. Derzeit ist dies für folgende Zahlungsarten verfügbar:

- Kreditkarte
- SEPA-Lastschrift
- PayPal

Sequenzdiagramm

Vorab-Auth mittels HPP

Paygate-Schnittstelle

Definitionen

Datenformate

Format	Beschreibung
a	alphabetisch
as	alphabetisch mit Sonderzeichen
n	numerisch
an	alphanumerisch
ans	alphanumerisch mit Sonderzeichen
ns	numerisch mit Sonderzeichen
bool	Bool'scher Ausdruck (true oder false)
3	feste Länge mit 3 Stellen/Zeichen

..3	variable Länge mit maximal 3 Stellen/Zeichen
enum	Aufzählung erlaubter Werte
dttm	ISODateTime (JJJJ-MM-TTThh:mm:ss)

Abkürzungen

Abkürzung	Beschreibung	Kommentar
CND	Bedingung (condition)	
M	Pflicht (mandatory)	Wenn ein Parameter Pflicht ist, dann muss er vorhanden sein
O	optional	Wenn ein Parameter optional ist, dann kann er vorhanden sein, ist aber nicht erforderlich
C	bedingt (conditional)	Wenn ein Parameter bedingt ist, dann gibt es eine Bedingungsregel, die angibt, ob er Pflicht oder optional ist

Hinweis: Bitte beachten Sie, dass die Bezeichnungen der Parameter in Groß- oder Kleinbuchstaben zurückgegeben werden können.

Aufruf der Schnittstelle

Autorisierungsanfrage:

Um eine AI Protect-Prüfung über eine Server-zu-Server-Verbindung durchzuführen, beachten Sie bitte die jeweiligen Zahlungsmethoden Zahlungen per Kreditkarte, Zahlungen per Lastschrift, PayPal V2

Der Parameter ***attempt_reference*** muss zusätzlich zu den jeweiligen Parametern der Zahlungsart angegeben werden.

Key	Format	CND	Beschreibung
attempt_reference	String maxlen *128 Zeichen*	R	Eindeutige ID, die beim Aufruf des Profiling-Skripts generiert wird