Integrations

- Discovery Webhooks
- Nimbusec Webhooks
- Plesk - Nimbusec Webhosting Security

Discovery Webhooks

Overview

Webhooks allow you to build or set up applications which subscribe to new reports in your Discovery account. When a new report was generated, we'll send a HTTP POST payload to the webhook's configured URL. Webhooks can be used to update an external issue tracker, feed into a SIEM, or even deploy a backup to your production server. You're only limited by your imagination.

Webhooks can be installed and configured exactly like your email or Slack notifiations. Once installed, the webhook will be triggered each time a new report is generated.

Setting up a webhook

Creating a webhook is a two-step process. You'll first need to set up how you want your webhook to behave through Discovery. After that, you'll set up your server to receive and manage the payload.

Setting up a webhook is similiar to setting up email or text notifications. Go to Settings > Notifications and click on "Add Custom Webhook".

Webhook URL: This is the server endpoint that will receive the webhook payload. This URL must be accessible by the Discovery services, so make sure it is not a private URL like in the example image, but publicly accessible.

We recommend that you protect the URL via HTTPS as well.

Payload

The second step to process webhooks is the server to handle the request. Discovery will send a `POST` request to the specified webhook URL for each new report it generates.

The body of the request has always the content type `application/json` and the following format:

```
{
   "id": "string",
   "time": "int",
   "origin": "string",
   "summary": {
      "discovered": "int",
      "responding": "int",
      "scanned": "int",
      "malware": "int",
      "defacement": "int",
      "reputation": "int",
      "application": "int",
      "tls": "int"
   }
}
```

Securing your webhooks

Once your server is configured to receive payloads, it'll listen for any payload sent to the endpoint you configured. For security reasons, you probably want to limit requests to those coming from Discovery.

Discovery uses a hash signature to sign each webhook payload. This hash signature is passed along with each request in the headers as `X-Nimbusec-Signature`. Discovery uses PKCS 1 using RSAwithSHA512 to calculate the signature, which is sent encoded as BASE64 in the headers value.

You can use the following example codes to validate the Discovery signature:

Using Java:

```
public boolean verifySignature(byte[] signature, byte[] data) {
   try {
      PKCS8EncodedKeySpec spec = new PKCS8EncodedKeySpec(Base64.decodeBase64(PRIVATE_KEY));
      PublicKey key = KeyFactory.getInstance("RSA").generatePublic(spec);

      Signature sig = Signature.getInstance("SHA512WithRSA");
      sig.initVerify(key);
```

```
        sig.update(data);

        return sig.verify(signature);
    } catch(InvalidKeySpecException | SignatureException | NoSuchAlgorithmException | InvalidKeyException ex)
{
        return false;
    }
}
```

Using Go:

```
package example

import (
    "crypto"
    "crypto/rsa"
    "crypto/sha512"
)

func VerifySignature(signature []byte, data []byte) bool {
    // Example code to get the Discovery signature
    // header := request.Header.Get("X-Nimbusec-Signature")
    // signature, _ := base64.StdEncoding.DecodeString(header)

    // Example code to get the response body data
    // data, _ := ioutil.ReadAll(r.Body)

    hashed := sha512.Sum512(data)
    err := rsa.VerifyPKCS1v15(PublicKey, crypto.SHA512, hashed[:], signature)
    return err == nil
}
```

The repository [nimbusec-oss/nimbusec-gelf] showcases how to verify Discovery webhook signatures as well.

The public key used to verify all Discovery webhook signatures is:

```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAlgbfxMniLiDMRYhRYY0f
fOLEACXSCWmX0/rSL+qib/3cbZAMOEknXUudich4ZuCHulZ9ApaPx/7u+x5jQSj4
aiZXJE+S+LecUqwbq1CSfByLPViyYu2xt2I0tqYdsQK6KmQs2Gl00UP/yxrHtcEz
```

NaZO8Z7bdL1AY3eW6oPjWeORK91FAEONbnCvXmPoGa/4+AUWr6FmMrjFiG8yM72K
eUvfzyWtZYNeFxJ+2UmqTco1oEdGmwJJYKgPAg4mRXOPBs1Il6W9+bwomUed/Rxd
GHuNPy4b9BOgSyFFoEHQJ2eL+W9IMpWegwV7VxXc37WlHQxoZ1886gO+u3hxvo++
+v0ami3JT1BZriTYdjSydktyUARQQzDaxAsYwUMTs/G++yiF3jt+J43pKvZ+ZSTP
+vXAKd+acbsUmH6WIxsu915BVPcnMgyeUWOK6NojiW4Z4BEuCWVKfqMKRU+LypFN
Hqpd3wxT26jnykJOm0a2xloXlmjS9x/LcHd6onN6I6wdPz8zSAU6lr0T2kWgPY+l
u0Ral9lpafe/Rq6GjPIvrlWNy2hjJhJ1FtzMCgySCs+XEqjFbM2GEOSK4M/NGY9+
zzkNgL4B0HpMHgRNeRfx0q+LuZtuHvNEDxmp/OvvfRqQGo5qqDhojm3rRi5qbsLa
k3siF46a7ml6ONtAD/Eib1kCAwEAAQ==
-----END PUBLIC KEY-----

Nimbusec Webhooks

Overview

Webhooks allow you to build or set up applications which subscribe to certain alerts in your Nimbusec account. When one of those alerts is triggered, we'll send a HTTP POST payload to the webhook's configured URL. Webhooks can be used to update an external issue tracker, feed into a SIEM, or even deploy a backup to your production server. You're only limited by your imagination.

Webhooks can be installed and configured exactly like your email and text notifiations. Once installed, the webhook will be triggered each time one or more subscribed alerts occur.

Setting up a webhook

Creating a webhook is a two-step process. You'll first need to set up how you want your webhook to behave through Nimbusec--what alerts it should listen to. After that, you'll set up your server to receive and manage the payload.

Setting up a webhook is similiar to setting up email or text notifications. Go to Settings > Notifications and click on "Add Webhook".

Image not found or type unknown

Webhooks require a few configuration options before you can make use of them. We'll go through each of these settings below.

Add Webhook Dialog

Image not found or type unknown

Webhook URL: This is the server endpoint that will receive the webhook payload. This URL must be accessible by the Nimbusec services, so make sure it is not a private URL like in the example image, but publicly accessible.

We recommend that you protect the URL via HTTPS as well.

Domains: Your webhook will only receive alerts for domains that you select. Select the domains you want to receive alert for in "Available Domains" and you the arrow buttons to move then to "Domains with notifications".

Notification levels: The webhook receives alerts for each new issue Nimbusec detecs (just like email and text notifications). Nimbusec issues are either rated as "*medium risk*" or "*severe risk*". You can select with the dropdowns in which alerts you are interested.

Payload

The second step to process webhooks is the server to handle the request. Nimbusec will send a `POST` request to the specified webhook URL for each new alert it detects.

The body of the request has always the content type `application/json` and the following format:

```
{
   "domain": {
      "id": "string",
      "name": "string",
      "url": "string",
      "responseIP": "string",
   },
   "issues": [
      {
         "id": "string",
         "event": "string",
         "category": "string",
         "severity": "int",
                  "regions": ["string"],
                  "viewports": ["string"],
         "details": { /* ... */ }
      },
      // ...
   ]
}
```

A sample application to receive webhooks can be found in our GitHub repository [nimbusec-oss/nimbusec-gelf](). This small example implementation receives any Nimbusec alerts and converts them to a GELF message that can be pushed to any GELF processing server like Graylog. It is used internally at nimbusec to feed our own alerts into our SIEM.

Securing your webhooks

Once your server is configured to receive payloads, it'll listen for any payload sent to the endpoint you configured. For security reasons, you probably want to limit requests to those coming from Nimbusec.

Nimbusec uses a hash signature to sign each webhook payload. This hash signature is passed along with each request in the headers as `X-Nimbusec-Signature`. Nimbusec uses PKCS 1 using RSAwithSHA512 to calculate the signature, which is sent encoded as BASE64 in the headers value.

You can use the following example codes to validate the Nimbusec signature:

Using Java:

```
public boolean verifySignature(byte[] signature, byte[] data) {
    try {
        PKCS8EncodedKeySpec spec = new PKCS8EncodedKeySpec(Base64.decodeBase64(PRIVATE_KEY));
        PublicKey key = KeyFactory.getInstance("RSA").generatePublic(spec);

        Signature sig = Signature.getInstance("SHA512WithRSA");
        sig.initVerify(key);
        sig.update(data);

        return sig.verify(signature);
    } catch(InvalidKeySpecException | SignatureException | NoSuchAlgorithmException | InvalidKeyException ex)
{
        return false;
    }
}
```

Using Go:

```
package example

import (
    "crypto"
    "crypto/rsa"
    "crypto/sha512"
)
```

```
func VerifySignature(signature []byte, data []byte) bool {
    // Example code to get the Nimbusec signature
    // header := request.Header.Get("X-Nimbusec-Signature")
    // signature, _ := base64.StdEncoding.DecodeString(header)

    // Example code to get the response body data
    // data, _ := ioutil.ReadAll(r.Body)

    hashed := sha512.Sum512(data)
    err := rsa.VerifyPKCS1v15(PublicKey, crypto.SHA512, hashed[:], signature)
    return err == nil
}
```

The repository [nimbusec-oss/nimbusec-gelf] showcases how to verify Nimbusec webhook signatures as well.

The public key used to verify all Nimbusec webhook signatures is:

```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAlgbfxMniLiDMRYhRYY0f
fOLEACXSCWmX0/rSL+qib/3cbZAMOEknXUudich4ZuCHulZ9ApaPx/7u+x5jQSj4
aiZXJE+S+LecUqwbq1CSfByLPViyYu2xt2I0tqYdsQK6KmQs2Gl00UP/yxrHtcEz
NaZO8Z7bdL1AY3eW6oPjWeORK91FAEONbnCvXmPoGa/4+AUWr6FmMrjFiG8yM72K
eUvfzyWtZYNeFxJ+2UmqTco1oEdGmwJJYKgPAg4mRXOPBs1Il6W9+bwomUed/Rxd
GHuNPy4b9BOgSyFFoEHQJ2eL+W9IMpWegwV7VxXc37WlHQxoZ1886gO+u3hxvo++
+v0ami3JT1BZriTYdjSydktyUARQQzDaxAsYwUMTs/G++yiF3jt+J43pKvZ+ZSTP
+vXAKd+acbsUmH6WIxsu915BVPcnMgyeUWOK6NojiW4Z4BEuCWVKfqMKRU+LypFN
Hqpd3wxT26jnykJOm0a2xloXlmjS9x/LcHd6onN6I6wdPz8zSAU6lr0T2kWgPY+l
u0Ral9lpafe/Rq6GjPIvrlWNy2hjJhJ1FtzMCgySCs+XEqjFbM2GEOSK4M/NGY9+
zzkNgL4B0HpMHgRNeRfx0q+LuZtuHvNEDxmp/OvvfRqQGo5qqDhojm3rRi5qbsLa
k3siF46a7ml6ONtAD/Eib1kCAwEAAQ==
-----END PUBLIC KEY-----
```

Plesk - Nimbusec Webhosting Security

This how-to guide describes the usage of the Nimbusec Webhosting Security Plugin for Plesk.

Installation & Uninstallation

There are two ways on how the manage the installation and uninstallation of the Plesk plugin.

Manage through Plesk extension store

For installation, please refer to the official Plesk plugin / extension store where the plugin is categorized under `Security` for easier search. To uninstall the plugin, click on the plugin and select the option `uninstall`.

Manage via Command-Line

This approach is not adviced for regular non-technical customers and should only be used under consideration of the risks. In order to manage the plugin via the Plesk command line, please refer to the detailed instruction page on the official Github repository: https://github.com/nimbusec-oss/nimbusec-plesk.

Usage

Setup

After successfully installing the Nimbusec Plesk Extension, an initial licence screen is presented as below.

Licence Screen

Image not found or type unknown

In order to gain access to the plugin, one must select either of two options and obtain the necessary permissions correspondingly.

Get a Licence

A licence containing the required permissions alongs with valid access credentials can be purchased through the official Plesk extensions store: https://www.plesk.com/extensions/nimbusec-agent-integration/. After receiving the licence, the correct option in the licence screen must be selected to active the extension and download the Nimbusec Agent.

Enter API credentials

Please enter your API Key and Secret and click on "Download Server Agent" to intiate the activation. If you don't have API credentials yet, get in contact with plesk@nimbusec.com or issue a support ticket on the official website nimbusec.com for purchase or trial subscriptions.

Settings

The settings screens allow for various configuations the extension:

- Register and unregister Plesk domains in the Nimbusec Portal
- View the Nimbusec Agent Configuration
- Configure schedule and advanced options for the Nimbusec Agent
- Activate the Nimbusec Agent

Register and unregister Plesk domains

Domains Settings

Image not found or type unknown

The register view shows all domains which are available within your Plesk installation. In order to allow the Nimbusec Agent to scan your domains you must first register them with a plan (e.g link them). To conduct this, select one or more domain you want to register, select the wished plan and click on "Register the selected domains".

The domains will be registered and are moved below the correct plan. From there it is possible to unregister the domains.

In the unregister view you see all domains which are already registered with Nimbusec and grouped by their corresponding plan. In case you want the Nimbusec Agent to stop scanning your domains, select the domains you wish to unregister and click on "Unregister the selected domains".

View the Nimbusec Agent Configuration

The Agent Congfiguration view shows you the current configuration file which is used by the Nimbusec Agent every time it starts scanning. This will give you the possibility of verifying the Agent's functionality. For more information about the agent configuration file, please refer to:

https://kb.nimbusec.com/Server Agent/agent-configuration.

Agent Settings
Image not found or type unknown

Set Run Settings for the Nimbusec Agent

In order to start the Nimbusec Agent's scanning process, it must be **activated**. This can be done by checking the "Status" checkbox and clicking on "Save settings" afterwards.

Additionally, the Nimbusec Agent must be set to a specific schedule interval. This means basically how often the Nimbusec Agent should scan every day. For that you can choose between one, two, three or four times a day. Again, save the settings afterwards by clicking the "Save settings" button.

Update Agent

This screen gives you an overview about your current Nimbusec Agent installation. The extension will check for an updated version in the background and will show a green/success or yellow/warning message on top of the page in case of an available update. Please check back to this page from time to time.

Dashboard

The Dashboard is very new and shows the issues found on your monitored websites. From there you have the option to

- view the possible malicious files,
- move them to quarantine
- remove from quarantine back to their origin
  - *this is much likely the case, if a file was moved that is crucial for the website to run*

Dashboard
Image not found or type unknown

License Key for Support

Sometimes you'll want to ask our support team questions about your Plesk installation. In that case it helps if you also provide the License Key / Subscription ID to us. Because of the General Data Privacy Regulations (GDPR) we do not get any personal / contact data from Plesk, and every entry

is pseudonymisied. Only the subscription ID is provided which we use for invoicing Plesk and also provide support if needed.

So here's how to get this information:

- Login to Plesk as admin
- Go to: Tool&Settings > License Management > Additional license key
- Find by “Key name” needed license
- Copy the content of column “Key number” for this license