

Fonksiyonlar

Bu özellik, TheHive'in 5.1 ve daha yüksek sürümleriyle kullanılabilir.

Fonksiyonlar, harici uygulamaları doğrudan TheHive işlemlerine entegre etmenizi sağlar.

Bir Fonksiyon, TheHive içinde çalışan özel bir JavaScript kod parçasıdır. Fonksiyon, dışarıdan gelen girişleri alabilir, işleyebilir ve doğrudan TheHive API'lerini çağırabilir.

Bu örneğin, verileri dönüştüren bir Python yapıştırma servisi olmadan TheHive içinde bildirimler oluşturmak için kullanılabilir.

Fonksiyon Oluştur

Sisteminizde bir olay meydana geldiğinde TheHive'da bir uyarı oluşturmak istediğinizi varsayalım. Harici sisteminiz için olaylar için kendi şemasına sahipsiniz, şöyle bir şey:

```
{
  "eventId": "d9ec98b1-410f-40eb-8634-cfe189749da6",
  "date": "2021-06-05T12:45:36.698Z",
  "title": "An intrusion was detected",
  "details": "An intrusion was detected on the server 10.10.43.2",
  "data": [
    { "kind": "ip", "value": "10.10.43.2", "name": "server-ip" },
    { "kind": "name", "value": "root", "name": "login" },
    { "kind": "ip", "value": "92.43.123.1", "name": "origin" }
  ]
}
```

Bu format, TheHive ile aynı değil, bu yüzden verileri TheHive uygun uyarı formatına dönüştürmeniz gerekiyor.

Bir org-admin olarak, bu girdiyi alabilir, TheHive formatına dönüştürebilir ve bundan bir uyarı oluşturabilirsiniz.

Fonksiyonun kodu şöyle olabilir:

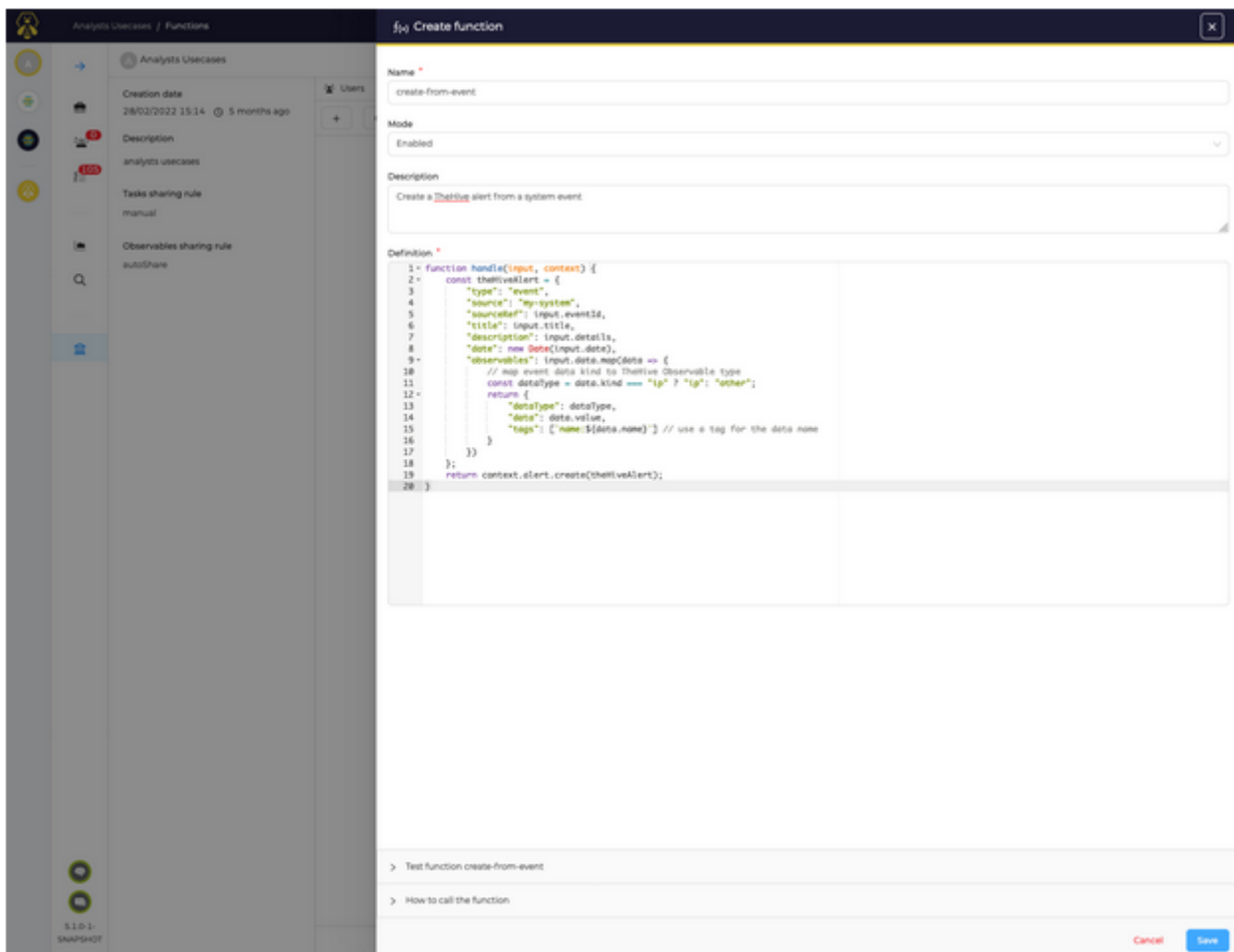
```
function handle(input, context) {
  const theHiveAlert = {
```

```

"type": "event",
"source": "my-system",
"sourceRef": input.eventId,
"title": input.title,
"description": input.details,
"date": (new Date(input.date)).getTime(),
"observables": input.data.map(data => {
  // map event data kind to TheHive Observable type
  const dataType = data.kind === "ip" ? "ip": "other";
  return {
    "dataType": dataType,
    "data": data.value,
    "tags": [`name:${data.name}`] // use a tag for the data name
  }
})
};

// call TheHive APIs, here alert creation
return context.alert.create(theHiveAlert);
}

```



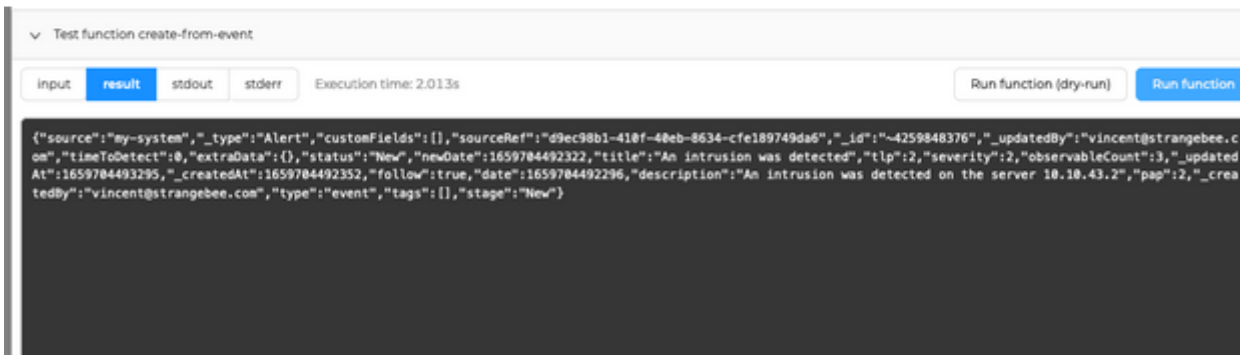
Bir fonksiyon, üç moddan birinde olabilir:

- **Etkin:** Çağrıldığında fonksiyon çalıştırılacaktır.
- **Devre Dışı:** Çağrıldığında fonksiyon çalıştırılmayacaktır.
- **Kuru Çalıştırma:** Fonksiyon çalıştırılacak ancak TheHive'ta hiçbir varlık oluşturulmayacak veya değiştirilmeyecek. Varlık oluşturma işlemleri null dönecektir. Bu, entegrasyonunuzu canlıya almadan önce test etmeniz için faydalı olabilir.

Oluşturma sayfası, fonksiyonunuzu test etmenize ve uygulandığında ne döndüreceğinizi görmek için olanak sağlar. Kuru çalıştırma modunda, fonksiyon çalıştırılacak ancak hiçbir kaynak oluşturma veya değiştirme işlemi gerçekleştirilmeyecektir.



```
1 - {
2   "eventId": "d9ec98b1-410f-40eb-8634-cfe189749da6",
3   "date": "2021-06-05T12:45:36.698Z",
4   "title": "An intrusion was detected",
5   "details": "An intrusion was detected on the server 10.10.43.2",
6   "data": [
7     { "kind": "ip", "value": "10.10.43.2", "name": "server-ip" },
8     { "kind": "name", "value": "root", "name": "login" },
9     { "kind": "ip", "value": "92.43.123.1", "name": "origin" }
10  ]
11 }
```



```
{ "source": "my-system", "type": "Alert", "customFields": {}, "sourceRef": "d9ec98b1-410f-40eb-8634-cfe189749da6", "id": "-4259848376", "_updatedBy": "vincent@strangebee.com", "timeToDetect": 0, "extraData": {}, "status": "New", "newDate": "1659704492322", "title": "An intrusion was detected", "tlp": 2, "severity": 2, "observableCount": 3, "_updatedAt": "1659704493295", "_createdAt": "1659704492352", "follow": true, "date": "1659704492296", "description": "An intrusion was detected on the server 10.10.43.2", "pap": 2, "_createdBy": "vincent@strangebee.com", "type": "event", "tags": [], "stage": "New" }
```

Fonksiyon Çağırma

Fonksiyon kaydedildikten sonra sisteminizden bir http çağrısı ile çağrılabilir:

```
curl -X POST -H 'Authorization: Bearer $API_KEY' https://<thehive_url>/api/v1/function/<function_name> -H 'Content-Type: application/json' --data '{
  "eventId": "d9ec98b1-410f-40eb-8634-cfe189749da6",
  "date": "2021-06-05T12:45:36.698Z",
  "title": "An intrusion was detected",
  "details": "An intrusion was detected on the server 10.10.43.2",
  "data": [
```

```
{
  "kind": "ip", "value": "10.10.43.2", "name": "server-ip" },
  {"kind": "name", "value": "root", "name": "login" },
  {"kind": "ip", "value": "92.43.123.1", "name": "origin" }
]
```

TheHive, girdinizi (HTTP çağrısının gövdesi), fonksiyonunuzun tanımını alacak ve girdiyle birlikte fonksiyonu çalıştıracaktır. HTTP çağrısına, fonksiyon tarafından döndürülen verilerle yanıt verecektir.

Örnek: Bir Splunk uyarısından bir alarm oluşturma

Bir Splunk uyarısı oluştururken, bir eylem olarak bir webhook tanımlayabilirsiniz. Bu nedenle uyarı tetiklendiğinde webhook, bir yük ile çağrılır. Ancak yük, Splunk tarafından tanımlanmış ve değiştirilemez.

Yük biraz böyle görünmelidir:

```
{
  "sid": "rt_scheduler__admin__search__RMD582e21fd1bdd5c96f_at_1659705853_1.1",
  "search_name": "My Alert",
  "app": "search",
  "owner": "admin",
  "results_link":
    "http://8afeb4633464:8000/app/search/search?q=%7Cloadjob%20rt_scheduler__admin__search__RMD582e21fd1bdd5c96f_at_1659705853_1.1%20%7C%20head%201%20%7C%20tail%201&earliest=0&latest=now",
  "result": {
    "_time": "1659705859.827088",
    "host": "8afeb4633464",
    "source": "audittrail",
    "sourcetype": "audittrail",
    "action": "edit_search_schedule_priority",
    "info": "granted",
    "user": "admin",
    "is_searches": "0",
    "is_not_searches": "1",
    "is_modify": "0",
    "is_not_modify": "1",
    "_confstr": "source::audittrail|host::8afeb4633464|audittrail",
    "_indextime": "1659705859",
    "_kv": "1",
```

```
"_raw": "Audit:[timestamp=08-05-2022 13:24:19.827, user=admin, action=edit_search_schedule_priority,
info=granted ]",
"_serial": "1",
"_si": [
"8afeb4633464",
"_audit"
],
"_sourcetype": "audittrail",
"_subsecond": ".827088"
}
}
```

Bu splunk uyarısını bir TheHive uyarısına dönüştürmek için aşağıdaki gibi bir işlev kullanılabilir:

```
function handle(input, context) {
  const theHiveAlert = {
    "type": "splunk",
    "source": input.search_name,
    "sourceRef": input.result._serial,
    "title": `Splunk Alert triggered: ${input.search_name} by ${input.result.sourcetype}`,
    "description": `Alert created by splunk search '${input.search_name}:\n${input.result._raw}'`,
    "date": (new Date(parseFloat(input.result._time)*1000)).getTime(),
    "observables": [
      {"dataType": "hostname", "data": input.result.host},
      {"dataType": "other", "data": input.result.action, "message": "action"},
      {"dataType": "other", "data": input.result._raw, "message": "raw"}
    ]
  };
  return context.alert.create(theHiveAlert);
}
```

Splunk'ta, web kancası url'sini TheHive işlev url'sine ayarlamanız gerekecektir.

Örnek: Soğuk dava otomasyonu

Çağırıldığında, bu işlev şunları yapacaktır:

New veya *InProgress* olan ve son bir ay içinde güncellenmemiş tüm vakaları bulun. Bu vakaların her birine bir *cold-case* etiketi ekleyin.

```
function handle(input, context) {
  const now = new Date();
```

```

const lastMonth = new Date();
lastMonth.setMonth(now.getMonth() - 1);
const filters = [
  {
    _name: "filter",
    _and: [
      {
        _or: [{ _field: "stage", _value: "New" }, { _field: "stage", _value: "InProgress" },]
      },
      {
        _lt: { _field: "_updatedAt", _value: lastMonth.getTime() }
      }
    ]
  }
];
const list = context.caze.find(filters);
const authorizedCases = list
  .filter(caze => caze.userPermissions.indexOf("manageCase/update") > 0);
console.log(authorizedCases.map(c => c.number));
console.log(`Will update ${authorizedCases.length} cases`);

authorizedCases.forEach(caze => {
  context.caze.update(caze._id, { addTags: ["cold-case"] })
});
}

```

Context API

Bilgi: Bağlam API'sindeki nesneler v1 Http Api'sinde kullanılanlarla aynıdır. Her nesnenin beklenen alanları hakkında daha fazla bilgi için lütfen Http Api Dokümantasyonuna bakın

Kullanıcı

userId: string : fonksiyonu çalıştıran kullanıcının kullanıcı adı

userName: string: fonksiyonu çalıştıran kullanıcının adı

Http isteği

request.queryString() : *Record<string, string[]>*: Harita olarak biçimlendirilmiş istek sorgu dizesini içeren sözlük

request.getQueryString(key: string): string | null: Sorgu dizesinden bir değer alın

request.getHeader(name: string): string | null: İstekten bir başlığın değerini alır

request.headers(): Record<string, string>: İstek başlıklarını alır
request.contentType: string: Content-Type istek başlığının değeri
request.remoteAddress(): Arayanın ip adresini alın

Sorgu(query))

query.execute(query: any[]): Veritabanı üzerinde bir sorgu çalıştırır (bkz. Api dokümanları => query)

Uyarı(Alert)

alert.create(input: InputCreateAlert): OutputAlert
alert.get(id: string): OutputAlert
alert.update(InputUpdateAlert): OutputAlert
alert.delete(alertId: string): void
alert.createCase(alert: InputCreateAlert): OutputCase
alert.bulkDelete(input: {ids: string[]}): void
alert.mergeWithCase(alertId: string, caseId: string): OutputCase
alert.bulkMergeWithCase({caseId: string, alertIds: string[]}): OutputCase
alert.followAlert(alertId: string): OutputAlert
alert.unfollowAlert(alertId: string): OutputAlert
alert.importInCase(alertId: string, caseId: string): OutputAlert
alert.bulkUpdate(input: {ids: string[]} & InputUpdateAlert): void
alert.find(query: any[]): OutputAlert[]

Case

case java'da ayrılmış bir anahtardır, bu nedenle bunun yerine caze kullanılır.

caze.create(input: InputCreateCase): OutputCase
caze.get(idOrNumber: string): OutputCase
caze.update(idOrNumber: string, update: InputUpdateCase): void
caze.merge(ids: string[]): OutputCase
caze.delete(idOrNumber: string): void
caze.changeCaseOwnership(idOrNumber: string, update: InputChangeCaseOwnership): void
caze.unlinkAlert(caseId: string, alertId: string): void
caze.mergeSimilarObservables(caseId: string): void
caze.bulkUpdate(update: {ids: string[]} & InputUpdateCase): void
caze.bulkApplyCaseTemplate(update: {ids: string[]} & InputApplyCaseTemplate): void
caze.find(query: any[]): OutputCase[]

Görevler(Task)

task.get(id: string): OutputTask
task.update(idOrName: string, update: Partial<OutputTask>): void
task.delete(id: string): void

task.find(query: any[]): OutputTask[]
task.setActionRequired(taskId: string, orgId: string): void
task.setActionDone(taskId: string, orgId: string): void
task.isActionRequired(taskId: string): Kayıt<string, bool>
task.createInCase(caseId: string, task: InputTask): OutputTask
task.bulkUpdate(update: {ids: string[]} & Partial<OutputTask>): void

Log

log.create(taskId: string, log: InputCreateLog): OutputLog
log.update(logId: string, update: InputUpdateLog): void
log.delete(logId: string): void
log.deleteAttachment(logId: string, attachmentId: string): void
log.find(query: any[]): OutputLog[]

Gözlemlenebilir (Observable Type)

observable.createInCase(caseId: string, observable: InputObservable): OutputObservable
observable.createInAlert(alertId: string, observable: InputObservable): OutputObservable
observable.bulkUpdate(update: {ids: string[]} & Partial<OutputObservable>)
observable.get(idOrName: string): OutputObservable
observable.update(id: string, update: Partial<OutputObservable>): void
observable.delete(id: string): void
observable.find(query: any[]): OutputObservable[]
observable.updateAllTypes(fromType: string, toType: String): void

Gözlemlenebilir Tip

observableType.get(id: string): OutputObservableType
observableType.delete(id: string): void
observableType.create(ot: InputObservableType)
observableType.find(query: any[]): OutputObservableType[]

CustomField

customField.list(): OutputCustomField[]
customField.update(idOrName: string, update: Partial<OutputCustomField>): void
customField.delete(idOrName: string): void
customField.create(cf: InputCustomField): OutputCustomField
customField.find(query: any[]): OutputCustomField[]

Vaka Şablonu(Case Template)

caseTemplate.get(idOrName: string): OutputCaseTemplate
caseTemplate.update(idOrName: string, update: Partial<InputCaseTemplate>): void
caseTemplate.delete(idOrName: string): void
caseTemplate.create(template: InputCaseTemplate): OutputCaseTemplate

caseTemplate.find(query: any[]): OutputCaseTemplate[]

Prosedür(Procedure)

procedure.bulkCreateInCase(caseld: string, input: {procedures: InputProcedure[]}): OutputProcedure[]

procedure.bulkCreateInAlert(alertId: string, input: {procedures: InputProcedure[]}): OutputProcedure[]

procedure.createInCase(caseld: string, procedure: InputProcedure): OutputProcedure

procedure.createInAlert(alertId: string, procedure: InputProcedure): OutputProcedure

procedure.update(id: string, procedure: Partial<OutputProcedure>): void

procedure.delete(id: string): void

procedure.find(query: any[]): void

Vaka Durumu(Case Status)

caseStatus.create(input: InputCreateCaseStatus): OutputCaseStatus

caseStatus.update(idOrName: string, update: InputUpdateCaseStatus): void

caseStatus.delete(idOrName: string): void

caseStatus.find(query: any[]): OutputCaseStatus[]

Uyarı Durumu (Alert Status)

alertStatus.create(input: InputCreateAlertStatus): OutputAlerttatus

alertStatus.update(idOrName: string, update: InputUpdateAlertStatus): void

alertStatus.delete(idOrName: string): void

alertStatus.find(query: any[]): OutputAlerttatus[]

Yorum(Comment)

comment.createInCase(caseld: string, comment: InputCreateComment): ÇıktıYorum

comment.createInAlert(alertId: string, comment: InputCreateComment): ÇıktıYorum

comment.update(id: string, update: InputUpdateComment): void

comment.delete(id: string): void

comment.find(query: any[]): OutputComment[]

Paylaş(Share)

share.setCaseShares(caseld: string, input: InputCreateShares): OutputShare[]

share.removeSharesFromCase(caseld: string, input: InputRemoveShares): void

share.removeShare(shareId: string): void

share.removeShares(input: {ids: string[]}): void

share.removeTaskShares(taskId: string, input: InputRemoveShares): void

share.removeObservableShares(observableId: string, input: InputRemoveShares): void

share.listShareCases(caseld: string): OutputShare[]

share.listShareTasks(taskId: string): OutputShare[]

share.listShareObservables(observableId: string): OutputShare[]

share.shareCase(caseld: string, input: InputCreateShare): OutputShare
share.shareTask(taskId: string, input: InputCreateShare): OutputShare
share.shareObservable(observableId: string, input: InputCreateShare): OutputShare
share.updateShare(shareId: string, update: InputUpdateShare): void

Organizasyon (Organisation)

organisation.get(orgIdOrName: string): OutputOrganisation
organisation.create(org: InputCreateOrganisation): OutputOrganisation
organisation.update(orgIdOrName: string, update: InputUpdateOrganisation): void
organisation.bulkLink(orgIdOrName: string, links: InputOrganisationBulkLink): void
organisation.listLinks(orgIdOrName: string): OutputOrganisationLink[]
organisation.listSharingProfiles(): OutputSharingProfile[]
organisation.link(orgA: string, orgB: string, link: InputOrganisationLink | null): void
organisation.unlink(orgA: string, orgB: string): void
organisation.find(query: any[]): OutputOrganisation[]

Profil(Profile)

profile.get(idOrName: string): OutputProfile
profile.update(profileIdOrName: string, update: InputUpdateProfile): void
profile.delete(profileIdOrName: string): void
profile.create(profile: InputCreateProfile): ÇıktıProfili
profile.find(query: any[]): OutputProfile[]

Özel Etkinlik (Custom Event)

customEvent.createInCase(caseld: string, input: InputCreateCustomEvent): OutputCustomEvent
customEvent.update(id: string, update: InputUpdateCustomEvent): void
customEvent.delete(id: string): void
customEvent.find(query: any[]): OutputCustomEvent[]

Fonksiyon (Function)

function.create(function: InputCreateFunction): OutputFunction
function.update(functionIdOrName: string, update: InputUpdateFunction): void
function.delete(functionIdOrName: string): void
function.find(query: any[]): OutputFunction