Domain model

Glossary

When we use these words, here's what we mean.

- **Team:** A group of users who use our products to help identify security weaknesses within their own organization, i.e.; a red team.

- **Target:** A potential victim/mark within an organization (team members, employees, shareholders, external contractors).
- **Organization/Org:** An organizational entity (companies, nonprofits, clubs).
- **Department/Dept:** A unit within an organization (domains, teams).
- **Peers:** A relationship with a target in the same organization (colleagues, managers).
- **Associate/Association:** A relationship between individuals outside of the organization (friends, family, enemies).
- **Affiliate/Affiliation:** A relationship between organizations (subsidiaries, partners)
- **Campaign:** A series of attacks over a period of time toward an organization (see Military Campaign).
- **Objective:** The high-level objective of a campaign: who to attack and when.
- **Objective Goal:** The thing you're trying to achieve with the objective: e.g. "click phishing link"
- **Campaign activity:** Activity from the target or team during the campaign (target clicked a link, team member canceled the campaign)
- **Attack**: An attempt to breach a target.
- **Attack log:** Consequential events occurred on both the AttackService and the Target during the attack (email sent, target clicked a link, target replied to email)
- **Happening:** Things to attend (company events, meetings, birthdays)
- **Location:** Places you can be (home, office, a place you partied once)
- **Drop/Loot/Breach/Phish/Catch (Informal):** Information that a target dropped/gave away during a campaign.

Domain diagram

This high-level diagram describes the entities and the relationships between our different subdomains, together with shared touchpoints.

```
graph LR
  subgraph Profile_Service[Profile Service]
```

```
  Department -- org_id --> Organization
  Department -- dept_location_id? --> Location
  Department -- parent_dept_id? --> Department

  Individual -- org_id --> Organization
  Individual -- dept_id? --> Department
  Individual -- home_location_id? --> Location

  Peer -- org_id --> Organization
  Peer -- of_individual_id --> Individual
  Peer -- to_individual_id --> Individual

  Associate -- individual_id --> Individual
  Affiliate -- org_id --> Organization

  Happening -- org_id? --> Organization
  Happening -- individual_id? --> Individual
  Happening -- dept_id? --> Department
end

subgraph Attack_Service[Attack Service]
  AttackLog -- attack_id --> Attack

  Attack -- org_id --> Organization
  Attack -- org_id + email --> Individual
end

subgraph Dashboard
  ProductInvite
  Team -. org_id .- Organization

  TeamMembership -- team_id --> Team
  TeamInvite -- team_id --> Team

  Campaign -- team_id --> Team
  Campaign -. org_id + email .- Individual
  Campaign -. org_id .- Organization
  Campaign -- objective_id --> Attack

  CampaignActivity -- team_id --> Team
  CampaignActivity -- campaign_id --> Campaign
  CampaignActivity -- attack_id? --> Attack
  CampaignActivity -- attack_log_id? --> AttackLog
end

subgraph Auth0
  User -- email --> ProductInvite
  User -- email --> TeamInvite
  User -- sub --> TeamMembership
end
```

> **Note:** The dashed lines between the “Dashboard” and “Profile Service” just aim to show how an entity may be identified in the profile service. The Dashboard cannot read any data from the Profile Service directly.

Dashboard

```
classDiagram
direction LR

class Team {
  +String team_id
  +String name
  +String org_name
  +String[] domain_names
  +String[] social_links
}

class Campaign {
  +String campaign_id
  +String team_id
  +Objective objective
  +Map<email, attack_id> attacks
}

class Objective {
  +String campaign_id
  +String goal
  +DateTime begins_at
  +DateTime expires_at
  +TargetOrganization target_org
  +TargetIndividual[] targets
}

class CampaignActivity {
  +String activity_id
  +String attack_id?
  +String attack_log_id?
  +String team_id
  +String activity_type
  +JSON payload
}

class TargetIndividual {
  +String email
  +String called_name?
  +String[] social_links?
}

class TargetOrganization {
```

```
  +String org_id
  +String name
  +String[] domain_names?
  +String[] social_links?
}

class Organization~ProfileService~ {
  +String org_id
  ...
}

class Attack~AttackService~ {
  +String attack_id
  +String org_id
  +String campaign_id
  ...
}

class AttackLog~AttackService~ {
  +String attack_log_id
  +String attack_id
  ...
}

Team .. Organization : team_id is org_id
Campaign .. Attack : campaign_id
Campaign .. Organization : org_id

AttackLog --> Attack : attack_id

CampaignActivity .. Attack : attack_id
CampaignActivity --> Campaign : campaign_id
CampaignActivity .. AttackLog : attack_log_id

Objective <-- Campaign : objective

TargetOrganization <-- Objective : target_org
TargetIndividual <-- Objective : targets
```

Discovery

Service domain boundaries

This is an exercise to figure out the “outermost bound of the domain” for each service.

Given the above domain diagram, let’s discuss and answer these questions:

1. Which entities are required in each service?

2. Which entities are shared between services?
3. Which service should “own” which entity?

Once we have answered these questions, we can split the domain diagram up into different sections for each service! 🎉

Entities that are “shared” will have their own version of the diagram, but the “data owner” would be considered the “source of truth” for that data, and other services will just have denormalized copies of the data or will fetch data directly from their APIs.

Profile data service

Owns: Individuals, Organizations, + their relationships

Receives: IndividualSeed, OrganizationSeed

Attack service

Owns: Attack, AttackLog

Fetches: Individuals (+ Relationships), Organizations (+ Relationships)

Receives: Campaign

Copies: Campaign metadata

Dashboard

All copies are denormalized and “safe for consumption”.

Owns: Campaign, CampaignActvitity, Team, TeamInvite, TeamMembership, ProductInvite

Fetches: Attack, AttackLog

DEPRECATED: Monolithic domain diagram

Left here as a copy/paste source for

- The data will be used to train an AI agent with a “background story”.
- Data should be easily cached, navigated, and flexible for experimentation.
- It should efficiently model relationships between targets and marks.

```
classDiagram

  class Organization {
    + UUID id
    + UUID office_location_id
    + String name
    + String industry?
    + String timezone?
    + String spoken_language?
    + String working_style?
    + String[] identifiers

    + Campaign[] Campaigns
    + Affiliate[] Affiliations
    + Associate[] Associates
    + Happening[] Happenings
  }

  class Happening {
    + UUID id
    + UUID org_id?
    + UUID dept_id?
    + UUID target_id?
    + UUID location_id?
    + String type
    + String name
    + String description?
    + String cadence?
    + String working_style?
    + Date start_date
    + Time start_time?
    + Duration length?
    + Date end_date?
  }

  class Target {
    + UUID id
    + UUID org_id
    + UUID dept_id?
    + UUID home_location_id?
    + String spoken_language?
    + String role_title?
    + String pronouns?
    + String called_name?
    + String first_name?
    + String last_name?
    + Date date_of_birth?
    + String[] identifiers

    + Contact[] Contacts
    + Associate[] Associations
    + Campaign[] Campaigns
```

```
  + Happening[] Happening
}

class Department {
  + UUID id
  + UUID org_id
  + UUID parent_department?
  + String working_style?
  + String name?
  + String local_name?
  + String primary_field?
  + String location_id?
  + String spoken_language?
}

class Contact {
  + UUID target_id
  + UUID contact_target_id
  + UUID org_id
  + String relationship
  + Number proximity
}

class Associate {
  + UUID org_id
  + UUID target_id
  + String role
  + String description
}

class Affiliate {
  + UUID org_id
  + UUID affiliate_org_id
  + String role
  + String description
}

class Campaign {
  + UUID id
  + UUID org_id
  + UUID target_id
  + String name
  + JSON plan
}

class AttackInteraction {
  + UUID id
  + UUID org_id
  + UUID target_id
  + UUID attack_id
  + DateTime happened_at
  + String type
  + JSON payload
}
```

```
class Location {
  + UUID id
  + String type
  + String spoken_language?
  + String name?
  + String address_line_1?
  + String address_line_2?
  + Number lat?
  + Number lng?
  + String city?
  + String country?
  + String timezone?
}


Organization --* Location : office_location_id

Department --* Target : dept_id
Department --* Organization : org_id
Department --* Location : location_id

Target --* Organization : org_id
Target --* Location : home_location_id

Contact --* Organization : org_id
Contact --* Target : target_id
Contact --* Target : contact_target_id

Associate --* Target : target_id
Associate --* Organization : org_id

Affiliate --* Organization : org_id
Affiliate --* Organization : affiliate_org_id

Campaign --* Target : target_id
Campaign --* Organization : org_id

Happening --o Organization : org_id?
Happening --o Target : target_id?
Happening --o Department : dept_id?
```

Organization profiles

- Name
- Industry
- Timezone

- Departments
- Offices, Locations
- Affiliates (e.g. which bank(s), payment systems, other affiliates, sub companies)

Personal profiles

Information that can be useful for phishing, might overlap with “sensitive data”

- Spoken language
- Email address
- timezone
- Full name
- Date of birth
- Home address
- Phone number
- Social Security number or national identification number
- Driver's license number
- Passport number
- Credit card number
- Bank account number
- Company or organization name
- Job title or position
- Personal interests or hobbies
- Family members or friends
- Recent purchases or transactions
- Travel history or plans
- Medical information
- Income or financial information

- text snippets of the language we'll need to use when phishing written by the person

Sensitive data ("prizes")

What an attacker would hope to obtain.

- passwords, SSH keys, and other keys: obvious high-value prizes
- software & versions: unveils new attack surfaces
- intellectual property: source code other intellectual assets
- personal information: improves quality of phishing emails
  - "Hi X, could you tell me where Y lives? It's for a birthday present…"
- internal memos: improves the quality of phishing emails
  - "Could you tell me when we are releasing X? I forgot…"
- access requests: ("My X@at-work.com email is temporarily blocked! Could you give me access to our docs to my personal email? xoxo")

Profile data sources

- Buying data on organizations and targets
- Organization websites
- Personal websites/blogs of targets
- Social media profiles for organizations and targets
  - LinkedIn
  - Facebook pages + related pages (e.g. fan pages)
  - Instagram
  - Twitter

This document discusses the domain model for Mole Security. It attempts to define the high-level concepts, taxonomy, and data contracts within the domain.