What are micro-perimeters in the Zero-Trusted Networks approach?

Jochen Kressin · 2020-10-01

The term "Zero-Trust Networks" seems to be the buzz at the moment and often seen as the holy grail to solve modern-day security problems. However, Zero Trust is not a single product, process, or architecture. It's more a concept than anything else, and, very simply speaking, means, well, to trust no one. That's, of course, not very helpful. In this mini-article series, we look at the key elements of the Zero Trust approach. We also explain how to apply them for Elasticsearch, Kibana, Logstash and the Elastic Stack.

What is traditional perimeter security?

Before we dig into how micro perimeters can improve your overall security architecture, let us first examine the traditional perimeter security approach.

The perimeter security approach makes two assumptions about your network:

1. Traffic that comes from outside your network is untrusted and must be validated and restricted
2. Traffic inside your network can be trusted by default

The perimeter is basically the border between two networks, here, the "outside network" (think: the internet) and the "inside network" (think: your datacenter or your cloud installation).

> Verify, then trust?

Any request that comes from the outside has to pass the "border controls." This can be firewalls, VPNs, IDS, IPS, SIEMs, or email gateways.

Once the request has passed those systems, it is considered secure and trusted. Hence, this approach is also called "verify, then trust."

Perimeter security is flawed

If this approach works well, why do we still see data breaches frequently? Simply because it doesn't work. The perimeter security model is flawed. It makes the false assumption that a user's identity is never compromised. And that all users act responsibly and can be trusted.

> Over 60% of attacks come from inside

A study by IBM (already conducted in 2016) revealed that over 60% of attacks come from inside your network. Not the outside. So how does perimeter security help if this is the case?

Another issue is that once a malicious user has made it into your network, the lateral movement throughout the network's applications is not restricted. The initial system a user attacks is, in most cases, not the primary target. It only acts as a door opener until the attacker reaches the final destination. Not restricting lateral movement inside the network makes it very easy to do so.

> Always verify, never trust!

So the fundamental change Zero Trust introduces is the new mantra, "Always verify, never trust."

Micro-perimeters and micro-segmentation

The definition of a perimeter of being the border between "outside" and "inside" is too broad and coarse. The antidote is micro-segmentation and micro-perimeters. Very simply speaking, this concept gives each and every service and application in your network its own dedicated perimeter.

> Move security to where your data lives

In other words, instead of defining one global perimeter which implements access controls and other security measures like encryption, each service and application has it's own. You are basically moving security to where your data actually lives. Or at least as close as possible.

Of course, you can probably not do that for every small piece of your infrastructure. So you should first identify your *protect surface*, which we will cover in another article.

A practical example

If this sounds a bit too abstract, let's look at a practical example. If you are familiar with Elasticsearch, you might know that the free Apache2-licensed version of it does not offer any security at all. Which means all data is accessible and changeable for everyone.

With the traditional perimeter-based network security approach, at first, it seems that we do not have any problem at all. We can install our Elasticsearch cluster inside the perimeter and trust that nothing goes wrong.

Security controls are too far away from the data

Even without any knowledge about Zero Trust, this already sounds a bit risky, right? Let's add a reverse proxy like Nginx in front of Elasticsearch and add some HTTP Basic Authentication. Done.

Not quite. The security controls we have implemented are too far away from the actual system that holds your data, Elasticsearch. Nginx knows nothing about the inner workings of Elasticsearch, so there are multiple possible attack vectors.

Security is detached from the system holding it

With Nginx, access control can only be implemented on a URL basis. Say, we have an index called "logs-2020-09-22" and want to restrict access to it. A simple match all query might simply look like:

```
https://elasticsearch.example.com:9200/logs-2020-09-22/_search
```

So we can implement an access control rule in Nginx that makes this URL only available for individual users identified by HTTP Basic Authentication.

But - what if someone uses a wildcard for the index name? What if someone uses "_all" as the index name? And what if there are index aliases that are added and removed dynamically? These problems cannot be solved by putting a proxy in front of Elasticsearch. The proxy simply does not know anything about wildcard queries or index aliases.

Using a micro-perimeter approach

Search Guard is architected as security and alerting plugin for Elasticsearch. This means that Search Guard can intercept and validate any traffic inside the Elasticsearch cluster. Security controls are now moved directly to the system that is housing your data. Hence, all specific aspects and details of that system can be considered when deciding whether a request should be allowed or declined. Any wildcard or index alias can now be resolved to the concrete underlying index name before applying access controls.

> Security is moving when your systems are moving

One other bonus you get from using a micro-perimeter approach is that security is literally moving with your systems. Say your Elasticsearch, Kibana and logstash setup is moving from one network segment to another. Since Search Guard is implemented as an Elasticsearch plugin, it will just move with it. In the proxy scenario above, you have to remember to move your already sub-par proxy solution with it.

Summary

The traditional perimeter security network approach does not work anymore. Implementing authentication, authorization, and access controls on the rim of the perimeter is not enough. Traffic inside

your network can not be trusted. Instead of the "Verify, then trust" strategy, you should instead implement an "Always verify, never trust" approach. One step of doing this is to move your security as close to your data and systems as possible. In an ideal world, each system will be protected by its own micro-perimeter. This ensures that lateral movement in your network is difficult.

Where to go next

- Read our post on the main concepts of Zero-Trusted Networks
- Download our slide deck explaining how Search Guard implements the Zero-Trusted Networks approach
- Download and install Search Guard to create a micro-perimeter around Elasticsearch, Logstash and the Elastic Stack

---

This article was published on the Search Guard blog: https://search-guard.com/blog/micro-perimeters-elasticsearch/
Search Guard is Security and Alerting for Elasticsearch and Kibana — search-guard.com · Start a free trial