D. Reinhardt, H. Langweg, B.C. Witt, M. Fischer et al. (Hrsg.): Sicherheit 2020,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Göttingen 2020 53

Context-based Access Control and Trust Scores in Zero Trust Campus Networks

Thomas Lukaseder[1] Maya Halter[1] Frank Kargl[1]

Abstract: Research networks are used daily by thousands of students and scientific staff both for education and research. Therefore, they contain a vast number of sensitive and valuable resources. The currently predominant perimeter security model is failing more and more often to provide sufficient protection against attackers. This paper analyses to what extent the *Zero Trust Model* that is popular in some commercial networks can also be applied to the open and heterogeneous research network of a German university. The concept presented herein to implement such an identity-based network model focuses in particular on the components which are necessary for authentication and authorization. The feasibility of the model is demonstrated by a self-implemented prototype that protects access control to a prominent eLearning system called Moodle. Non-functional performance tests show an increase in performance compared to the current system where access control is only conducted inside the web application. The Zero Trust Model enables the determination of the trustworthiness of individual identities and thus offers valumable new ways to secure a research network.

Keywords: Network Security; Network Management; Zero-Trust; Trust Scores; Subjective Logic

1 Introduction

Research networks on university campuses are used daily by thousands of students and scientific staff for education and research. This means that they contain a large number of sensitive and valuable resources such as grades or research results. Many users bring their own computers, smartphones, or IoT devices. As device security is often not centrally managed but left to device owners, this creates a rather large attack surface.

These are some reasons why the currently predominant perimeter security models — relying mostly on firewalls and securing access to the network — fail more and more often to provide sufficient protection against attackers. Some organizations have realized this early. One example is Google that was hit by severe attacks from nation state attackers in 2009. The attack was termed "Operation Aurora"[2] and targeted GMail accounts of Chinese dissidents along with source code repositories. Googles reacted with the project *"BeyondCorp"*, a complete restructuring of their network security approach based on the idea of Zero Trust. They describe the design, implementation, and roll-out in a series of articles [Be17;

[1] Universität Ulm, Institut für Verteilte Systeme, Albert-Einstein-Allee 11, 89081 Ulm, Germany, firstname.lastname@uni-ulm.de

[2] https://en.wikipedia.org/wiki/Operation_Aurora

doi:10.18420/sicherheit2020_04

Es17; Ja18; Os16; Sp16; WB14]. The core is an access proxy that resides in front of the service and controls all accesses to services. The basic architecture that is also basis for our implementation can be seen in Figure 1. The access control decision itself is taken by a policy engine enforcing policies set by administrators in combination with a trust engine which calculates trust scores of users and devices accessing the network. BeyondCorp relies heavily on the exclusive use of managed devices, a policy that cannot be enforced in University networks where students and staff are heavily relying on their own, private devices. The term Zero Trust does not mean the absence of trust, it means that trust must be earned. A client wanting to access a service must explicitly establish this trust through appropriate means and demonstrate that an access can securely be granted. This *context-based access control* is typically based on a combination of checks such as device certificates, 2-factor authentication, or patch status of the accessing device.

First preliminary implementations of this concept are available, for example by DeCusatis et al. [De16] and Eidle et al. [Ei17]. The latter even describe first concepts of automatic reaction to adjust access policies in case of attacks. What both publications as well as Google's model are missing are documented concepts how trust engines are calculating a trust score based on different inputs. Moreover, none of the publications publish their software implementation.

Based on the issues identified in related work, in this paper, we present the following contributions:

- Faced with many private devices and BYOD policies, we list possible inputs for trust score calculation.
- We propose a subjective logic based concept for the trust engine component.
- We present Alekto[3], an implementation of a Zero Trust network framework.

We focus in particular on the components which are necessary for authentication and authorization and especially how to flexibly take context-based access control decisions and calculate trust scores.

We evaluate the flexibility and feasiblity of Alekto. We use a prominent eLearning system called Moodle that is frequently used in our university as a real-world example. In order to identify potential performance penalties that might occur due to the access proxy, we also conduct a detailed performance analysis. Interestingly enough, these tests show *increased* performance compared to the original system where access control is conducted by the web application itself.

Through our analysis, we could demonstrate that the Zero Trust Model enables the determination of the trustworthiness of individual identities and thus offers valuable new ways to secure a research network.

[3] `https://github.com/vs-uulm/alekto`

2 Concept

In the following, we describe the framework, the trust score calculation, and discuss possible trust score calculation inputs. The framework as described was implemented and published on github. All basic components of the concept were implemented.

2.1 Alekto

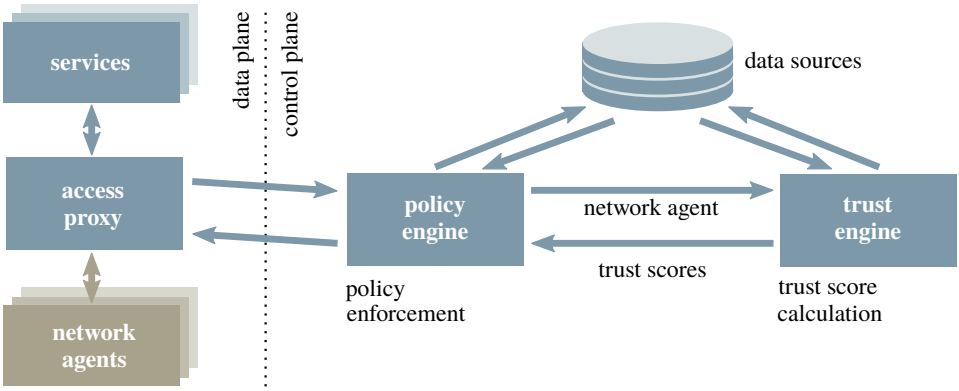

Fig. 1: Architecture of the zero-trust framework.

Figure 1 contains the basic architectural model of Alekto described in the following. It follows Google's *BeyondCorp* terminology and basic setup.

The *network agent* represents the union of a user and a device and contains both the individual trustworthiness of user and device and the trustworthiness of the combination of both. Additionally, user role and category as well as authentication method are important. This way, the security risk can be determined based on capabilities of a user.

Different *data sources* provide information about the network agent. As many independent data sources as possible should be used to mitigate the effects of compromised individual databases or sensors. An authenticatable identity of each network participant is necessary to link the data sources. This allows the storage of uniquely assignable information of a network agent. Basically, a distinction is made between two data source types and associated data base types. Inventory data bases are used to store the current state while historical data bases store past activities of a resource or entity.

The *Services* are all university or research network services that require access control. One example of such a service are the Moodle instances of the universities.

The main task of the *access proxy* is to ensure the global availability of a service. For this purpose it is located in the data plane between client and service and manages the direct network flow between both parties. The proxy should be able to listen via different open

ports to the different protocols required for its downstream service. Access to the services are handled by a single sign-on service located on the proxy.

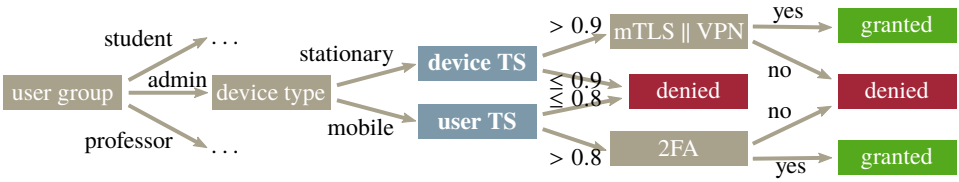

Fig. 2: Example decision tree of the policy engine.

While the access proxy in this system assumes the role of the executive in the sense of separation of powers (i. e. executing the decisions), the *policy engine* represents the judiciary (i. e. judging the network agent). It receives the requests of the applicants forwarded by the Access Proxy and decides on their access rights based on the applicable guidelines. The policy engine is responsible for collecting all information about the requesting entity required for the evaluation of the guidelines and represents it in a network agent. Policies can be defined in form of yaml files. An example for such a policy yaml file can be seen in the appendix (List 4). While the policy engine is very powerful in and of itself, the decisions it can make are exclusively binary. To facilitate more complex decisions, the policy engine is supported by the *trust engine*. The policy engine forwards its created network agent to the trust engine, from which it then receives the trust score determined based on the agent properties. Figure 2 shows an example decision tree the policy engine could follow. The policy engine can feature different policies for different user groups such as admins, students, or professors. Data sources offer a large variety both of device and user properties such as device type or whether the device uses a VPN, two factor authentication (2FA), or mutual TLS. This data can be used for both the trust score calculation and as singular parameters in the decision tree for binary decisions.

2.2 Trust Score Calculation

The *trust engine* is responsible for calculating the trust scores. It can only be addressed by the policy engine for this purpose. As an example, we determine a trust score based on the authentication attempts of the users. These are checked for signs that may indicate an attack or compromised user accounts and devices. We base our trust score calculation on *Subjective logic* by Audun Jøsang [Jø16]. The trust scores are determined using the *binomial opinion*, which is formed over an entity which can be a network agent, a user, or a device. *Subjective logic* is used to calculate the confidence values. Subjective logic is a probabilistic logic that explicitly considers the uncertainty and trustworthiness of a source. Subjective opinions represent the belief in the substance of a statement under consideration of a possible uncertainty. Be $Z \in \{x, \overline{x}\}$ a binary definition range with binomial random

variable $X \in Z$. A binomial opinion about the truth of value x is the ordered quadruple $\omega_x = (b_x, d_x, u_x, a_x)$, with

$$b_x + d_x + u_x = 1 \tag{1}$$

and the corresponding parameters defined as

$$b_x : \text{Belief in the assumption that } x \text{ is TRUE (i. e. } X = x), \tag{2}$$
$$d_x : \text{Belief in the assumption that } x \text{ is FALSE (i. e. } X = \overline{x}), \tag{3}$$
$$u_x : \text{Uncertainty that represents the absence of evidence,} \tag{4}$$
$$a_x : \text{Base rate, i. e. previous probability of } x \text{ without proof.} \tag{5}$$

For this work, exclusively binomial opinions were considered. Here, a statement can assume exactly two values — harmful or not harmful or trustworthy or not trustworthy. The different opinions were linked by a *cumulative belief fusion* in order to reduce the uncertainty of the individual opinions. A belief fusion is used in this work to fuse different opinions about the trustworthiness of an entity. The cumulative belief fusion describes that increased indications for or against the trustworthiness of an entity are confirmed in their statement and reduce the individual uncertainties. Several independent sources, which point to an offense, can thus increase the belief in this offense. In the case that two binomial opinions A and B show a perfect certainty ($u_x = 0$), their values are fused according to:

$$b_x^{(A \circ B)} = \frac{1}{2} \cdot (b_x^A + b_x^B) \tag{6}$$
$$u_x^{(A \circ B)} = 0 \tag{7}$$
$$a_x^{(A \circ B)} = \frac{1}{2} \cdot (a_x^A + a_x^B) \tag{8}$$

and describe the unweighted average of the two opinions. For the implementation of the trust score calculations, the evaluation of past user authentications was chosen as an example, since statements can be made about both the user and their device.

Log entries show when which user attempted authentication. Since device authentication was performed before user authentication, the attempt can also be assigned to a clearly identifiable device. The logs also describe the type of user authentication — for example LDAP login or two factor authentication. In addition, the *failed* flag can be used to indicate whether the login attempt failed. The IP address and the timestamp provide additional information about when and where the attempt took place. This information can be used to detect brute-force attacks or misuse of a user's access data and thus adjust the trustworthiness of the user and device. Two different kinds of brute-force attacks need to be examined here [HMT06; PS02]: For one an attack against a single user account with a list of possible passwords to determine that user's password. This shows up in logs as many failed login attempts for a specific user. Blocking a user account on the basis of this finding would in

turn allow a denial of service attack. The decisive advantage of the Zero Trust Model here is that it is always possible to differentiate between user and device. If such a behavior is discovered, the trustworthiness of the user can be reduced due to a possible compromise of their account. More essential is that the trustworthiness of the executing device is greatly minimized, as it can be assumed that the device is in the hand of an attacker. Finally, the network agent — i. e. the combination of user and device — must also be marked as untrusted, as this combination is likely to be the attacker. The other type of brute-force attacks is directed against a large number of user accounts simultaneously (bulk guessing), without frequent unsuccessful attempts for an account. This attack can be detected by the enormous number of logon attempts made from one device. If such a behavior is detected, trustworthiness of the executing device is reduced.

2.3 Possible Inputs for Trust Score Calculation

To calculate trust scores, Google relies heavily on parameters we cannot obtain. Patch level of private devices or results of virus scanners on the devices are not accessible. Instead, we need different parameters that help us assess the trustworthiness of network participants. In addition to the aforementioned login attempts, we identified other possible inputs that can be used for the trust score calculation.

For one, there are mechanisms that at first glance make it easy to make a binary decision if a network agent is trustworthy or not, e. g. two factor authentication, client TLS, usage of a trusted VPN service, to just name a few. However, the concrete implementation of these services (e. g. which second factor is used) can lead to different trust levels. A fingerprint scanner, an iris scanner, or face recognition technology common on many modern smartphones can support the second factor.

Depending on the service, access from a large distance can be a sign of a possible compromise; additional assurance of identity (e. g. a second factor) could then in turn be required. With rough estimation of the location based on IP, rudimentary tracking can be implemented to see if movement of the device is reasonable (e. g. if a network agent logs in from New York and from Moscow within a time span that is unreasonable for a flight).

Browser fingerprinting can be used to determine additional features of the network agent. Upathilake et al. [ULM15] analyzed and classified several fingerprinting techniques such as using the HTML5 <canvas> element to render an image and determine pixel for pixel how exactly the element was drawn. Subtle differences in the browser implementations allow us to differentiate between the browsers. This should be handled with caution as the features can be falsified. A device suddenly using a different browser to log in can be a sign of an attack. Cookies can make it easy to detect if the browser was used before to successfully log in. Similarly, the type of device (i. e. mobile or stationary) can give insights into how trustworthy it is.

3 Evaluation

We tested the prototype performance which we describe in the following. Two different test scenarios were created for this purpose. The newly developed components will be tested against the existing system. This means that the classical Moodle instance with LDAP authentication is compared to the new Zero Trust components with a Moodle instance extended by JSON web token (JWT) authentication with the goal to examine to what extent the basic performance is influenced by the intermediate Zero Trust System. The Zero Trust components are then re-tested as a target service using a minimal web application. Here, the performance of the components can be tested independently of Moodle.

3.1 Test Setup

All services used for the tests were set up as Docker[4] containers. To create the same conditions for each run, the containers are restarted before the run starts and all existing volumes are deleted after the run ends. For the Moodle instance an already existing image of Bitnami[5] was used, which was extended by a specially developed JWT authentication plugin. The Moodle web server is started with the help of a Docker-Compose file together with the corresponding MariaDB database and an LDAP server. As dummy service, a minimal HTTPS web service was written in Go, which responds to all requests only with a twenty character string.

The Zero Trust components were each created on the basis of the same base image. This image consists of golang:1.10.4[6] and the specially written *shared lib*, which contains all shared basic functions of the different components. The individual containers were then merged using Docker Compose. As LDAP server, a Docker Image by Römhild[7] was used for the Moodle instance as well as for the Zero Trust components.

Three desktop computers were used for the tests. One computer simulates the client requests, one runs Alekto and one runs the service protected by Alekto. The service machine runs either Moodle, including MariaDB database and an LDAP server for Moodle authentication via LDAP or the simple HTTPS web service. Table 2 in the appendix contains hardware specifications of the service machine and the Alekto machine.

Glances[8] was used as hardware monitoring tool. Current status is logged every two seconds. K6[9] is a JavaScript and Go based load testing tool. As one of few tools it offers HTTPS with client certificates. Unfortunately, only one global certificate can be set here for all

[4] https://www.docker.com/
[5] https://github.com/bitnami/bitnami-docker-moodle
[6] https://hub.docker.com/_/golang
[7] https://github.com/rroemhild/docker-test-openldap
[8] https://glances.readthedocs.io/en/stable/quickstart.html
[9] https://docs.k6.io/docs/welcome

requests. For the performance tests this does not matter, because the duration of a certificate validation is measured here and not the semantic meaningfulness of the requests.

3.2 Test Cases

variables	values test 1	values test 2
(1) performance	request duration, CPU, memory	request duration, CPU, memory
(2) systems	proxy + moodle, moodle (LDAP)	proxy + dummy dervice, dummy dervice
(3) # clients	5, 10, 15	25, 50, 62, 75
(4) client groups	mix	students ; admins ; mix

Tab. 1: Test cases to test the Zero Trust components with the help of a dummy service.

The individual test cases are listed in Table 1. Within these cases, the different client groups are additionally considered. The properties of the different groups can be read in the appendix in Tables 3 and 4. There is a big difference between the groups in device authentication, which only takes place in the *admin* and *mix* groups via mTLS. For standard users (*students*) only two guidelines with a total of five requirements are checked. The calculation of the trust values takes place exclusively for the user, since there is no device identity here. Highly privileged users (*admins*) are also checked for two guidelines, but these contain a total of twelve different requirements for users and devices. In addition, their device trust score is calculated based on four properties related to past authentication processes. The policies and trust value calculations are evaluated, but a negative authorization decision for the test runs is ignored and each request is forwarded to the target service to create equal test conditions.

3.3 Results

Preliminary tests already suggested, that the Moodle service was significantly slower than Alekto. First, the implementation with an intermediate proxy is examined. The Moodle instance reached its limits in terms of CPU and memory usage during the tests. It reached 13.6 requests per second ($\frac{r}{s}$) with an average CPU utilization of 51% ($\sigma = 15.79$) with 5 virtual users. Simultaneously, the proxy computer has a CPU utilization of 18% ($\sigma = 8.17$). As the number of users increases, the difference between the two computers increases. While the CPU utilization of the Zero Trust computer increases only slowly with ten ($\mu = 19.72$, $\sigma = 8.33$) and fifteen ($\mu = 20.29$, $\sigma = 8.21$) users, the service machine reaches a critical state with fifteen users with $27\frac{r}{s}$ and a CPU utilization of 93% ($\mu = 93.45$, $\sigma = 18.09$). This confirms the initial assumption that the Moodle web server is the bottleneck of the system.

In the following, the results of the implemented system are compared to the baseline. The baseline consists of the same Moodle instance, but uses the classic LDAP authentication as used in university operations. The goal of this test was to measure the impact of the

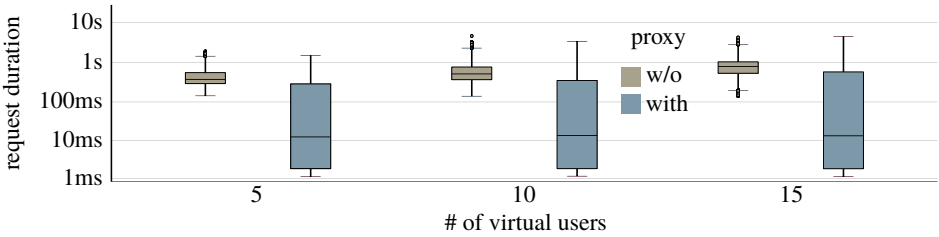

Fig. 3: Diagram of the request duration in logarithmic scale.

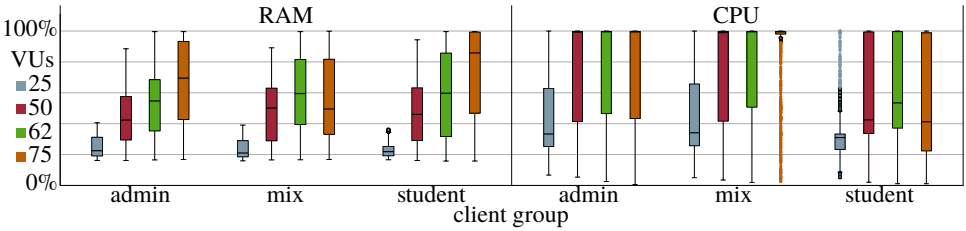

Fig. 4: Resource consumption dependent on user group and number of users.

intermediate proxy. The fact that the requests with proxy and Moodle authentication plugin (as illustrated in Figure 3) are on average shorter (μ = 130.02, σ = 210.00) than those without proxy and with LDAP authentication (μ = 442.58, σ = 230.36) was surprising. Hard to see in logarithmic scale, the response time of the Moodle service without proxy almost doubles from five users (μ = 442.58, σ = 230.37) to fifteen users (μ = 808.05, σ = 354.64).

This result clearly shows that the Zero Trust components have a positive effect on the performance of the web server in terms of response time. This can be attributed to the LDAP authentication outsourced to the single sign-on service which replaced the classic Moodle LDAP authentication. While the CPU utilization of the service machine is the same in both cases (+/- 2 %), it is noticeable that the RAM in the Zero Trust variant is higher. This already starts with five users with 4% (μ = 25.41, σ = 1.06 to μ = 21.08, σ = 0.32) and reaches 8% with fifteen users (μ = 34.196, σ = 7.37 to μ = 26.33, σ = 0.41). Since the only difference here was the authentication plugin used, it can be assumed that this result was caused by the cryptographic validation of the JWT.

Since both the CPU and the memory usage of the Zero Trust components during tests with a Moodle service reached a maximum of 20% (CPU: μ = 20.29, σ = 8.22; RAM: μ = 19.78, σ = 2.02), more detailed tests were performed with a minimal dummy web application running on the service machine instead of Moodle. The limits of the Zero Trust system were firstly explored by using the bisection method. The tests were carried out with 25, 50, 62, and 75 virtual users (VUs).

Figure 4 shows the CPU and memory utilization of the Alekto machine. Clear differences

between the user groups can be seen. This difference most likely results from the fact that students did not send any client certificates in the requests. All in all, the amount of requests per second that were possible ranged from 67 to 148 depending on user group and number of virtual users. Keeping in mind, that the system runs on a regular desktop PC and that the system has proven to be a lot more efficient than Moodle, the evaluation shows that the system scales well for practical applications.

3.4 Discussion

The non-functional performance tests showed that the performance of the baseline is not negatively affected by the use of Alekto. It has even been improved in terms of processed HTTP requests per second and their required response time. As the Moodle web server was the bottleneck of the system, the performance could be optimized by the outsourced LDAP authentication. To make a general statement about the non-functional performance of the Zero Trust components, additional tests against other target services are needed.

4 Conclusion

With regard to the current research on the implementation of the Zero Trust Model, it becomes clear that this model is mainly used in commercial enterprises with the help of proprietary software. Some companies that have successfully implemented the concept provide insights into their fundamental structures; their technologies, however, remain classified as well as the calculation of trust scores. In university networks, many of the factors often used to determine trust cannot be used, such as device patch levels.

In this work, we introduced a model on how to calculate trust scores, listed possible inputs for trust score calculation, and introduced Alekto, a Zero Trust network framework. The evaluation showed the feasibility of our approach to perform its tasks.

In the future, we plan to extend our system, plan to evaluate the system in a wider context closer to a production network, and develop recommended policies for the policy engine as well as recommendations for trust score metrics.

Acknowledgment

This work was supported in the bwNET100G+ project by the Ministry of Science, Research and the Arts Baden- Württemberg (MWK). The authors alone are responsible for the content of this paper.

References

[Be17] Beyer, B. (E.; Beske, C. M.; Peck, J.; Saltonstall, M.: Migrating to BeyondCorp: Maintaining Productivity While Improving Security. Usenix ;login: 42/2, pp. 49–55, 2017.

[De16] DeCusatis, C.; Liengtiraphan, P.; Sager, A.; Pinelli, M.: Implementing Zero Trust Cloud Networks with Transport Access Control and First Packet Authentication. In: 2016 IEEE International Conference on Smart Cloud (SmartCloud). Pp. 5–10, Nov. 2016.

[Ei17] Eidle, D.; Ni, S. Y.; DeCusatis, C.; Sager, A.: Autonomic security for zero trust networks. In: 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON). Pp. 288–293, Oct. 2017.

[Es17] Escobedo, V. M.; Zyzniewski, F.; Beyer, B. (E.; Saltonstall, M.: BeyondCorp 5: The User Experience. Usenix ;login: 42/3, pp. 38–43, 2017.

[HMT06] Hole, K. J.; Moen, V.; Tjostheim, T.: Case study: Online banking security. IEEE Security & Privacy 4/2, pp. 14–20, 2006.

[Ja18] Janosko, M.; King, H.; Beyer, B. (E.; Saltonstall, M.: BeyondCorp 6: Building a Healthy Fleet. Usenix ;login: 43/3, pp. 24–30, 2018.

[Jø16] Jøsang, A.: Subjective logic. Springer, 2016.

[Os16] Osborn, B.; McWilliams, J.; Beyer, B.; Saltonstall, M.: BeyondCorp: Design to Deployment at Google. Usenix ;login: 41/1, pp. 28–34, 2016.

[PS02] Pinkas, B.; Sander, T.: Securing passwords against dictionary attacks. In: Proceedings of the 9th ACM conference on Computer and communications security. ACM, pp. 161–170, 2002.

[Sp16] Spear, B.; Beyer, B. (E.; Cittadini, L.; Saltonstall, M.: Beyond Corp Part III: The Access Proxy. Usenix ;login: 41/4, pp. 28–32, 2016.

[ULM15] Upathilake, R.; Li, Y.; Matrawy, A.: A classification of web browser fingerprinting techniques. In: 2015 7th International Conference on New Technologies, Mobility and Security (NTMS). Pp. 1–5, July 2015.

[WB14] Ward, R.; Beyer, B.: BeyondCorp: A New Approach to Enterprise Security. Usenix ;login: 39/6, pp. 6–11, 2014.

Appendix

	service machine	Alekto machine
Processor	Intel(R) Core(TM) i5-4590	Intel(R) Core(TM) i5-4590
RAM	DIMM DDR3 8GiB	DIMM DDR3 8GiB
HDD	250GB	500GB
OS	Ubuntu 18.04.2 LTS	Ubuntu 18.04.2 LTS

Tab. 2: Hardware properties of the test environment.

Authentication	Students	Admins	Mix (Students, Admins)
user	LDAP	LDAP	LDAP
device	-	mTLS	mTLS

Tab. 3: Type of user and device authentication by client group. In the *mix* group, each request is authenticated using mTLS.

Authentication	Students	Admins	Mix (Students, Admins)
# policy variables	2	2	2
# policy requirements	5	12	5-12
# trust device variables	0	1	1
# trust device opinions	0	4	4
# trust user variables	1	1	1
# trust user opinions	3	3	3

Tab. 4: Number of required evaluations of the policy and trust engine by client group.

```
#global policy for admin users
- kind: policy
  metadata:
    name: global-admin
    scope: global
    path:
    desciption:
       Highly priviliged users of a service may only access this service
            with the highest security levels
  subjects:
  - kind: userRole
    name: admin
  requires:
    user:
      trustscore: 0.9
      authentication:
        values: [mfa, 2fa]
```

```
    device:
      trustscore: 1.0
      authentication:
        values: [ipsec, mtls]
        operator: and
      type:
        values: [mobile]
        not: true
      networkagent:
        trustscore: 1.0

#global policy for mobile devices
- kind: policy
  metadata:
    name: global-mobile-device
    scope: global
    path:
    desciption:
      Mobile users may never access services with high privileged user
          rolse
  subjects:
  - kind: deviceType
    name: mobile
  requires:
    user:
      role:
        values: [admin, secretary]
        not: true

#global policy for student users
- kind: policy
  metadata:
    name: global-student
    path:
    scope: global
    description:
       Students only need basic authentication, mtls, and an acceptable
           trust score
    subjects:
    - kind: userRole
      name: student
    requires:
      user:
```

```
      trustscore: 0.8
    device:
      authentication:
        values: [mtls]
```

List. 1: Example policy configuration file.