SECURITY-TOOLBOX-ALS-WEBDIENST-IN-DER-CLOUD

Stevan Medic

27. JUNI 2025
BISON SCHWEIZ AG
Theaterstrasse 15a, 8401 Winterthur
Version: 1.4

Inhalt

Einleitung

Im Rahmen dieses Projekts wurde eine Security Toolbox als Webdienst in der Cloud umgesetzt. Ziel war es, einen öffentlichen Webserver bereitzustellen, auf dem verschiedene Sicherheits-Tools über eine benutzerfreundliche Oberfläche erreichbar sind. Dazu gehört ein Passwortgenerator, ein IP-Check mit Geo-Daten und ein HTTP-Header-Scanner. Die Tools wurden entweder clientseitig mit JavaScript oder serverseitig mit Python (Flask) entwickelt. Die Weboberfläche wurde mit HTML und CSS erstellt, gehostet auf einem eigenen Cloudserver. Abschliessend wurde der Dienst abgesichert und öffentlich zugänglich gemacht – inklusive HTTPS-Verschlüsselung, Input-Validierung und Logging.

Mögliche Risiken zu Beginn der Arbeit

Cloudserver-Konfiguration: Probleme bei der Einrichtung des Servers oder des Webservers (Nginx/Apache) könnten das Projekt verzögern.

Sicherheit: Unsichere Eingaben oder falsch konfigurierte Tools könnten Schwachstellen im System darstellen.

API-Ausfälle: Externe Dienste (z. B. für IP-Geo-Daten) könnten nicht erreichbar sein oder Fehler liefern.

HTTPS-Zertifikate: Die Einrichtung von Let's Encrypt und Certbot könnte scheitern, wenn Domain oder DNS nicht korrekt eingerichtet sind.

Zeitdruck: Da alle Tools entwickelt, integriert und abgesichert werden müssen, ist eine gute Planung entscheidend.

Checkliste

Wichtige Hinweise	Dokumentation ist für Fachpersonen verständlich	o
	Eigenleistung und Unterstützungen sind klar deklariert	o
Allgemein	Kopfzeile Projektname Titel	o
	Fusszeile Datum, Autorin, Seitenzahl	o
	Seitenlayout: Keine Überlappung Seitenränder, Quer/Hochformat	o
	Beschriftung der Bilder/Grafiken	o
	Einsatz von aussagekräftigen Grafiken (Netzwerkplan, DB-Schema)	o
Titelblatt	Klar und übersichtlich gestaltet	o
	Überbegriff: Individuelle Abschlussprojekt BLJ	o
	Projektname (aussagekräftig / max. 20 Zeichen)	o
	Name, Abgabedatum, Name der Lehrfirma	o
	Version des Dokuments	o

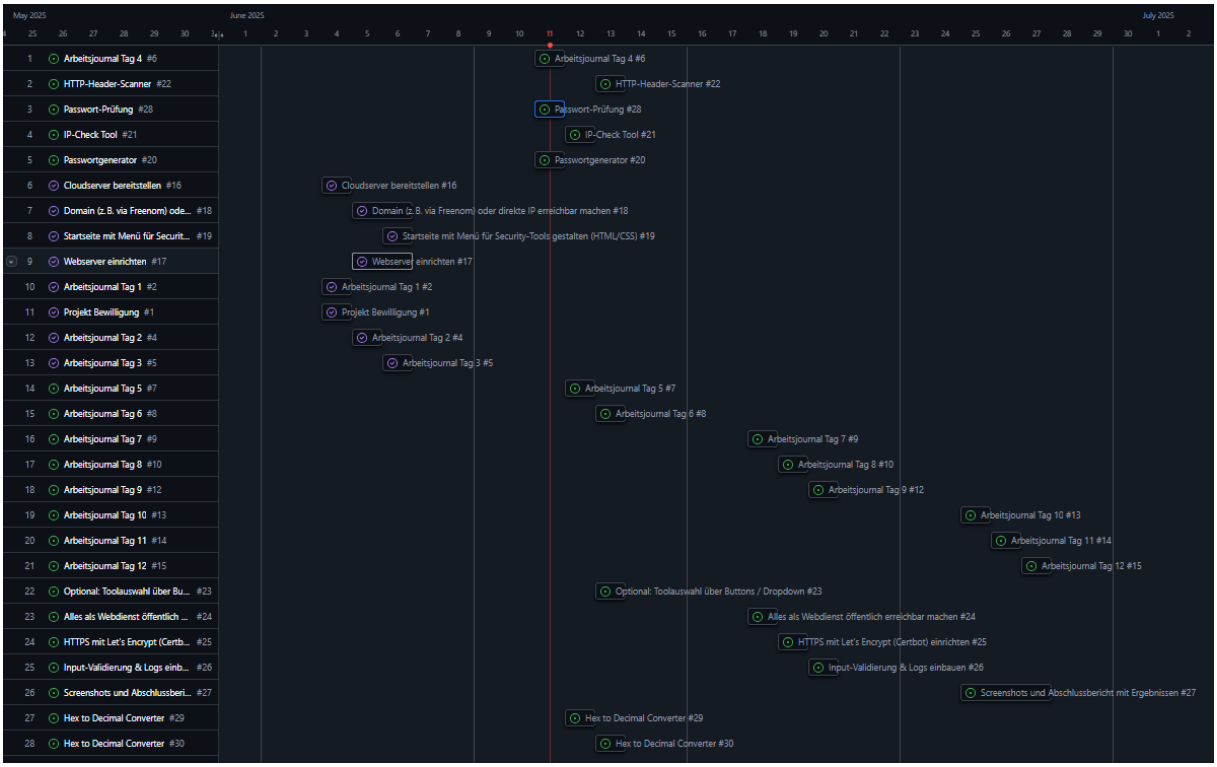
Inhaltsverzeichnis	Kapitel nummeriert (z.B.: 2.1; 2.1.1 usw.)	o
	Seitenzahlen korrekt angegeben	o
	Formatierung überprüft	o
Einleitung	Änderungstabelle/Versionierung (tabellenform)	o
	Aufgabenstellung und Projektbeschreibung	o
	Mögliche Risiken vor Projektbeginn	o
Planung	Terminplan (Gantt Diagramm oder Screenshot GitHub Project) vorhanden	o
	Entscheidungswege und Möglichkeiten (Entscheidungsmatrix)	o
Hauptteil	Detaillierte Beschreibung des Vorgehens und der Zwischenschritte	o
	Ergebnisse der Arbeit	o
	Entscheidungen sind formuliert	o
	Arbeitsjournal vorhanden	o
	Testplan/Testfälle mit Ergebnissen	o
	Persönliches Fazit	o
Anhang	Quellenangaben und Literaturverzeichnis vorhanden	o
	Glossar/Begriffserklärungen vorhanden *(GitHub / Dokument)*	o
	Bildverzeichnis vorhanden	o
	Programm-Code, Scripts, Foto-Dokumentation *(GitHub / Dokument)*	o
	Relevant KI-Chat-Prompts/Auszüge	o
	Testplan/Testfälle (optional)	o
	Ausgefüllte Checkliste	o
Code/Konfigurationen *(Dateien bevorzugt auf GitHub)*	Link zum GitHub Repository vorhanden	o
	Programm-Code folgt Clean-Code Richtlinien	o
	Wichtige Module, Klassen, Funktionen sind kommentiert	o
	Aussagekräftige Namen für Dateien, Klassen, Funktionen, DB-Felder, …	o
	Programm-Dateien Header mit Autor, Datum, Version, Beschreibung	o

Änderungstabelle

Datum	Version	Änderung	Autor
03.06.2025	0.1	Projektidee gewählt, Projektfreigabe erhalten	Stevan Medic
04.06.2025	0.2	Cloudserver vorbereitet, erste Netzwerkeinstellungen	Stevan Medic

Datum	Version	Änderung	Autor
05.06.2025	0.3	Domain-Überlegungen & Startseite begonnen	Stevan Medic
06.06.2025	0.4	Webserver eingerichtet, erste HTML-Struktur	Stevan Medic
09.06.2025	0.5	Passwortgenerator & Passwort-Checker erstellt	Stevan Medic
10.06.2025	0.6	IP-Check Tool + HTTP-Header Scanner (Frontend) fertig	Stevan Medic
11.06.2025	0.7	sizeconverter + Hex-Konverter erstellt	Stevan Medic
12.06.2025	0.8	Dienste als systemd eingerichtet, Backend-Tests abgeschlossen	Stevan Medic
17.06.2025	0.9	HTTPS mit Let's Encrypt versucht, fiel auf selbstsigniertes Zertifikat zurück	Stevan Medic
21.06.2025	1.0	Öffentliche Erreichbarkeit versucht, erste Firewall-Tests mit OPNsense	Stevan Medic
24.06.2025	1.1	Sicherheitsprüfung, Logs + Validierung ergänzt	Stevan Medic
25.06.2025	1.2	Backup-Konzept eingebaut, letzte Tools getestet	Stevan Medic
26.06.2025	1.3	Präsentationsvorbereitung, Dokumentation überarbeitet	Stevan Medic
27.06.2025	1.4	Finale Version: Screenshots, Arbeitsjournal Tag 12, Doku finalisiert	Stevan Medic

Gantt Diagramm



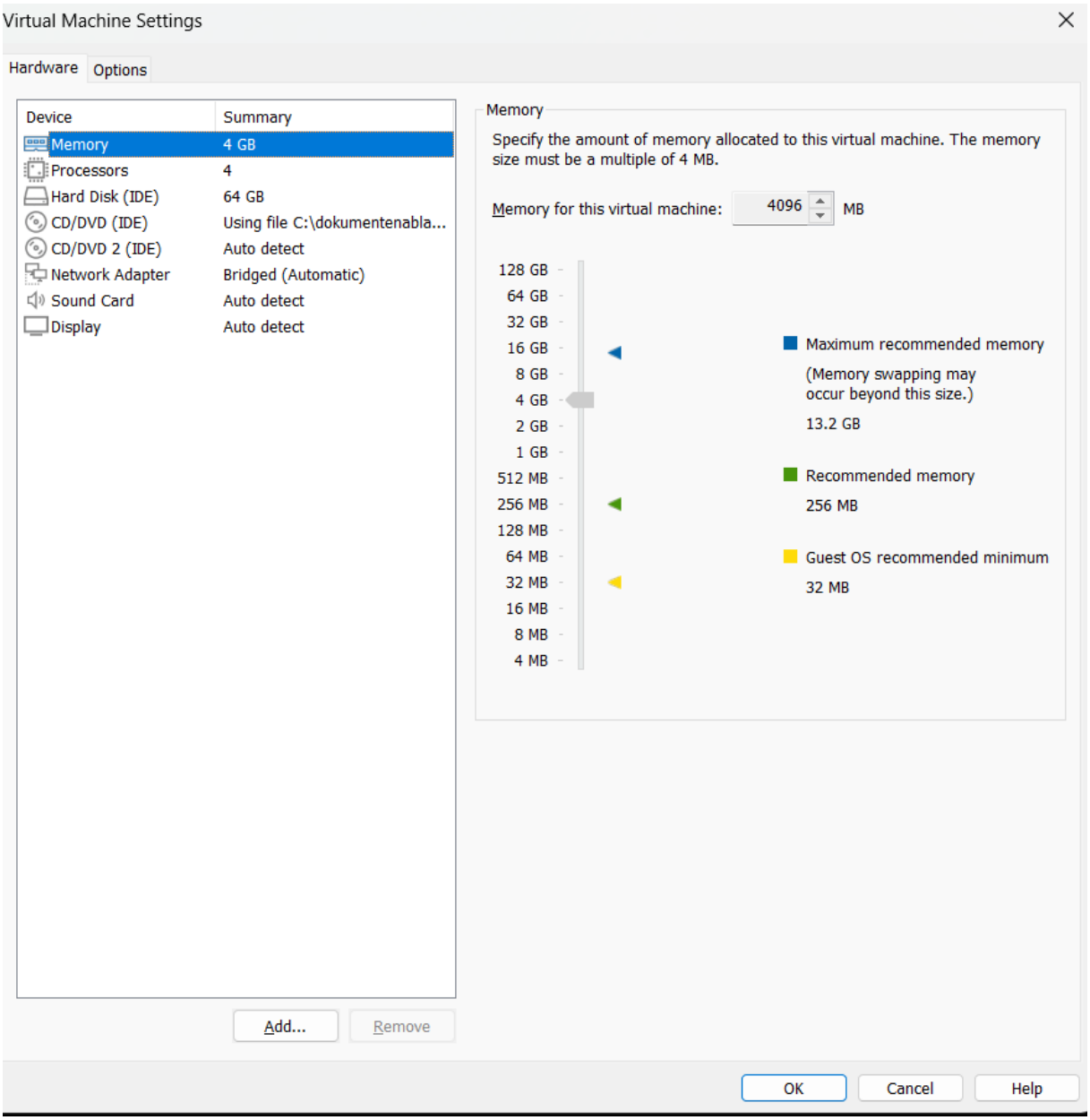
Entscheidungsmatrix

Entscheidungspunkt	Option A: Oracle Cloud (Free Tier)	Option B: Hetzner Cloud (bezahlt)	Option C: Lokal via Proxmox VM
Kosten	Kostenlos (Free Tier)	Niedrig (ab ca. 4 CHF/Monat)	Keine laufenden Kosten
Verfügbarkeit / Erreichbarkeit	Hoch, weltweit erreichbar	Hoch, weltweit erreichbar	Nur lokal oder mit Portweiterleitung
Performance / Ressourcen	Eingeschränkt (1vCPU, 1GB RAM)	Skalierbar, je nach Tarif	Abhängig von PC
Einrichtung / Aufwand	Etwas komplexer (Cloud-Zugang, Keys)	Mittel (Webinterface, SSH)	Einfach (man kontrolliert alles lokal)
Domain-Anbindung (Freenom)	Sehr gut möglich (IPv4 erreichbar)	Sehr gut möglich	Nur mit DynDNS oder Portforwarding
Latenz	Sehr niedrig	Sehr niedrig	Sehr niedrig (lokal)

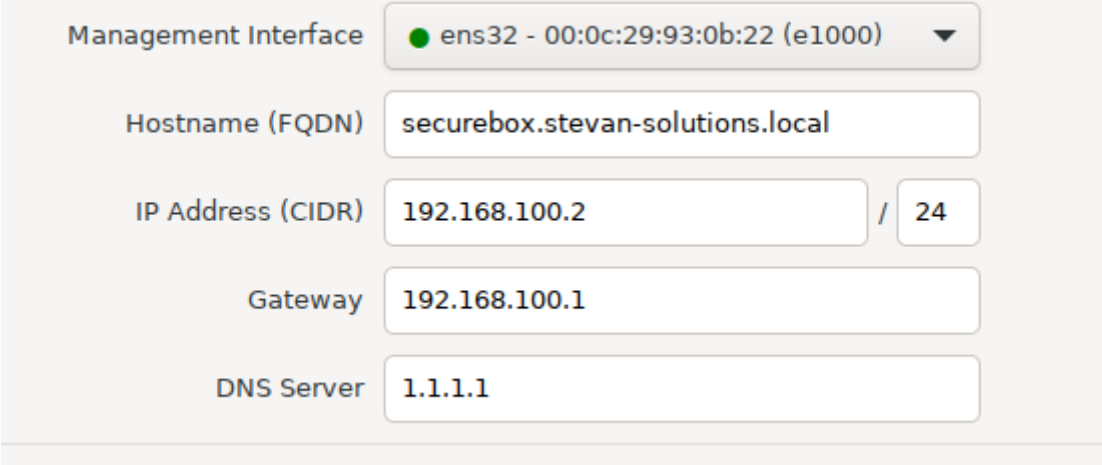
Entscheidungspunkt	Option A: Oracle Cloud (Free Tier)	Option B: Hetzner Cloud (bezahlt)	Option C: Lokal via Proxmox VM
Zugang von aussen	Standardmässig möglich	Standardmässig möglich	Nur mit Netzwerkkonfig / Firewallregel
Erfahrung für Zukunftsprojekte	Cloud-Knowhow (gut für Portfolio)	Cloud-Knowhow	Infrastruktur-Knowhow (Proxmox Skills)
Sicherheit	Sehr gut, wenn Firewall & SSH hart konfig.	Gut, abhängig vom Setup	Muss lokal gut geschützt werden

Server bereitstellen

Server Installieren



Server statische IP setzen

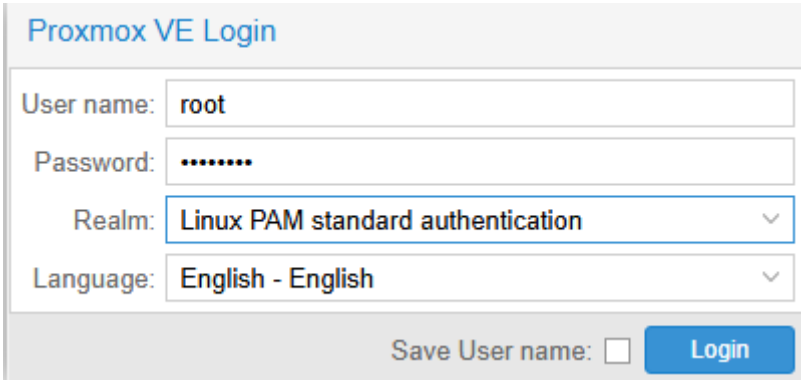


Die Statische IP konnte ich ganz einfach in der Konfiguration geben.

Option	Value
Filesystem:	ext4
Disk(s):	/dev/sda
Country:	Switzerland
Timezone:	Europe/Zurich
Keymap:	de-ch
Email:	stevanmedic32@gmail.com
Management Interface:	ens32
Hostname:	securebox
IP CIDR:	192.168.100.2/24
Gateway:	192.168.100.1
DNS:	1.1.1.1

Proxmox Web interface verbinden

https://192.168.100.2:6000 in einen Browser eingeben und das dann ausfüllen.



Das Passwort ist, das das man am Anfang in der Konfiguration eingegeben hatte.

Webserver einrichten

Der Server wurde als LXC-Container unter Proxmox eingerichtet. Dabei kam das Ubuntu 24.04 Live Server ISO zum Einsatz. Nach erfolgreicher Netzwerkkonfiguration (IP-Adresse, Gateway, DNS) konnte über die Konsole sowie über das Proxmox-Webinterface auf das System zugegriffen werden. Apache2 wurde als Webserver installiert und für die Bereitstellung der HTML-Oberfläche konfiguriert.

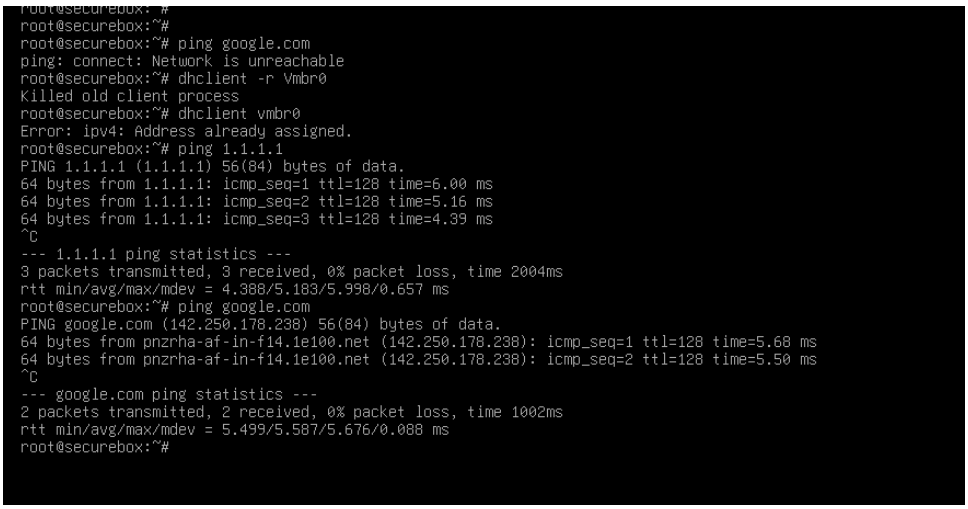


Abbildung: Erfolgreiche Verbindung nach Netzwerksetup

Startseite mit Toolmenü

Die HTML-Startseite listet alle verfügbaren Tools übersichtlich auf. Diese befindet sich unter /var/www/html/index.html und enthält direkte Verlinkungen auf alle Tools im Unterordner /tools/.

Tool-Übersicht

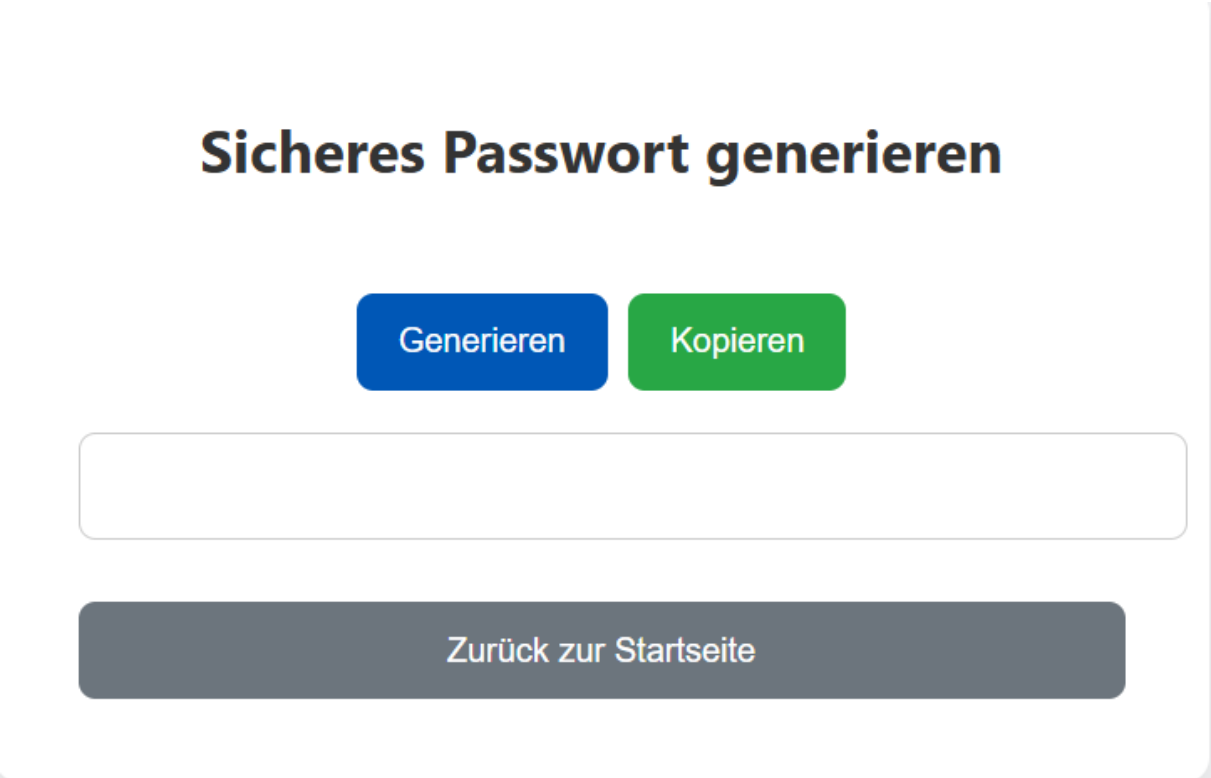
Passwortgenerator

Ein einfaches HTML/JS-Tool zur Generierung sicherer Zufallspasswörter. Die Erzeugung erfolgt rein im Browser, ohne Serververbindung. Der Passwortgenerator wurde in HTML und JavaScript umgesetzt und läuft vollständig im Browser des Nutzers. Es handelt sich dabei um eine rein clientseitige Lösung, das bedeutet: Es werden keine Daten an einen Server gesendet oder gespeichert. Sobald die Seite aufgerufen wird, kann der Benutzer per Klick ein zufälliges Passwort generieren lassen.

Im Hintergrund enthält der JavaScript-Code eine vordefinierte Zeichentabelle. Diese besteht aus Grossbuchstaben, Kleinbuchstaben, Zahlen sowie verschiedenen Sonderzeichen. Beim Klick auf den Button zur Passwortgenerierung wird eine Schleife ausgeführt, die nacheinander zufällig Zeichen aus dieser Tabelle auswählt. Die Anzahl der Wiederholungen (zum Beispiel 12) bestimmt dabei die Länge des finalen Passworts.

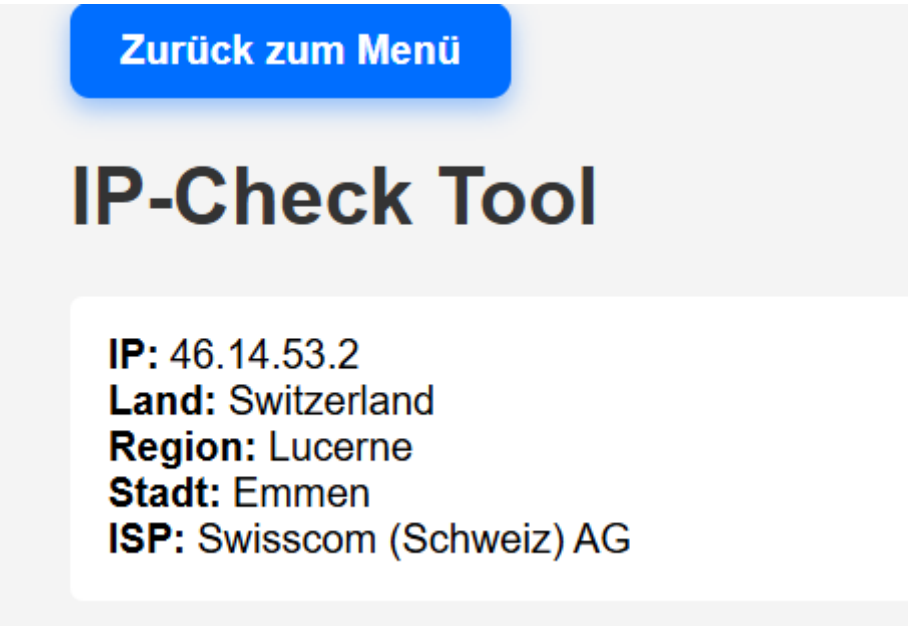
Für jede Stelle im Passwort wird ein Zeichen per Zufallsfunktion (Math.random()) ausgewählt und zum Ergebnis hinzugefügt. Das fertige Passwort wird anschliessFend direkt in einem Eingabefeld angezeigt, sodass der Benutzer es einfach kopieren kann. Da alles im Browser passiert, ist der Generator nicht nur schnell, sondern auch datenschutzfreundlich.

Diese Umsetzung eignet sich besonders gut für eine Toolbox-Webseite, da sie ohne zusätzliche Serverkomponenten funktioniert und sofort einsetzbar ist. Der Generator kann beliebig erweitert werden – zum Beispiel mit Optionen für die Passwortlänge oder zum Ein-/Ausschalten bestimmter Zeichengruppen.



IP-Check Tool

Das IP-Check Tool zeigt dem Benutzer seine öffentliche IP-Adresse und Standortinfos wie Stadt, Region und Provider an. Dafür wird im Hintergrund eine externe API (ip-api.com) aufgerufen – aber nicht vom Server, sondern direkt vom Browser des Besuchers. So bekommt jeder Benutzer genau seine eigene IP angezeigt. Das Tool funktioniert vollständig ohne backend.



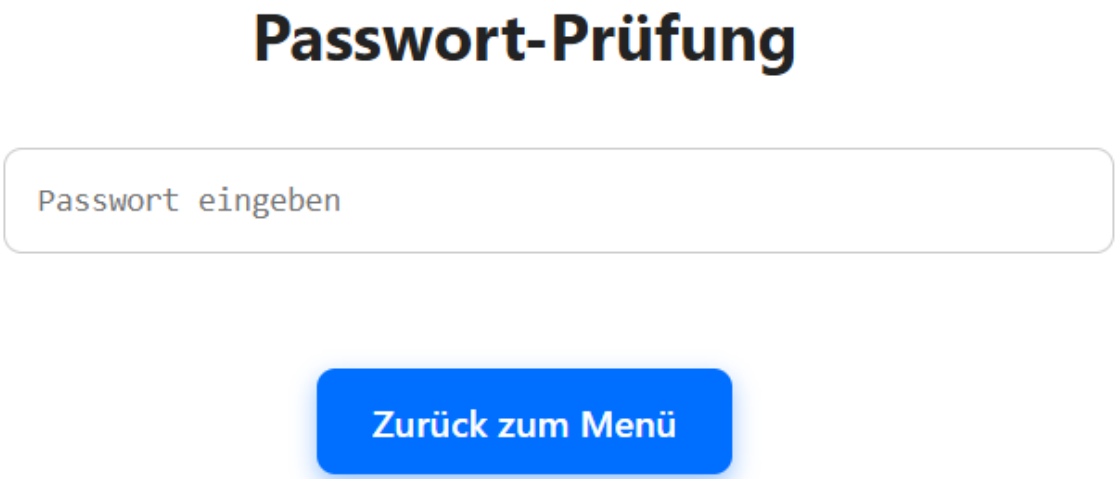
HTTP-Header-Scanner

Der HTTP-Header-Scanner ist ein kleines Python-Backend mit Flask. Die Benutzeroberfläche besteht aus einem HTML-Formular, in das man eine URL eingibt. Diese wird dann per POST an das Flask-Backend gesendet. Dort wird die URL mit requests oder curl aufgerufen, und die HTTP-Header werden analysiert. Das Ergebnis wird zurückgegeben und im Browser angezeigt. Der Dienst läuft als systemd-Dienst und startet automatisch.

Passwort-Stärke-Prüfer

Der Passwort-Checker bewertet die Sicherheit von eingegebenen Passwörtern. Geprüft wird, ob Gross- und Kleinbuchstaben, Zahlen, Sonderzeichen enthalten sind und ob das Passwort lang genug ist. Je nach Score zeigt das Tool visuell an, ob das Passwort schwach, mittel oder stark ist. Auch dieses Tool basiert rein auf JavaScript und ist sofort einsatzbereit ohne Server.Hash-Generator

Ein Client-seitiges Tool zur Berechnung von SHA-256, SHA-512 und MD5-Hashes mithilfe von JavaScript. Es benötigt keine Serververbindung und funktioniert offline.



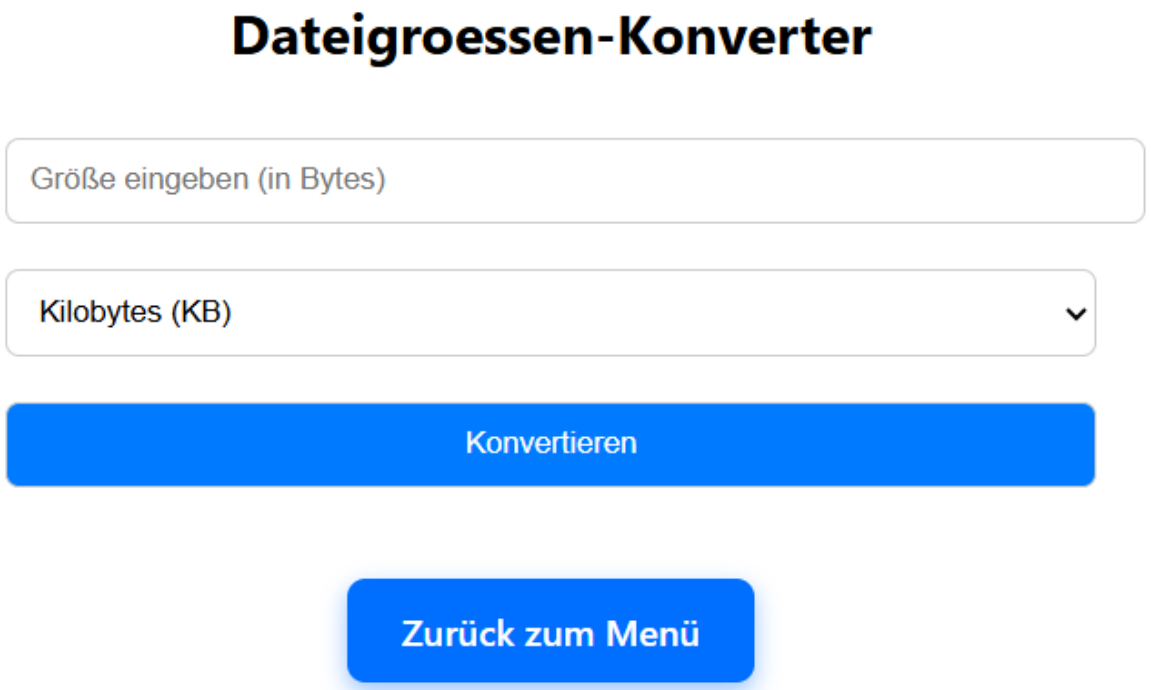
Hash-Generator

Der Hash-Generator erstellt aus einem eingegebenen Text einen Hashwert – zum Beispiel mit SHA-256, SHA-512 oder MD5. Dabei wird JavaScript im Browser genutzt. Der Benutzer gibt den Text ein, wählt das Hash-Verfahren, und das Tool zeigt den

erzeugten Hash an. Es wird keine Verbindung zum Server gebraucht – alles läuft offline und sicher.

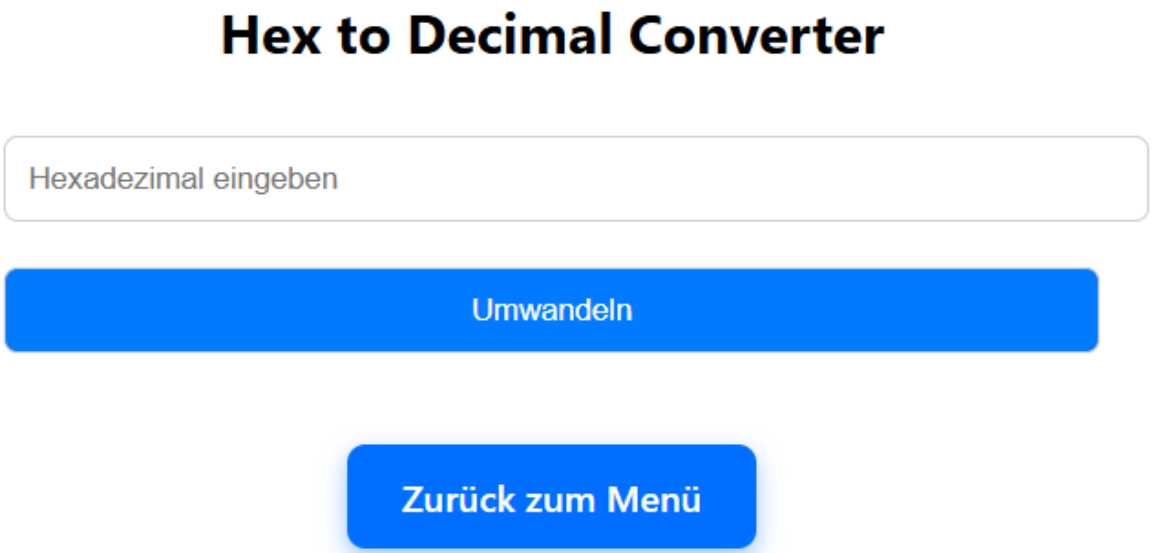
Dateigrössen-Konverter

Der Dateigrössen-Konverter ist ein einfaches Tool, das es ermöglicht, verschiedene Dateigrössen zu konvertieren. Wenn man zum Beispiel eine Datei in Bytes misst, kann es schwer sein, sich die Grösse in anderen Einheiten wie Kilobytes (KB), Megabytes (MB) oder Gigabytes (GB) vorzustellen. Genau hier kommt der Konverter ins Spiel. Mit nur wenigen Eingaben kann man die Dateigrösse in die gewünschte Einheit umwandeln und so schnell feststellen, wie gross eine Datei wirklich ist.



Hexa-Konverter

Der Hexadezimal-zu-Dezimal-Konverter ist besonders nützlich für Programmierer und Techniker, die häufig mit unterschiedlichen Zahlensystemen arbeiten müssen. Hexadezimale Zahlen werden oft in der Programmierung verwendet, beispielsweise bei der Arbeit mit Speicheradressen oder Farbcodes. Der Converter ermöglicht es, eine hexadezimale Zahl in eine dezimale Zahl umzuwandeln, um sie besser zu verstehen oder weiterzuverarbeiten.



Dauerhafte Dienste

Die Flask-basierten Tools (IP-Check und Header-Scanner) wurden als systemd-Dienste eingerichtet. Dadurch starten sie automatisch beim Containerstart und können zuverlässig über Apachen weitergeleitet werden.

Verzeichnisstruktur

```
root@badgebox:~# tree /var/www/html/
/var/www/html/
|-- index.html
|-- style.css
`-- tools
    |-- datei.html
    |-- hashgen.html
    |-- headerscan.html
    |-- hexconverter.html
    |-- ipcheck.html
    |-- password.html
    |-- passwordcheck.html
    |-- sizeconverter.html
    `-- style.css
```

Selbstsigniertes Zertifikat

Ich habe versucht, ein Let's Encrypt-Zertifikat für die HTTPS-Verbindung zu verwenden, aber es hat nicht funktioniert. Der Versuch, das Zertifikat über Certbot zu bekommen, ist

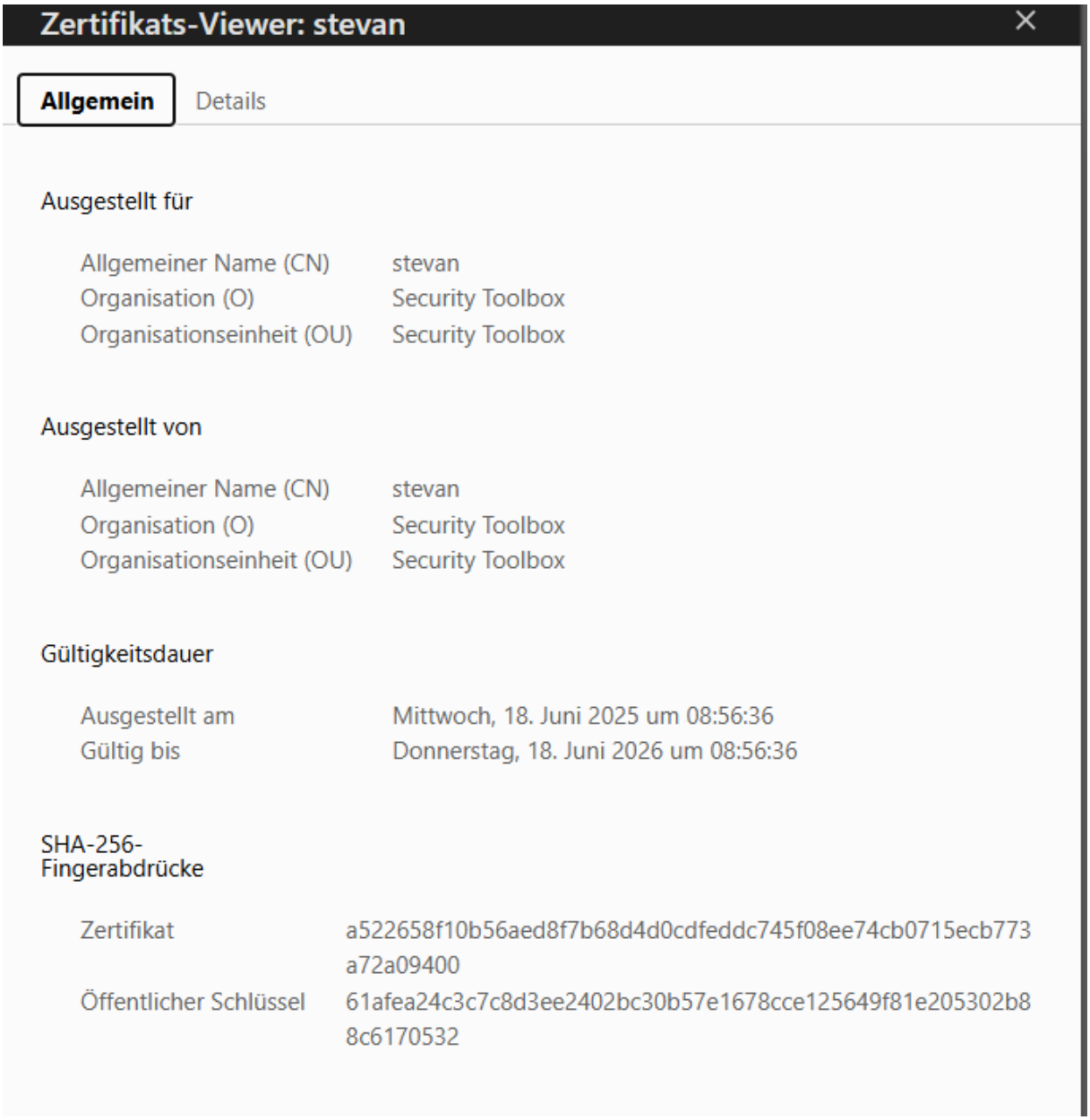
fehlgeschlagen, weil es Probleme bei der Domain-Validierung gab. Der Server konnte das Zertifikat nicht von Let's Encrypt herunterladen. Das könnte daran liegen, dass die öffentliche IP-Adresse nicht korrekt zugewiesen oder die Firewall-Einstellungen nicht richtig konfiguriert waren.

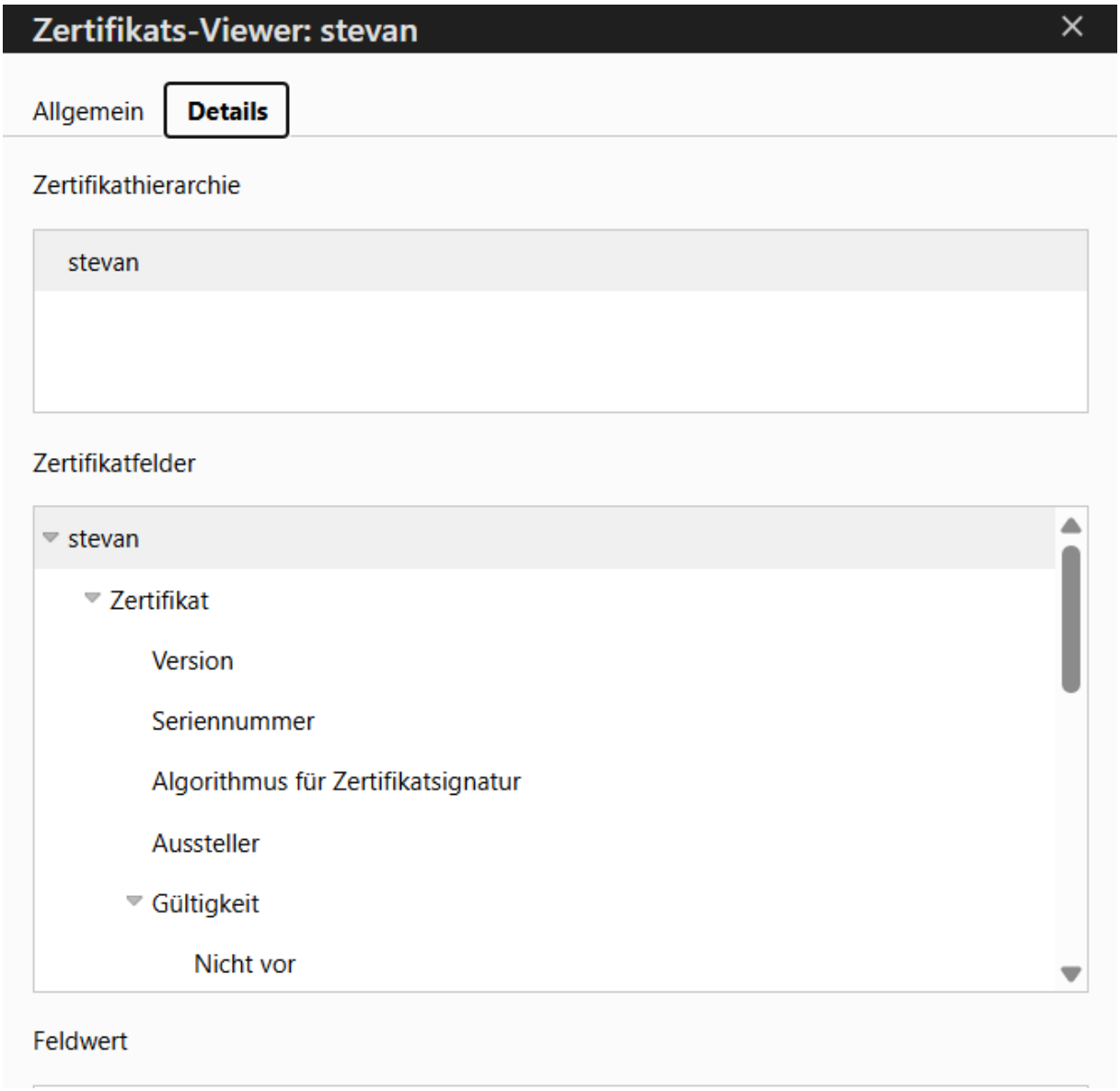
Da das mit Let's Encrypt nicht funktioniert hat, habe ich ein selbstsigniertes Zertifikat erstellt, um HTTPS zumindest für den internen Gebrauch zu aktivieren. Dadurch wird die Verbindung verschlüsselt, aber im Browser wird eine Sicherheitswarnung angezeigt, weil das Zertifikat nicht von einer vertrauenswürdigen Zertifizierungsstelle kommt. Man kann diese Warnung zwar umgehen, aber es ist trotzdem kein offizielles Zertifikat.

Das grösste Problem war aber, dass die Webseite weiterhin nicht von aussen erreichbar ist. Ich konnte die Seite über die öffentliche IP-Adresse nicht aufrufen. Die Portweiterleitung in der Firewall oder dem Router war nicht richtig eingerichtet, was dazu führte, dass die Verbindung blockiert wurde. Auch eine externe Verbindung über die IP-Adresse zeigte entweder eine Fehlermeldung oder eine nicht erreichbare Seite.

Also ist der Webserver trotz HTTPS-Konfiguration noch nicht für externe Nutzer zugänglich. Um die Seite wirklich öffentlich zu machen, müssen die Portweiterleitungen und Firewall-Einstellungen angepasst werden, damit die Ports 80 (HTTP) und 443 (HTTPS) von aussen erreichbar sind.

```
root@badgebox:~# sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
 -keyout /etc/ssl/private/selfsigned.key \
 -out /etc/ssl/certs/selfsigned.crt
....+.....+.......++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
......+...+.+..............+.......++++++++++++++++++++++++++++++++++++++++++++
..+.+...+.....+.....................+.........+....+..+................+......+.
.+........++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*...
...++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CH
State or Province Name (full name) [Some-State]:Zürich
Locality Name (eg, city) []:Zürich
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Security Toolbox
Organizational Unit Name (eg, section) []:Security Toolbox
Common Name (e.g. server FQDN or YOUR name) []:stevan
Email Address []:stevanmedic32@gmail.com
```


Zertifikats-Viewer: stevan
Allgemein
Details
Ausgestellt für
Allgemeiner Name (CN) stevan
Organisation (O) Security Toolbox
Organisationseinheit (OU) Security Toolbox
Ausgestellt von
Allgemeiner Name (CN) stevan
Organisation (O) Security Toolbox
Organisationseinheit (OU) Security Toolbox
Gültigkeitsdauer
Ausgestellt am Mittwoch, 18. Juni 2025 um 08:56:36
Gültig bis Donnerstag, 18. Juni 2026 um 08:56:36
SHA-256-Fingerabdrücke
Zertifikat a522658f10b56aed8f7b68d4d0cdfeddc745f08ee74cb0715ecb773a72a09400
Öffentlicher Schlüssel 61afea24c3c7c8d3ee2402bc30b57e1678cce125649f81e205302b88c6170532



Fazit

Ganz ehrlich Ich hätte am Anfang nicht gedacht, dass ich dieses Projekt wirklich so weit bringen würde. Ich war mir unsicher, ob das Thema für mich passt, und hatte ursprünglich sogar ein ganz anderes Projekt im Kopf. Aber nachdem ich mit ChatGPT ein paar Ideen gesammelt hatte, kam ich auf die Idee mit der Security Toolbox – und das hat mich direkt gepackt. Ich wollte etwas machen, das mit Sicherheit, Webentwicklung und Servern zu tun hat, etwas, das man wirklich brauchen kann. Und genau das hab ich mir zugetraut.

Der Start war aber alles andere als einfach. Ich hatte direkt Probleme mit dem Netzwerk, die VM hatte zwar eine statische IP, kam aber nicht ins Internet. Erst später habe ich

gemerkt, dass ich eigentlich mit Proxmox arbeiten wollte, aber komplett woanders angefangen habe.

Im Verlauf des Projekts hatte ich mit vielen technischen Problemen zu kämpfen. Proxmox hat nicht so funktioniert, wie ich wollte. Die Netzwerkkonfiguration hat mich teilweise komplett durcheinandergebracht. Ich habe versucht, Let's Encrypt für ein offizielles SSL-Zertifikat zu verwenden, aber das hat wegen der Portweiterleitungen und DNS-Einstellungen nicht geklappt. Trotzdem hab ich eine Alternative gefunden und ein selbstsigniertes Zertifikat erstellt, damit zumindest lokal eine verschlüsselte Verbindung möglich war.

Was ich in diesen zwölf Tagen gelernt habe, ist schwer in Worte zu fassen. Ich habe nicht nur technisch unglaublich viel mitgenommen – vom Aufsetzen eines Apache-Servers über das Einrichten von Flask-Backends und systemd-Diensten bis hin zu JavaScript-Tools, die komplett im Browser laufen – sondern ich habe vor allem gelernt, wie wichtig Geduld, Eigeninitiative und Durchhaltevermögen sind. Ich habe mein Projekt selbständig geplant, strukturiert und durchgeführt. Ich habe Fehler gemacht, aber aus jedem einzelnen etwas gelernt.

Auch wenn mein Webdienst am Schluss nicht von aussen erreichbar war, funktioniert alles lokal einwandfrei. Die Tools laufen stabil, die Oberfläche ist übersichtlich und das System ist so aufgebaut, dass es jederzeit erweitert werden kann. Ich bin mit dem Ergebnis mehr als zufrieden. Nicht, weil alles perfekt war – sondern weil ich es trotz aller Probleme geschafft habe, mein Ziel zu erreichen und etwas Eigenes auf die Beine zu stellen.

Dieses Projekt hat mir gezeigt, dass ich in der Lage bin, ein komplettes System von null an aufzubauen – mit allen Höhen und Tiefen, die dazu gehören. Und genau deshalb bin ich stolz auf das, was ich gemacht habe.

Probleme

Problem 1

Heute habe ich weiter an meinem Projekt gearbeitet und bin bei der Einrichtung von Proxmox auf ein paar Probleme gestossen. Ich habe Proxmox in einer VM unter VMware Workstation installiert und wollte eigentlich direkt mit dem Webinterface weiterarbeiten. Ich habe ihm eine feste IP-Adresse gegeben, aber danach konnte ich weder das Webinterface aufrufen noch ins Internet pingen.

Ich habe versucht, mit dem Browser auf die Adresse https://192.168.100.2:8006 zuzugreifen, aber es kam nichts. Ich war mir sicher, dass mein Host im gleichen Subnetz war, also dachte ich zuerst, dass vielleicht etwas mit dem Netzwerkadapter nicht stimmt. Dann habe ich gesehen, dass das Interface vmbr0 zwar eine IP hatte, aber als DOWN angezeigt wurde. Auch das physische Interface ens32 war im Zustand NO-CARRIER, also nicht verbunden.

Ich habe dann mit ChatGPT zusammen Schritt für Schritt das Problem analysiert. Erst habe ich versucht es mit einer Bridge zu lösen, aber das hat nicht funktioniert, weil wahrscheinlich meine Netzwerkkarte in Windows nicht richtig mit VMware gebridget war. Dann habe ich es mit NAT probiert.

Ich habe die VM auf NAT gestellt und in Proxmox die Netzwerkkonfiguration so angepasst, dass vmbr0 per DHCP eine IP bekommt. Ich habe auch manuell den DNS gesetzt auf 1.1.1.1 und danach die Netzwerkschnittstellen neu geladen. Nachdem ich das alles gemacht habe, hat Proxmox endlich eine IP aus dem NAT-Bereich bekommen und ich konnte ins Internet pingen. Auch das Webinterface war wieder erreichbar.

Damit war das Problem gelöst und ich konnte weitermachen mit dem Herunterladen vom Ubuntu Template und dem Einrichten meiner Umgebung für die Toolbox.

Problem 2

Heute habe ich weiter an meiner Security Toolbox gearbeitet und wollte in Proxmox eine Ubuntu-VM erstellen, um meinen Webdienst darauf aufzusetzen. Ich hatte das Ubuntu-ISO bereits hochgeladen und bin die Einstellungen für die neue VM wie geplant durchgegangen. Doch als ich die VM starten wollte, kam direkt eine Fehlermeldung: „KVM virtualization configured, but not available**“**.

Zuerst dachte ich, es liegt an einer falschen Einstellung, aber nach ein paar Tests wurde klar, dass das Problem tiefer liegt. Proxmox selbst lief ja in einer VM unter VMware Workstation, und um darin wieder eigene VMs mit KVM zu starten, muss sogenannte Nested Virtualization aktiv sein – also verschachtelte Virtualisierung.

Ich habe dann im BIOS nachgesehen und gesehen, dass Virtualisierung bereits aktiviert war. Auch in den VMware-Einstellungen hatte ich den Haken für „Virtualize Intel VT-x/EPT“ gesetzt. Trotzdem liess sich die Ubuntu-VM nicht starten und es kam immer wieder dieselbe Fehlermeldung.

Zusammen mit ChatGPT habe ich dann alle möglichen Ursachen durchgecheckt. Ich habe in Windows alle Funktionen deaktiviert, die Virtualisierung blockieren konnten, also Hyper-V, Virtual Machine Platform und Windows Hypervisor Platform. Auch Secure Boot habe ich geprüft. Am Ende stellte sich aber heraus, dass meine VMware-Umgebung trotz korrekter BIOS-Einstellungen und deaktivierter Windows-Features die

Weitergabe von VT-x einfach nicht erlaubt – wahrscheinlich durch Einschränkungen auf Firmware-Ebene oder wegen der Kombination aus Host, VMware und Proxmox.

Da ich an dem Punkt nicht weiterkam, habe ich mich entschieden, auf Proxmox zu verzichten und stattdessen direkt eine Ubuntu-VM in VMware Workstation zu erstellen. Das reicht für mein Projekt vollkommen aus – ich brauche keinen KVM-Zugriff für Apache, HTML oder Python. Damit konnte ich dann direkt weitermachen und meinen Webserver aufsetzen. Ich werde dann später probieren die Ubuntu VM in die Proxmox Umgebung einzufügen.

Problem 3

Das Hauptproblem, das ich beim Einrichten des Let's Encrypt-Zertifikats hatte, war, dass der Certbot den Domain-Validierungsprozess nicht erfolgreich abschliessen konnte. Dies führte dazu, dass das SSL-Zertifikat von Let's Encrypt nicht heruntergeladen werden konnte. Der Fehler trat vermutlich aufgrund einer fehlerhaften Konfiguration der DNS-Einträge oder Portweiterleitungen auf, was dazu führte, dass der Webserver nicht ordnungsgemäss mit der Domain verknüpft werden konnte.

Zusätzlich hatte ich auch Schwierigkeiten, den Webserver von aussen zugänglich zu machen. Auch wenn die HTTPS-Konfiguration auf dem Server richtig gesetzt war, war der Zugriff über die öffentliche IP-Adresse aufgrund fehlender Portweiterleitungen und nicht korrekt eingerichteter Firewall-Regeln nicht möglich. Das führte dazu, dass der Server nicht extern erreichbar war, und die Seite nicht im Internet aufgerufen werden konnte.

Arbeitsjournal

Tag 1

Heute habe ich mein Gedanken zu meinem Projekt gemacht. Ich wollte zuerst ein anderes Thema wählen, aber ich dachte das das für mich ein bisschen zu viel wäre. Ich habe meine Ideen suche mit ChatGPT gemacht mit dem Prompt, der uns zur Verfügung gestellt wurde. Nach dem ich auf ein Thema gekommen bin, das mir gefällt habe, ich angefangen mit dem GitHub Repo und Projekt und habe dann meine Meilensteine festgelegt und dann die Bestätigung von Jörg bekommen zum Anfangen. Ich habe mit der Erstellung von der VM angefangen mit auf der dann alles laufen wird. Ich mein Problem bis jetzt ist aber das ich auf der VM die statische IP setzten konnte aber ich konnte nachdem nichtmehr raus ins Internet kommen ich weiss nicht, was das Problem sein kann.

Tag 2

Direkt am Anfang vom Tag habe ich gemerkt das ich immer noch das Problem habe, das ich nicht raus ins Internet komme mit meinem Server. Ich weiss nicht an was es liegen kann ich habe jetzt auch mit ChatGPT geschaut an was en liegen kann. Ich habe gemerkt das ich Komplet falsch angefangen habe. Ich musste es auf Proxmox machen ich habe es auf Linux gemacht :). Ich konnte jetzt Proxmox richtig installieren und dann von einer weiteren VM darauf zugreifen. Ich habe jetzt aber ein kleines Problem, das ich auf die Webseite zugreifen kann, aber ich nicht den Container für den Webserver erstellen kann, weil ich von der Proxmox nicht raus ins Internet komme.

Tag 3

Heute Morgen habe ich direkt das Problem, was ich hatte flicken können. Danach war es aber nicht besser ich musste «Virtualize Intel VT-x/EPT» auf meinem Proxmox Processors einschalten, aber es ging nicht. Ich ging dann in meinen eigenen BIOS-Einstellungen um «Virtualization Technology (VT-x)», «VT-d» zu aktivieren. Es war schon aktiviert also ging ich in meine Windows Features Hyper-V, Virtual Machine Platform und Windows Hypervisor Platform zu deaktivieren in der Hoffnung, dass das hilft. Es funktionierte Trotzdem nicht. Ich habe mir dann umentschieden ich mach den Webserver nicht in Proxmox direkt, sondern einfach in einer neuen VM.

Tag 4

Heute habe ich gestartet mit der Erstellung der Tools von meiner Webseite Ich werde zuerst, denn Passwortgenerator erstellen und ich weiss nicht wie lange das Gehen wird also weiss ich nicht, ob ich heute dann einen anderes noch machen kann. Ich bin aber sehr gut in der Zeit. Ich habe jetzt die Seite erstellt für den Passwortgenerator und er funktioniert auch. Man kann sich per Knopfdruck das Passwort generieren und auch per Knopfdruck kopieren. Ich konnte auch mein nächstes Tool fertigstellen. Mit dem kann man testen, wie stark sein Passwort ist.

Tag 5

Heute habe ich mit dem IP-Check Tool angefangen. Zuerst habe ich die API eingebunden, die die öffentliche IP-Adresse des Nutzers abruft und die zugehörigen geographischen Daten wie Land, Region und Stadt anzeigt. Nachdem ich die grundlegende HTML-Struktur erstellt hatte, konnte ich das Tool schnell zum Laufen bringen. Es zeigte die IP und die entsprechenden geographischen Daten korrekt an. Als nächstes habe ich den HTTP Header Scanner umgesetzt. Ich habe eine Benutzeroberfläche erstellt, die es ermöglicht, eine URL einzugeben und die HTTP-Header der Seite anzuzeigen. Dafür habe ich JavaScript genutzt, um die Header-Daten abzurufen und darzustellen. Beide Tools liefen einwandfrei und wurden auf meiner Webseite integriert.

Tag 6

Heute habe ich mit dem Dateigrössen-Konverter Tool begonnen. Zuerst habe ich das grundlegende Layout erstellt, bei dem der Benutzer eine Zahl in Bytes eingeben kann. Danach konnte der Benutzer die gewünschte Einheit wählen (KB, MB oder GB), und das Tool berechnet die Umrechnung und zeigt das Ergebnis an. Nachdem ich die JavaScript-Logik implementiert hatte, um die Umrechnung korrekt durchzuführen, konnte ich das Tool erfolgreich testen und in die Webseite integrieren.

Anschliessend habe ich mich dem Hexadezimal zu Dezimal Konverter gewidmet. Hier habe ich eine einfache Eingabemaske für den Benutzer erstellt, bei der er eine hexadezimale Zahl eingeben kann. Die Umwandlung in die dezimale Zahl habe ich mit der parseInt() Funktion erledigt, die die hexadezimale Zahl automatisch in eine Dezimalzahl umwandelt. Auch dieses Tool konnte ich problemlos testen und es funktioniert einwandfrei.

Beide Tools wurden erfolgreich erstellt und in das Projekt integriert. Sie ermöglichen es den Benutzern, schnell und einfach Dateigrössen umzurechnen und Hexadezimalzahlen in Dezimalzahlen umzuwandeln, was für meine Webseite von grossem Nutzen ist.

Tag 7

Heute habe ich mich mit der Erstellung und Einrichtung des selbstsignierten SSL-Zertifikats beschäftigt. Ich habe das Zertifikat auf dem Webserver erstellt und erfolgreich in die Apache-Konfiguration integriert. Dadurch konnte ich die Webseite nun auch über HTTPS anstelle von nur HTTP erreichen, allerdings wird im Browser eine Warnung angezeigt, da es sich um ein selbstsigniertes Zertifikat handelt.

Zusätzlich habe ich meine Dokumentation erweitert und bearbeitet, um den aktuellen Stand der Implementierung sowie die Schritte zur Konfiguration des Zertifikats und der Sicherheitsmassnahmen festzuhalten. Die Dokumentation wurde dabei gründlich aktualisiert, um alle wichtigen Schritte klar zu erläutern.

Ich habe auch mein GitHub-Project auf den neuesten Stand gebracht. Dadurch bleibt das Project immer auf dem neuesten Stand und alle Änderungen sind nachvollziehbar.

Tag 8

Heute habe ich versucht, meine Webseite öffentlich zugänglich zu machen. Die Seite läuft lokal auf dem Server einwandfrei, aber sobald ich versuche, von aussen über die öffentliche IP darauf zuzugreifen, funktioniert es nicht. Ich habe alle Konfigurationen nochmals überprüft und auch die Firewall-Regeln in der Proxmox-Verwaltung kontrolliert und angepasst. Trotzdem bleibt der Zugriff von extern blockiert. Woran genau es liegt,

weiss ich noch nicht. Morgen werde ich weiter an dem Problem arbeiten und versuchen herauszufinden, ob es an der Netzwerkkonfiguration oder doch an einer versteckten Firewall-Regel liegt. Ziel ist es, die Seite endlich öffentlich erreichbar zu machen.

Tag 9

habe ich mich damit beschäftigt, die Firewall-Einstellungen so zu konfigurieren, dass meine Webseite von aussen erreichbar ist. Ich habe zuerst direkt auf dem Server geschaut, ob irgendwelche Regeln blockieren, und dann versucht, das Ganze über die Proxmox-Firewall zu steuern. Ich habe dort entsprechende Regeln angepasst und freigegeben, aber leider hat das nicht den gewünschten Effekt gebracht – der Zugriff von aussen funktionierte immer noch nicht.

Da ich mit der Proxmox-Firewall nicht weiterkam, habe ich mich entschieden, eine eigene OPNsense-Firewall aufzusetzen und zu konfigurieren. Ich habe die grundlegende Einrichtung gemacht und erste Regeln definiert, aber komplett fertig bin ich mit der Konfiguration noch nicht geworden. Das werde ich in den nächsten Tagen fertigstellen, damit der Zugriff endlich richtig funktioniert.

Tag 10

Heute habe ich begonnen, die Abschlussdokumentation zu überarbeiten und zu vervollständigen. Ich habe das Inhaltsverzeichnis aktualisiert, die Formatierung kontrolliert und einige Abschnitte verständlicher formuliert. Besonders bei den Problembeschreibungen und beim Fazit habe ich nochmals auf eine klare Ausdrucksweise geachtet. Zudem habe ich erste Ideen für die Präsentation gesammelt und grob die Folienstruktur geplant

Tag 11

Den heutigen Tag habe ich vollständig der Präsentation gewidmet. Ich habe die Folien mit den wichtigsten Punkten aus dem Projekt befüllt – Ziele, Umsetzung, Herausforderungen und Tools. Dabei habe ich besonders darauf geachtet, dass alles kurz, aber verständlich erklärt wird. Auch Screenshots und visuelle Elemente habe ich eingebaut. Am Ende habe ich die Präsentation getestet, damit sie gut verständlich und flüssig vorgetragen werden kann.

Tag 12

Am letzten Tag habe ich sowohl die Präsentation finalisiert als auch die Dokumentation nochmals überprüft und als PDF exportiert. Ich habe Kleinigkeiten wie Seitenzahlen, Bildunterschriften und das Titelblatt kontrolliert und korrigiert. Danach habe ich die

Abgabe vorbereitet, alle Dateien organisiert und sichergestellt, dass die finale Version vollständig und korrekt ist. Damit ist mein Projekt abgeschlossen.

Anhang

Index.html

```
<!DOCTYPE html>
<html lang="de">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1" />
 <title>Security Toolbox</title>
 <style>
  html, body {
   margin: 0;
   padding: 0;
   height: 100%;
   background: #f3f4f6;
   font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  }

  body {
   display: flex;
   justify-content: center;
   align-items: center;
   min-height: 100vh;
  }

  .container {
   background: white;
   padding: 40px 30px;
   border-radius: 12px;
   box-shadow: 0 8px 20px rgba(0,0,0,0.1);
   max-width: 480px;
   width: 100%;
```

password.html

```
<!DOCTYPE html>
<html lang="de">
<head>
 <meta charset="UTF-8">
 <title>Passwortgenerator</title>
 <link rel="stylesheet" href="style.css">
 <style>
  body {
   font-family: 'Segoe UI', sans-serif;
   background-color: #f3f4f6;
   margin: 0;
   padding: 40px;
   display: flex;
   justify-content: center;
  }

  .container {
   background-color: white;
   padding: 40px;
   border-radius: 12px;
   box-shadow: 0 8px 20px rgba(0,0,0,0.1);
   max-width: 500px;
   width: 100%;
   text-align: center;
  }

  h1 {
   margin-bottom: 30px;
   font-size: 28px;
```

ipcheck.html

```
<!DOCTYPE html>
<html lang="de">
<head>
 <meta charset="UTF-8">
 <title>IP-Check Tool</title>
 <style>
  body { font-family: sans-serif; padding: 20px; background: #f4f4f4; }
  h1 { color: #333; }
  .info { background: #fff; padding: 15px; border-radius: 5px; margin-top: 10px; }
 </style>
</head>
<body>
<div style="margin: 20px 0;">
 <a href="/index.html" style="
  display: inline-block;
  padding: 10px 20px;
  background-color: #006eff;
  color: white;
  font-weight: 600;
  border-radius: 8px;
  text-decoration: none;
  box-shadow: 0 4px 10px rgba(0,110,255,0.4);
  transition: background-color 0.3s ease;
 "
 onmouseover="this.style.backgroundColor='#0050c8';"
 onmouseout="this.style.backgroundColor='#006eff';"
 >Zur  ck zum Men  </a>
</div>
```

ipcheck.py

```
from flask import Flask, request, jsonify
import requests

app = Flask(__name__)

@app.route("/ip")
def ipinfo():
    forwarded = request.headers.get("X-Forwarded-For", request.remote_addr)
    ip = forwarded.split(',')[0]  # falls es mehrere sind, nimm den ersten
    try:
        r = requests.get(f"http://ip-api.com/json/{ip}")
        data = r.json()
    except:
        data = {"error": "API request failed"}

    return jsonify({
        "ip": ip,
        "country": data.get("country", "Unbekannt"),
        "city": data.get("city", "Unbekannt"),
        "isp": data.get("isp", "Unbekannt")
    })

if __name__ == "__main__":
    app.run(host="127.0.0.1", port=5000)
```

headerscan.html

```
<!DOCTYPE html>
<html lang="de">
<head>
 <meta charset="UTF-8">
 <title>HTTP Header Scanner</title>
 <style>
  body { font-family: sans-serif; padding: 20px; background: #f4f4f4; }
  textarea { width: 100%; height: 300px; font-family: monospace; }
  input[type=text] { width: 80%; padding: 10px; }
  button { padding: 10px; }
 </style>
</head>
<body>
<div style="margin: 20px 0;">
 <a href="/index.html" style="
  display: inline-block;
  padding: 10px 20px;
  background-color: #006eff;
  color: white;
  font-weight: 600;
  border-radius: 8px;
  text-decoration: none;
  box-shadow: 0 4px 10px rgba(0,110,255,0.4);
  transition: background-color 0.3s ease;
 "
 onmouseover="this.style.backgroundColor='#0050c8';"
 onmouseout="this.style.backgroundColor='#006eff';"
 >Zur  ck zum Men  </a>
</div>
```

headerscan.py

```
from flask import Flask, request, jsonify
import requests

app = Flask(__name__)

@app.route("/scan", methods=["POST"])
def scan():
    url = request.form.get("url")
    try:
        r = requests.get(url, timeout=5)
        return jsonify(dict(r.headers))
    except Exception as e:
        return jsonify({"Fehler": str(e)}), 400

if __name__ == "__main__":
    app.run(host="127.0.0.1", port=5050)
app.run(host="127.0.0.1", port=5050)
```

passwordcheck.html

```
<!DOCTYPE html>
<html lang="de">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Passwortprüfung</title>
<style>
 body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background: #f3f4f6;
  height: 100vh;
  margin: 0;
  display: flex;
  justify-content: center;
  align-items: center;
 }
 .container {
  background: white;
  padding: 40px 30px;
  border-radius: 12px;
  box-shadow: 0 8px 20px rgba(0,0,0,0.1);
  max-width: 420px;
  width: 100%;
  text-align: center;
 }
 input[type="password"] {
  width: 100%;
  padding: 10px;
  font-size: 1.1em;
```

hashgen.html

```
<!DOCTYPE html>
<html lang="de">
<head>
 <meta charset="UTF-8">
 <title>Hash-Generator</title>
 <link rel="stylesheet" href="style.css">
 <script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.1.1/crypto-
js.min.js"></script>
</head>
<body>
 <div class="container">
  <h1> ^=^t^p Hash-Generator</h1>
  <textarea id="input" placeholder="Text eingeben..."></textarea>

  <select id="algo">
   <option value="SHA256">SHA-256</option>
   <option value="SHA512">SHA-512</option>
   <option value="MD5">MD5</option>
  </select>

  <div class="btn-row">
   <button class="generate-btn" onclick="generate()">Hash generieren</button>
   <button class="copy-btn" onclick="copyToClipboard()">Kopieren</button>
  </div>

  <input type="text" id="output" readonly>

  <a href="/index.html"><button class="back-btn">Zur  ck zur Startseite</button></a>
 </div>
```

sizeconverter.html

```
<!DOCTYPE html>
<html lang="de">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1" />
 <title>Dateigrößen-Konverter</title>
 <style>
  body {
   font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
   background: #f3f4f6;
   display: flex;
   justify-content: center;
   align-items: center;
   height: 100vh;
   margin: 0;
  }
  .container {
   background: white;
   padding: 30px;
   border-radius: 12px;
   box-shadow: 0 8px 20px rgba(0,0,0,0.1);
   max-width: 480px;
   width: 100%;
   text-align: center;
  }
  input[type="number"], select, button {
   width: 100%;
   padding: 10px;
   margin: 10px 0;
```

hexconverter.html

```
<!DOCTYPE html>
<html lang="de">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1" />
 <title>Hex to Decimal Converter</title>
 <style>
  body {
   font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
   background: #f3f4f6;
   display: flex;
   justify-content: center;
   align-items: center;
   height: 100vh;
   margin: 0;
  }
  .container {
   background: white;
   padding: 30px;
   border-radius: 12px;
   box-shadow: 0 8px 20px rgba(0,0,0,0.1);
   max-width: 480px;
   width: 100%;
   text-align: center;
  }
  input[type="text"], button {
   width: 100%;
   padding: 10px;
   margin: 10px 0;
```

Link zum GitHub Repository

https://github.com/Stevo-08/Abschlussprojekt_2025_PLA-2_stemed_Board_Security-Toolbox-als-Webdienst-in-der-Cloud