1.6 Prüfungen kontrolliert durchführen und Ergebnisse bewerten

Eine gute Hypothese ist nur dann nützlich, wenn sie mit einem kontrollierten und aussagekräftigen Test geprüft wird.

Dabei gilt:

> Gleicher Ausgangszustand, gleicher Test, nur eine veränderte Variable und ein vorher festgelegtes erwartetes Ergebnis.

Wer mehrere Dinge gleichzeitig verändert, kann die Wirkung später keiner einzelnen Maßnahme zuordnen.

Ziel dieser Seite

Nach diesem Arbeitsschritt sollten:

- Prüfziel und Hypothese eindeutig feststehen,
- Ausgangszustand und Vergleichswerte dokumentiert sein,
- Risiken und mögliche Auswirkungen des Tests bekannt sein,
- Erfolgskriterien und Abbruchkriterien definiert sein,
- möglichst nur eine Variable verändert werden,
- Befehle, Ausgaben und Rückgabewerte gesichert sein,
- Ergebnisse reproduzierbar sein,
- unklare Tests nicht als Bestätigung interpretiert werden,
- der Zustand nach dem Test erneut geprüft sein.

---

1. Kennzeichnung der Befehle

Kennzeichnung	Bedeutung
`[RO]`	Read-only: liest Informationen aus
`[TEST]`	Führt eine aktive Prüfung aus und kann Netzwerkverkehr oder Protokolleinträge erzeugen
`[FILE]`	Erstellt oder überschreibt eine Datei
`[PRIV]`	Benötigt möglicherweise Administrator- oder Root-Rechte
`[CHANGE]`	Verändert Konfiguration oder Betriebszustand
`[DISRUPTIV]`	Kann Benutzer oder produktive Dienste beeinträchtigen

Kennzeichnung	Bedeutung
`[SENSITIV]`	Ausgabe kann vertrauliche Daten enthalten

Auch ein `[RO]`- oder `[TEST]`-Befehl ist nicht vollständig spurlos. Er startet mindestens einen Prozess und kann in Protokollen, Shell-Historien, Firewalls oder SIEM-Systemen erscheinen.

---

2. Der kontrollierte Prüfablauf

Phase	Tätigkeit
1. Prüfziel	Welche Frage soll der Test beantworten?
2. Hypothese	Welche mögliche Ursache wird geprüft?
3. Ausgangszustand	Welche Messwerte liegen vor dem Test vor?
4. Vorhersage	Welches Ergebnis wird erwartet?
5. Risiko	Welche Nebenwirkungen kann der Test haben?
6. Abbruchkriterium	Wann muss der Test sofort beendet werden?
7. Durchführung	Exakten Befehl oder Ablauf verwenden
8. Aufzeichnung	Zeit, Ausgabe und Rückgabewert sichern
9. Wiederholung	Ergebnis bei Bedarf kontrolliert reproduzieren
10. Rückkehr	Ausgangszustand wiederherstellen
11. Vergleich	Vorher- und Nachher-Ergebnis vergleichen
12. Bewertung	Hypothese bestätigen, stützen, offenlassen oder widerlegen

---

3. Testarten nach Eingriffsrisiko

Testart	Beschreibung	Beispiel	Risiko
Passive Abfrage	Liest vorhandenen Zustand	Prozessliste, Logs, Routingtabelle	Sehr niedrig
Aktiver Funktionstest	Sendet eine kontrollierte Anfrage	DNS-, Ping-, TCP- oder HTTP-Test	Niedrig
Vergleichstest	Vergleicht betroffenes und funktionierendes System	Client A gegen Client B	Niedrig
Wiederholungstest	Prüft, ob das Ergebnis reproduzierbar ist	fünf identische TCP-Tests	Niedrig bis mittel
Isolationstest	Verändert nur den Testkontext	bestimmtes Backend mit `curl --resolve`	Niedrig bis mittel

Testart	Beschreibung	Beispiel	Risiko
Reversible Änderung	Verändert vorübergehend eine Einstellung	freigegebene Testregel	Mittel
Dienstbezogener Test	Startet oder beendet einen Dienst	Dienstneustart	Hoch
Failover-Test	Verlagert produktiven Betrieb	Wechsel auf zweiten Server	Hoch
Lasttest	Erzeugt zusätzliche Last	viele parallele Anfragen	Hoch
Wiederherstellungstest	Setzt Konfiguration oder Version zurück	Rollback	Hoch
Destruktiver Test	Kann Daten oder Zustand zerstören	Löschen, Zurücksetzen, Neuinstallation	Sehr hoch

Passive, aktive und vergleichende Tests sollten normalerweise vor verändernden oder potenziell störenden Tests durchgeführt werden.

---

4. Prüfung und Reparatur nicht vermischen

Reine Prüfung

`[TEST] Test-ComputerSecureChannel -Verbose`

Dieser Befehl prüft auf einem Windows-Domänenmitglied den sicheren Kanal.

Verändernde Reparatur

`[CHANGE] Test-ComputerSecureChannel -Repair`

Der Parameter `-Repair` verändert den Zustand und ist keine reine Prüfung mehr.

Ein ähnlicher Unterschied besteht bei vielen Werkzeugen:

Prüfung	Veränderung
`Get-Service`	`Restart-Service`
`systemctl status`	`systemctl restart`
`ipconfig /displaydns`	`ipconfig /flushdns`
`Get-NetRoute`	`New-NetRoute` oder `Remove-NetRoute`
`journalctl`	Löschen oder Rotieren von Logs
`diff`	Überschreiben der Konfiguration
Firewall-Log ansehen	Firewall-Regel ändern
Zertifikat anzeigen	Zertifikat ersetzen

Prüfung	Veränderung
Updateverlauf anzeigen	Update installieren oder entfernen

Prüfung und Reparatur sollten im Ticket als getrennte Schritte dokumentiert werden.

---

5. Prüfziel präzise formulieren

Ungeeignet

`Ich teste das Netzwerk.`

Besser

`Es wird geprüft, ob Client PC-030 aus VLAN 30 am 30.07.2026 um 15:30 Uhr eine TCP-Verbindung zu server.example auf Port 443 herstellen kann.`

Noch besser

`Hypothese H1: Eine ACL blockiert TCP 443 aus VLAN 30. Erwartet wird, dass der TCP-Test aus VLAN 30 fehlschlägt und derselbe Test aus VLAN 40 erfolgreich ist.`

Ein präzises Prüfziel enthält:

- Hypothesennummer,
- Quellsystem,
- Quell-IP oder Quellnetz,
- Zielsystem,
- Ziel-IP,
- Protokoll,
- Port,
- Benutzerkontext,
- Zeitpunkt und Zeitzone,
- erwartetes Ergebnis,
- widerlegendes Ergebnis.

---

6. Ausgangszustand als Baseline erfassen

Eine Baseline ist der dokumentierte Zustand vor dem Test.

Typische Baseline-Werte

- aktuelle Uhrzeit,
- Systemzustand,
- Dienststatus,
- Prozess-ID,
- CPU- und Speicherauslastung,

- Netzwerkverbindungen,
- IP-Konfiguration,
- DNS-Ergebnis,
- Routingweg,
- Anwendungsergebnis,
- Antwortzeit,
- Fehlercode,
- Ereignisprotokolle,
- Monitoringzustand.

Beispiel

Messwert	Vor dem Test
Zeitpunkt	2026-07-30 15:30:00 CEST
Quelle	PC-030
Quell-IP	192.0.2.30
Quell-VLAN	VLAN 30
Ziel	server.example
Ziel-IP	198.51.100.20
Zielport	TCP 443
DNS-Auflösung	erfolgreich
TCP-Test	fehlgeschlagen
HTTP-Test	keine Verbindung
Ping	keine Antwort
Andere Ziele	erreichbar
Vergleichsclient aus VLAN 40	funktioniert

Die IP-Adressen aus den Netzen `192.0.2.0/24` und `198.51.100.0/24` sind für Dokumentationsbeispiele reserviert.

---

7. Nur eine Variable verändern

Ungeeigneter Test

Gleichzeitig werden:

- Client neu gestartet,
- DNS-Cache geleert,
- Netzwerkkabel gewechselt,
- Firewall deaktiviert,

- Update installiert,
- Dienst neu gestartet.

Wenn der Fehler danach verschwunden ist, bleibt unklar, welche Maßnahme wirksam war.

Besseres Vorgehen

1. Ausgangszustand sichern.
2. DNS-Auflösung prüfen.
3. TCP-Verbindung prüfen.
4. Vergleichsclient testen.
5. anderes Kabel oder anderen Port testen.
6. nach jedem Schritt denselben Funktionstest wiederholen.
7. erst danach eine freigegebene Konfigurationsänderung durchführen.

---

8. Kontrollvariable und Vergleichssystem verwenden

Ein Vergleichssystem hilft zu erkennen, ob eine Ursache:

- benutzerbezogen,
- clientbezogen,
- netzwerkbezogen,
- serverbezogen oder
- anwendungsbezogen ist.

Test	Konstante Werte	Veränderte Variable
Gleicher Benutzer, anderer Client	Benutzer, Ziel, Funktion	Client
Anderer Benutzer, gleicher Client	Client, Ziel, Funktion	Benutzer
Gleicher Client, anderes VLAN	Client, Benutzer, Ziel	Netzwerkpfad
Gleiches VLAN, anderer Client	Netzwerk, Ziel	Client
Gleiches Ziel, direkter Porttest	Client, Netzwerk, Ziel	Anwendungsebene
Gleiche URL, bestimmtes Backend	Client, URL, Hostname	Backend
Gleiche Datei, anderes Konto	Datei, Client, Pfad	Berechtigung
Gleicher Dienst, localhost	Server, Dienst, Port	Netzwerkpfad entfällt

Je weniger Variablen sich zwischen den Testfällen unterscheiden, desto aussagekräftiger ist der Vergleich.

---

9. Testzeitpunkt dokumentieren

Windows

`[RO] Get-Date -Format o`

UTC-Zeit

`[RO] (Get-Date).ToUniversalTime().ToString("o")`

Linux

`[RO] date --iso-8601=seconds`

macOS

`[RO] date "+%Y-%m-%dT%H:%M:%S%z"`

Der Zeitpunkt sollte unmittelbar vor und nach dem Test erfasst werden. Dadurch kann die Ausgabe mit Server-, Firewall-, Proxy- und SIEM-Protokollen verglichen werden.

---

10. PowerShell-Sitzung protokollieren

PowerShell kann Befehle und Konsolenausgaben einer Sitzung in einer Transkriptdatei aufzeichnen.

Protokollierung starten

`[FILE][SENSITIV] Start-Transcript -Path "<Zielpfad>\Testprotokoll.txt" -IncludeInvocationHeader -NoClobber`

Protokollierung beenden

`[FILE] Stop-Transcript`

`-IncludeInvocationHeader` ergänzt Zeitinformationen zu den ausgeführten Befehlen.

`-NoClobber` verhindert, dass eine bereits vorhandene Datei überschrieben wird.

Zu beachten

- Das Transkript kann Benutzernamen, interne Pfade und Befehlsparameter enthalten.
- Zugangsdaten oder Token dürfen nicht direkt in Befehle geschrieben werden.
- GUI-Aktionen werden nicht erfasst.
- Nicht jede externe Anwendung wird vollständig abgebildet.
- Die Datei muss an einem freigegebenen Speicherort abgelegt werden.

---

11. Linux- und macOS-Terminalsitzung protokollieren

Das Werkzeug `script` erstellt eine Aufzeichnung der Terminalsitzung.

Aufzeichnung starten

`[FILE][SENSITIV] script "<Zielpfad>/Testprotokoll.txt"`

Anschließend werden die vorgesehenen Diagnosebefehle ausgeführt.

Aufzeichnung beenden

`exit`

Alternativ kann die Sitzung mit `Strg + D` beendet werden.

Zu beachten

- Die Datei kann Steuerzeichen enthalten.
- Eingaben und Ausgaben können sensible Informationen enthalten.
- Interaktive Programme werden nicht immer als sauberer Text dargestellt.
- Der genaue Funktionsumfang unterscheidet sich zwischen Linux und macOS.
- Die lokale Dokumentation kann mit `man script` aufgerufen werden.

---

12. Einzelne PowerShell-Ausgabe in Datei und Konsole schreiben

`Tee-Object` zeigt die Ausgabe auf der Konsole an und speichert sie gleichzeitig in einer Datei.

`[FILE] Test-NetConnection server.example -Port 443 -InformationLevel Detailed | Tee-Object -FilePath "<Zielpfad>\tcp-test.txt"`

Mit Zeitstempel

```
Get-Date -Format o |
    Tee-Object -FilePath "<Zielpfad>\tcp-test.txt"

Test-NetConnection server.example -Port 443 -InformationLevel Detailed |
    Tee-Object -FilePath "<Zielpfad>\tcp-test.txt" -Append
```

`-Append` fügt die Ausgabe an die vorhandene Datei an.

---

13. Einzelne Linux- oder macOS-Ausgabe sichern

Ausgabe und Fehlermeldungen in eine Datei schreiben

`[FILE] <Befehl> > "<Zielpfad>/test-output.txt" 2>&1`

Ausgabe gleichzeitig anzeigen und speichern

`[FILE] <Befehl> 2>&1 | tee "<Zielpfad>/test-output.txt"`

> Bei einer Pipeline mit `tee` kann der direkt danach angezeigte Rückgabewert zur letzten Komponente der Pipeline gehören. Wenn der ursprüngliche Rückgabewert wichtig ist, sollte der Befehl zunächst ohne Pipeline ausgeführt und anschließend die Datei angezeigt werden.

Rückgabewert zuverlässig sichern

```
started_at=$(date "+%Y-%m-%dT%H:%M:%S%z")
curl --connect-timeout 5 --max-time 15 https://server.example/ > "<Zielpfad>/curl-test.txt" 2>&1
test_rc=$?
ended_at=$(date "+%Y-%m-%dT%H:%M:%S%z")

echo "Start: $started_at"
echo "Ende: $ended_at"
echo "Exit-Code: $test_rc"
cat "<Zielpfad>/curl-test.txt"
```

Der Rückgabewert muss unmittelbar nach dem geprüften Befehl in einer Variablen gesichert werden.

---

14. Rückgabewerte unter Windows auswerten

PowerShell-Cmdlets

`$?` zeigt an, ob die unmittelbar vorherige PowerShell-Operation erfolgreich abgeschlossen wurde.

`[RO] $?`

Mögliche Werte:

- `$true`
- `$false`

Native Programme

`$LASTEXITCODE` enthält den Rückgabewert des zuletzt ausgeführten nativen Programms.

```
curl.exe --connect-timeout 5 --max-time 15 https://server.example/
$curlExitCode = $LASTEXITCODE
"curl Exit-Code: $curlExitCode"
```

Windows-Eingabeaufforderung

`[RO] echo %ERRORLEVEL%`

> Der Rückgabewert muss direkt nach dem betreffenden Programm geprüft werden. Ein nachfolgender Befehl kann den gespeicherten Wert verändern.

---

15. Rückgabewerte unter Linux und macOS auswerten

`$?` enthält den Rückgabewert des zuletzt ausgeführten Befehls.

```
curl --connect-timeout 5 --max-time 15 https://server.example/
test_rc=$?
echo "Exit-Code: $test_rc"
```

Unter Unix-artigen Systemen bedeutet üblicherweise:

Rückgabewert	Allgemeine Bedeutung
`0`	Programm meldet Erfolg
ungleich `0`	Programm meldet Fehler oder besonderen Zustand

Die genaue Bedeutung hängt vom jeweiligen Programm ab und muss in dessen Dokumentation geprüft werden.

---

16. Erfolg des Werkzeugs und Erfolg der Funktion unterscheiden

Ein Rückgabewert `0` bedeutet nicht automatisch, dass die vom Benutzer benötigte Funktion erfolgreich war.

Beispiel mit curl

Ohne zusätzliche Option kann curl eine HTTP-Antwort `404`, `403` oder `500` empfangen und trotzdem mit Exit-Code `0` enden. Die Netzwerkübertragung war erfolgreich, obwohl die Anwendung einen Fehlerstatus zurückgab.

Deshalb müssen mindestens zwei Ebenen getrennt werden:

Ebene	Beispiel
Werkzeugausführung	curl konnte eine HTTP-Antwort empfangen
Fachliche Funktion	Benutzer konnte sich erfolgreich anmelden

Ebene	Beispiel
Anwendungsstatus	Server antwortete mit HTTP 200
Inhaltliche Richtigkeit	Gelieferte Daten sind vollständig und korrekt

HTTP-Fehler als curl-Fehler behandeln

`[TEST] curl --fail --connect-timeout 5 --max-time 15 https://server.example/`

Die Option `--fail` lässt curl bei HTTP-Statuscodes ab 400 einen Fehler melden. Dabei wird der Antworttext normalerweise nicht ausgegeben.

Für eine Diagnose ist es häufig besser, zunächst HTTP-Status und Ausgabe separat zu dokumentieren.

---

17. Zeitbegrenzungen verwenden

Tests sollten nicht unbegrenzt warten.

Windows

`[TEST] curl.exe --connect-timeout 5 --max-time 15 https://server.example/`

Linux

`[TEST] curl --connect-timeout 5 --max-time 15 https://server.example/`

macOS

`[TEST] curl --connect-timeout 5 --max-time 15 https://server.example/`

Option	Bedeutung
`--connect-timeout 5`	maximal fünf Sekunden für die Verbindungsphase
`--max-time 15`	maximal 15 Sekunden für den gesamten Transfer

Zur Verbindungsphase gehören bei curl unter anderem DNS-Auflösung sowie TCP-, TLS- oder QUIC-Verbindungsaufbau.

---

18. HTTP-Zeiten gezielt messen

curl kann einzelne Zeitabschnitte einer HTTP-Verbindung ausgeben.

Windows PowerShell

`[TEST] curl.exe -sS -o NUL -w 'code=%{http_code} remote=%{remote_ip} dns=%{time_namelookup} connect=%{time_connect} tls=%{time_appconnect} first_byte=%{time_starttransfer} total=%{time_total}\n' --`

`connect-timeout 5 --max-time 15 https://server.example/`

Linux und macOS

`[TEST] curl -sS -o /dev/null -w "code=%{http_code} remote=%{remote_ip} dns=%{time_namelookup} connect=%{time_connect} tls=%{time_appconnect} first_byte=%{time_starttransfer} total=%{time_total}\n" --connect-timeout 5 --max-time 15 https://server.example/`

Bedeutung der Werte

Wert	Bedeutung
`http_code`	empfangener HTTP-Status
`remote_ip`	tatsächlich verwendete Ziel-IP
`time_namelookup`	Zeit bis zum Abschluss der Namensauflösung
`time_connect`	Zeit bis zum TCP-Verbindungsaufbau
`time_appconnect`	Zeit bis zum Abschluss des TLS-Handshakes
`time_starttransfer`	Zeit bis zum ersten empfangenen Byte
`time_total`	gesamte Übertragungszeit

Die Zeitwerte werden in Sekunden ausgegeben.

---

19. HTTP-Zeitwerte interpretieren

Beobachtung	Mögliche Ursache
`time_namelookup` hoch	DNS-Server, Resolver oder Weiterleitung langsam
`time_connect` hoch	Netzwerkpfad, Firewall, Überlastung oder Server
`time_appconnect` deutlich höher	TLS-Handshake, Zertifikat oder Kryptografie
`time_starttransfer` hoch	Anwendung oder Backend benötigt lange
`time_total` hoch, erstes Byte schnell	langsame Datenübertragung oder große Antwort
wechselnde `remote_ip`	DNS-Rotation, Load Balancer oder mehrere Backends
wechselnder HTTP-Status	unterschiedliche Backends oder instabiler Dienst
`http_code=000`	keine gültige HTTP-Antwort empfangen

Ein einzelner Messwert reicht bei sporadischen Fehlern nicht aus. Mehrere Messungen mit Zeitstempel sind aussagekräftiger.

---

20. Tests wiederholen, ohne unnötige Last zu erzeugen

Windows-Ping

`[TEST] ping -n 10 <IP-Adresse>`

Linux und macOS

`[TEST] ping -c 10 <IP-Adresse>`

Wiederholter TCP-Test unter Windows

```
1..5 | ForEach-Object {
    $result = Test-NetConnection server.example -Port 443 -InformationLevel Detailed

    [pscustomobject]@{
        Time          = Get-Date -Format o
        RemoteAddress = $result.RemoteAddress
        SourceAddress = $result.SourceAddress
        Success       = $result.TcpTestSucceeded
    }

    Start-Sleep -Seconds 2
}
```

Wiederholter TCP-Test unter Linux und macOS

```
for test_number in 1 2 3 4 5
do
    date "+%Y-%m-%dT%H:%M:%S%z"
    nc -vz -w 3 server.example 443
    sleep 2
done
```

Vor Wiederholungstests ist zu prüfen:

- Wie viel Last erzeugt eine Anfrage?
- Kann ein Konto gesperrt werden?
- Greift ein Rate Limit?
- Wird ein Alarm im SIEM ausgelöst?
- Kann die Anwendung Daten verändern?
- Ist der Test in der Produktion zulässig?

---

21. Sporadische Fehler richtig untersuchen

Ein sporadischer Fehler kann übersehen werden, wenn nur ein einzelner Test durchgeführt wird.

Zu dokumentieren sind:

- Anzahl der Tests,
- zeitlicher Abstand,
- Anzahl erfolgreicher Tests,
- Anzahl fehlgeschlagener Tests,
- genaue Fehlercodes,
- verwendete Ziel-IP,
- verwendetes Backend,
- Antwortzeiten,
- Zeitpunkt jedes Fehlers.

Beispiel

Versuch	Uhrzeit	Ziel-IP	TCP	HTTP	Gesamtzeit
1	15:30:00	198.51.100.20	erfolgreich	200	0,21 s
2	15:30:05	198.51.100.21	erfolgreich	503	4,92 s
3	15:30:10	198.51.100.20	erfolgreich	200	0,19 s
4	15:30:15	198.51.100.21	erfolgreich	503	5,01 s
5	15:30:20	198.51.100.20	erfolgreich	200	0,22 s

Das Muster deutet auf ein Problem mit dem Backend `198.51.100.21` hin.

---

22. Bestimmtes HTTPS-Ziel ohne DNS-Änderung testen

Der Aufruf einer HTTPS-Seite direkt über ihre IP-Adresse kann zu falschen Ergebnissen führen, weil:

- der HTTP-Host-Header verändert wird,
- TLS-SNI nicht zum erwarteten Hostnamen passt,
- das Zertifikat nicht für die IP-Adresse ausgestellt ist,
- ein anderer virtueller Host antwortet.

Mit `curl --resolve` kann für einen einzelnen Test ein bestimmter Hostname einer bestimmten IP-Adresse zugeordnet werden, ohne DNS oder Hosts-Datei zu verändern.

Windows

`[TEST] curl.exe --resolve "server.example:443:<IP-Adresse>" -v -o NUL https://server.example/`

Linux und macOS

`[TEST] curl --resolve "server.example:443:<IP-Adresse>" -v -o /dev/null https://server.example/`

Dadurch bleiben erhalten:

- URL-Hostname,
- HTTP-Host-Header,
- TLS-SNI,
- normale Zertifikatsprüfung.

Nur die Ziel-IP wird für diesen curl-Aufruf vorgegeben.

Beispiel für zwei Backends

[TEST] curl --resolve "server.example:443:198.51.100.20" -v -o /dev/null https://server.example/

[TEST] curl --resolve "server.example:443:198.51.100.21" -v -o /dev/null https://server.example/

> Das direkte Testen eines Backends kann Load Balancer, WAF oder andere Schutzsysteme umgehen. Es darf nur erfolgen, wenn dieser Test technisch vorgesehen und autorisiert ist.

---

23. Lokalen Dienst und Netzwerkpfad getrennt testen

Windows-Server

[TEST] Test-NetConnection localhost -Port <Port>

[TEST] Test-NetConnection <Eigene-Server-IP> -Port <Port>

[TEST] Test-NetConnection <Servername> -Port <Port>

Linux-Server

[TEST] nc -vz -w 3 localhost <Port>

[TEST] nc -vz -w 3 <Eigene-Server-IP> <Port>

[TEST] nc -vz -w 3 <Servername> <Port>

---

macOS-Server

[TEST] nc -vz -w 3 localhost <Port>

[TEST] nc -vz -w 3 <Eigene-Server-IP> <Port>

[TEST] nc -vz -w 3 <Servername> <Port>

Interpretation

localhost	Eigene IP	Remote-Client	Möglicher Fehlerbereich
erfolgreich	erfolgreich	erfolgreich	Netzwerkgrundfunktion vorhanden
erfolgreich	erfolgreich	fehlgeschlagen	Netzwerkpfad oder Firewall
erfolgreich	fehlgeschlagen	fehlgeschlagen	Listener-Bindung oder lokale Firewall
fehlgeschlagen	fehlgeschlagen	fehlgeschlagen	Dienst oder Anwendung
fehlgeschlagen	erfolgreich	erfolgreich	besondere Proxy-, Container- oder Weiterleitungskonfiguration

---

24. Test mit und ohne Namensauflösung

DNS-basierter Test

`[TEST] Test-NetConnection server.example -Port 443`

Direkter TCP-Test zur bekannten IP

`[TEST] Test-NetConnection <IP-Adresse> -Port 443`

Unter Linux und macOS:

`[TEST] nc -vz -w 3 server.example 443`

`[TEST] nc -vz -w 3 <IP-Adresse> 443`

Interpretation

Name	IP	Möglicher Fehlerbereich
erfolgreich	erfolgreich	DNS und TCP-Grundfunktion vorhanden
fehlgeschlagen	erfolgreich	DNS oder Namensauswahl
erfolgreich	fehlgeschlagen	Testfehler oder unterschiedliche Zieladresse prüfen
fehlgeschlagen	fehlgeschlagen	Netzwerk, Firewall, Dienst oder Zielsystem

Der direkte IP-Test ist für TCP geeignet. Für HTTPS sollte wegen Host-Header, SNI und Zertifikatsprüfung `curl --resolve` verwendet werden.

---

25. Anwendungstest und Benutzerfunktion trennen

Ein technischer Test kann erfolgreich sein, obwohl die Benutzerfunktion weiterhin fehlschlägt.

Prüfebene	Beispiel
DNS	Hostname wird aufgelöst
Netzwerk	Ziel-IP ist erreichbar
Transport	TCP-Port ist erreichbar
TLS	Zertifikat und Handshake funktionieren
HTTP	Server liefert eine Antwort
Authentifizierung	Benutzer kann sich anmelden
Autorisierung	Benutzer darf die Funktion verwenden
Anwendung	gewünschte Aktion wird verarbeitet
Datenebene	Datenbank speichert oder liefert korrekte Daten
Geschäftsprozess	gesamter Arbeitsablauf funktioniert

Beispiel

- TCP 443 erreichbar
- TLS erfolgreich
- HTTP 200 auf Startseite
- Anmeldung erfolgreich
- Datei-Upload schlägt fehl

Die Ursache liegt dann nicht in der grundlegenden Erreichbarkeit, sondern möglicherweise in:

- Upload-Berechtigung,
- Dateigrößenlimit,
- Reverse Proxy,
- WAF,
- Speicherplatz,
- Anwendung,
- Datenbank,
- Backend-Storage.

---

26. Vorher- und Nachher-Test identisch durchführen

Nach einer Maßnahme muss derselbe definierte Test erneut ausgeführt werden.

Ungeeigneter Vergleich

- Vorher: Benutzer berichtet, dass es nicht funktioniert.
- Nachher: Administrator pingt den Server.

Diese Ergebnisse prüfen unterschiedliche Funktionen.

Geeigneter Vergleich

- Vorher: `curl` gegen dieselbe URL mit dokumentiertem Benutzerkontext.
- Maßnahme: eine freigegebene Konfiguration wird geändert.
- Nachher: derselbe `curl`-Befehl vom selben Client.
- Danach: ursprünglicher Benutzerablauf wird wiederholt.

---

27. Vorher-Nachher-Protokoll

Feld	Vorher	Nachher
Zeitpunkt		
Quellsystem		
Quell-IP		
Benutzer		
Zielsystem		
Ziel-IP		
Protokoll und Port		
DNS-Ergebnis		
TCP-Ergebnis		
TLS-Ergebnis		
HTTP-Status		
Antwortzeit		
Anwendungsfunktion		
Fehlermeldung		
Monitoringstatus		
Rückgabewert		

---

28. Erfolgskriterien definieren

Ein Erfolgskriterium muss messbar sein.

Ungeeignet

`Das System läuft wieder normal.`

Besser

- TCP-Verbindung ist bei fünf aufeinanderfolgenden Tests erfolgreich.
- HTTP-Status ist fünfmal `200`.
- Antwortzeit liegt unter dem vereinbarten Schwellenwert.
- Benutzer kann sich anmelden.
- Datei kann hochgeladen und wieder heruntergeladen werden.
- Monitoring meldet für 30 Minuten keinen neuen Fehler.
- Ereignisprotokoll enthält keine neuen relevanten Fehler.
- Vergleichssystem und betroffenes System zeigen dasselbe Ergebnis.

Schwellenwerte dürfen nicht frei erfunden werden. Sie müssen aus:

- SLA,
- Monitoring-Baseline,
- Herstellerangabe,
- Anwendungsanforderung oder
- dokumentiertem Normalzustand

abgeleitet werden.

---

29. Abbruchkriterien definieren

Ein Test muss abgebrochen werden, wenn beispielsweise:

- Datenintegrität gefährdet ist,
- unerwartete Systeme betroffen sind,
- Fehlerrate stark ansteigt,
- CPU-, RAM- oder I/O-Auslastung kritisch steigt,
- Benutzer produktiv beeinträchtigt werden,
- Monitoring neue kritische Alarme meldet,
- eine Sicherheitswarnung auftritt,
- eine Rückfallmaßnahme nicht funktioniert,
- der Test außerhalb des genehmigten Umfangs wirkt,
- das erwartete Zeitfenster überschritten wird.

Abbruchvorlage

Feld	Eintrag
Abbruchbedingung	
Messgröße	
Grenzwert	
Verantwortliche Person	
Sofortmaßnahme	
Rückfallmaßnahme	

Feld	Eintrag
Kommunikationsweg	

30. Testergebnisse klassifizieren

Status	Bedeutung
Bestanden	Definiertes Erfolgskriterium vollständig erreicht
Nicht bestanden	Definiertes Erfolgskriterium nicht erreicht
Teilweise bestanden	Nur ein Teil der Kriterien erreicht
Nicht eindeutig	Ergebnis lässt mehrere Erklärungen zu
Nicht reproduzierbar	Fehler trat bei Wiederholung nicht erneut auf
Abgebrochen	Abbruchkriterium wurde erreicht
Nicht durchgeführt	Test konnte nicht ausgeführt werden
Ungültig	Testbedingungen oder Messung waren fehlerhaft

„Nicht eindeutig“ ist ein gültiges Ergebnis. Es ist besser als eine unberechtigte Bestätigung.

31. Hypothese nach dem Test bewerten

Testergebnis	Bewertung der Hypothese
Vorhersage vollständig eingetroffen	Hypothese wird gestützt
Vorhersage mehrfach reproduziert	Hypothese wird stark gestützt
Gegenprobe erfolgreich	Ursache möglicherweise bestätigt
Ergebnis teilweise passend	Hypothese bleibt offen
Ergebnis mehrdeutig	neuer trennschärferer Test erforderlich
Vorhersage nicht eingetroffen	Hypothese wird geschwächt
Widerlegendes Ergebnis eingetroffen	Hypothese gilt als widerlegt
Testbedingungen fehlerhaft	keine Bewertung möglich

> Ein fehlgeschlagener Test bestätigt nicht automatisch die untersuchte Ursache. Er zeigt zunächst nur, dass die getestete Funktion unter den verwendeten Bedingungen fehlgeschlagen ist.

32. Falsch positive Ergebnisse

Ein Test meldet einen Fehler, obwohl die eigentliche Funktion verfügbar ist.

Beispiele

Test	Falsch positives Ergebnis
Ping	keine Antwort, weil ICMP blockiert ist
Traceroute	Sternchen, obwohl Ziel erreichbar ist
Portscanner	Port wirkt gefiltert, weil Sicherheitskomponente reagiert
DNS-Test	anderer Load-Balancer wird als „falsche IP“ interpretiert
Zertifikatstest	direkter IP-Aufruf erzeugt absichtlich Namensfehler
Dienststatus	Dienst ist gestoppt, weil er socket-aktiviert gestartet wird
HTTP-HEAD	Server lehnt `HEAD` ab, verarbeitet aber `GET`
Monitoring	veralteter Alarm bleibt nach Wiederherstellung aktiv

---

33. Falsch negative Ergebnisse

Ein Test meldet Erfolg, obwohl die Benutzerfunktion weiterhin gestört ist.

Beispiele

Test	Falsch negatives Ergebnis
Ping erfolgreich	Anwendung oder Port kann trotzdem ausgefallen sein
TCP-Port erreichbar	Anwendung kann fehlerhafte Antworten liefern
HTTP 200	Anmeldung oder Fachfunktion kann trotzdem fehlschlagen
Dienststatus „Running“	Prozess kann intern hängen
DNS-Auflösung erfolgreich	zurückgegebene Adresse kann falsch sein
Einzelner Test erfolgreich	sporadischer Fehler wird nicht erfasst
Testkonto funktioniert	normales Benutzerkonto kann weiterhin falsche Rechte haben
Ein Backend funktioniert	anderes Backend kann fehlerhaft sein
Speicherplatz vorhanden	Inodes oder Quota können trotzdem erschöpft sein

---

34. Testwerkzeug und Testpunkt dokumentieren

Ein Ergebnis ist nur verständlich, wenn bekannt ist, wo und wie gemessen wurde.

Zu dokumentieren

- Werkzeugname,
- Werkzeugversion,
- Betriebssystem,
- genauer Befehl,
- Quellsystem,
- Quell-IP,
- Quell-VLAN,
- Benutzerkontext,
- Zielname,
- Ziel-IP,
- Protokoll,
- Port,
- Zeitpunkt,
- Zeitzone,
- Wiederholungsanzahl,
- vollständige Ausgabe,
- Rückgabewert.

Werkzeugversionen anzeigen

PowerShell

`[RO] $PSVersionTable`

curl

`[RO] curl --version`

OpenSSL

`[RO] openssl version -a`

Linux-Kernel

`[RO] uname -a`

macOS

`[RO] sw_vers`

Unterschiedliche Werkzeugversionen können verschiedene Optionen, Protokolle oder Standardwerte verwenden.

---

35. Produktivumgebung gegen Testumgebung abwägen

Umgebung	Vorteil	Nachteil

Produktionssystem	reales Fehlerbild	Risiko für Benutzer und Daten
Testsystem	geringeres Risiko	möglicherweise nicht identische Konfiguration
Staging	produktionsähnlich	möglicherweise andere Daten oder Last
isolierter Client	einzelne Variable kontrollierbar	nicht jeder Netzwerkpfad wird abgebildet
Snapshot-Kopie	Zustand kann untersucht werden	flüchtige Informationen fehlen möglicherweise
Labor	Tests gut reproduzierbar	Produktionsabhängigkeiten fehlen

Ein Testsystem ist nur dann aussagekräftig, wenn relevante Unterschiede zur Produktion dokumentiert sind.

---

36. Änderungen in der Produktion absichern

Vor einer verändernden Prüfung sollten geklärt sein:

- Freigabe,
- Wartungsfenster,
- betroffene Systeme,
- betroffene Benutzer,
- Backup oder Konfigurationssicherung,
- Rückfallplan,
- Abbruchkriterien,
- Monitoring,
- Kommunikation,
- verantwortliche Person,
- erwartete Testdauer,
- Beobachtungszeit nach der Änderung.

Freigabevorlage

Feld	Eintrag
Ticketnummer	
Hypothese	
geplante Änderung	
betroffene Systeme	
erwartete Auswirkung	
Risiko	
Wartungsfenster	

Feld	Eintrag
Backup vorhanden	Ja / Nein
Rückfallplan	
Abbruchkriterium	
Freigabe durch	
ausführende Person	
beobachtende Person	

---

37. Tests mit Anmeldungen vorsichtig durchführen

Anmeldetests können:

- Konten sperren,
- MFA-Anfragen auslösen,
- Sicherheitsalarme erzeugen,
- Sitzungen beenden,
- Token erzeugen,
- Lizenzplätze belegen,
- Änderungen im Benutzerprofil auslösen.

Vor einem Anmeldetest prüfen:

- Ist das Kennwort sicher bekannt?
- Wie viele Fehlversuche sind erlaubt?
- Existiert ein freigegebenes Testkonto?
- Wird MFA ausgelöst?
- Darf das Konto auf diesem System verwendet werden?
- Kann die Sitzung wieder sauber beendet werden?
- Werden durch den Test Daten verändert?
- Wird ein SIEM-Alarm erwartet?

Kennwörter, Token und API-Schlüssel dürfen nicht in Protokolldateien, Terminaltranskripten oder Befehlszeilen erscheinen.

---

38. Beispiel eines kontrollierten Tests

Störung

Eine Webanwendung liefert sporadisch HTTP `503`.

Hypothesen

- H1: DNS liefert zeitweise eine falsche IP-Adresse.

- H2: Eines von zwei Backends ist fehlerhaft.
- H3: Der TLS-Handshake schlägt sporadisch fehl.
- H4: Der Client verliert zeitweise die Netzwerkverbindung.

Prüfplan

Schritt	Test	Erwarteter Erkenntnisgewinn
1	DNS mehrfach abfragen	verwendete Zieladressen bestimmen
2	curl-Zeitmessung wiederholen	HTTP-Status, Ziel-IP und Zeitstufe erfassen
3	Backends einzeln mit `--resolve` testen	fehlerhaftes Backend identifizieren
4	TCP- und TLS-Ergebnisse vergleichen	Netzwerk- und TLS-Ursachen bewerten
5	Serverlogs zum Zeitpunkt prüfen	Backendfehler bestätigen

Ergebnis

- DNS liefert abwechselnd `198.51.100.20` und `198.51.100.21`.
- Anfragen an `.20` liefern HTTP `200`.
- Anfragen an `.21` liefern HTTP `503`.
- TCP- und TLS-Verbindung funktionieren zu beiden Adressen.
- Backendprotokoll von `.21` zeigt eine nicht erreichbare Datenbank.

Bewertung

- H1 wird widerlegt: Beide DNS-Adressen sind vorgesehen.
- H2 wird stark gestützt: Der Fehler ist an Backend `.21` gebunden.
- H3 wird widerlegt: TLS funktioniert bei beiden Backends.
- H4 wird widerlegt: TCP-Verbindungen sind stabil.
- Als nächste Hypothese wird die Datenbankverbindung von Backend `.21` geprüft.

---

39. Testprotokoll-Vorlage

Feld	Eintrag
Ticketnummer	
Testnummer	
Hypothesennummer	
Prüfziel	
Datum	
Startzeit	
Endzeit	

Feld	Eintrag
Zeitzone	
ausführende Person	
Quellsystem	
Quell-IP und VLAN	
Zielsystem	
Ziel-IP	
Protokoll und Port	
Benutzerkontext	
Werkzeug und Version	
exakter Befehl	
Ausgangszustand	
erwartetes Ergebnis	
widerlegendes Ergebnis	
tatsächliches Ergebnis	
Rückgabewert	
Ausgabedatei	
Hypothesenbewertung	
Nebenwirkungen	
Rückfall durchgeführt	Ja / Nein / Nicht erforderlich
Zustand nach dem Test	
nächster Schritt	

Kurzcheckliste

- [ ] Prüfziel eindeutig formuliert
- [ ] Hypothese und Vorhersage dokumentiert
- [ ] Widerlegendes Ergebnis festgelegt
- [ ] Ausgangszustand gesichert
- [ ] Quell- und Zielsystem dokumentiert
- [ ] Benutzerkontext dokumentiert
- [ ] Risiko bewertet
- [ ] Freigabe bei veränderndem Test eingeholt

- [ ] Rückfallplan vorhanden
- [ ] Abbruchkriterien definiert
- [ ] Möglichst nur eine Variable verändert
- [ ] Vergleichssystem verwendet
- [ ] Start- und Endzeit dokumentiert
- [ ] Exakten Befehl dokumentiert
- [ ] Vollständige Ausgabe gesichert
- [ ] Rückgabewert gesichert
- [ ] Test bei Bedarf kontrolliert wiederholt
- [ ] Werkzeugversion dokumentiert
- [ ] Vorher- und Nachher-Test identisch durchgeführt
- [ ] Benutzerfunktion zusätzlich technisch geprüft
- [ ] Falsch positive Ergebnisse berücksichtigt
- [ ] Falsch negative Ergebnisse berücksichtigt
- [ ] Hypothesenstatus aktualisiert
- [ ] Zustand nach dem Test kontrolliert
- [ ] Ergebnis im Ticket dokumentiert

Ergebnis dieses Arbeitsschrittes

Am Ende dieses Schrittes liegt ein nachvollziehbares und reproduzierbares Testergebnis vor.

Ein brauchbares Testergebnis beantwortet:

- Was wurde geprüft?
- Welche Hypothese wurde untersucht?
- Von welchem System wurde getestet?
- Unter welchem Benutzerkontext wurde getestet?
- Wann wurde getestet?
- Welcher genaue Befehl wurde verwendet?
- Was wurde vorher erwartet?
- Was wurde tatsächlich beobachtet?
- Welcher Rückgabewert wurde erzeugt?
- Welche Hypothese wird dadurch gestützt oder widerlegt?
- Wurde der Ausgangszustand wiederhergestellt?
- Welcher nächste Schritt ergibt sich daraus?

Nächste Seite:
1.7 Ursache bestätigen und alternative Erklärungen ausschließen

---

Offizielle Hersteller- und Projektdokumentation

- Microsoft Learn – Start-Transcript
- Microsoft Learn – Tee-Object
- Microsoft Learn – Automatische PowerShell-Variablen
- Microsoft Learn – PowerShell-Fehlerbehandlung
- Microsoft Learn – Test-NetConnection
- curl – Offizielle Befehlsreferenz
- curl – Exit-Codes
- util-linux – script(1), technischer Manual-Spiegel
- GNU Bash – Exit Status
