**JURTEKSI (Jurnal Teknologi dan Sistem Informasi)** ISSN 2407-1811 (Print)
Vol. XII No 1, Desember 2025, hlm. 9 – 20 ISSN 2550-0201 (Online)
DOI: http://dx.doi.org/ 10.33330/jurteksi.v12i1.4237
Available online at http://jurnal.stmikroyal.ac.id/index.php/jurteksi

# GRAFANA-BASED DOMAIN EXPIRATION AND SSL CERTIFICATE MONITORING SYSTEM FOR PREVENTIVE SECURITY

**Nur Asyiyah[1], Hafiyyan Putra Pratama[1*]**
[1]Telecommunication System, Universitas Pendidikan Indonesia
*email*: *nurasyiyah@upi.edu hafiyyan@upi.edu

**Abstract:** Manual management of domain validity periods and SSL certificates is prone to human error and can cause service disruptions, as was the case at PT XYZ. A reactive approach that relies on vendor notifications has proven to be insufficient to ensure operational continuity. This research aims to design and implement an automated monitoring system to transform this manual approach into a preventive and proactive security framework. The method used is the implementation of an open-source stack consisting of Prometheus to collect metrics from specialized exporters (Blackbox and Domain Exporter), and Grafana for informative centralized dashboard visualization. The system is also integrated with early warning notifications via Telegram for rapid incident response. The result is a functional system with a centralized dashboard that visually displays the remaining validity period of assets using color markers (green for safe status, yellow for early warning, and red for critical status). System testing showed very high accuracy, reaching 100% for domains (MAE 0 days) and 99.45% for SSL certificates (MAE 1.0 days). This system has successfully transformed manual processes into automated and preventive ones, significantly mitigating the risk of human error and ensuring the reliability of digital services.

**Keywords:** domain; grafana; monitoring; prometheus; SSL certificate.

**Abstrak:** Pengelolaan manual masa berlaku domain dan sertifikat SSL rentan terhadap *human error* dan dapat menyebabkan gangguan layanan, seperti yang pernah terjadi di PT XYZ. Pendekatan reaktif yang mengandalkan notifikasi *vendor* terbukti tidak lagi memadai untuk menjamin kontinuitas operasional. Penelitian ini bertujuan merancang dan mengimplementasikan sistem pemantauan otomatis untuk mentransformasi pendekatan manual tersebut menjadi kerangka kerja keamanan yang preventif dan proaktif. Metode yang digunakan adalah implementasi *stack open-source* yang terdiri dari Prometheus untuk mengumpulkan metrik dari *exporter* spesialis (Blackbox dan Domain Exporter), serta Grafana untuk visualisasi dasbor terpusat yang informatif. Sistem ini juga diintegrasikan dengan notifikasi peringatan dini melalui Telegram untuk respons insiden yang cepat. Hasilnya adalah sebuah sistem fungsional dengan *dashboard* terpusat yang menampilkan sisa masa berlaku aset secara visual menggunakan penanda warna (hijau untuk status aman, kuning untuk peringatan dini, dan merah untuk status kritis). Pengujian sistem menunjukkan akurasi yang sangat tinggi, mencapai 100% untuk domain (MAE 0 hari) dan 99.45% untuk sertifikat SSL (MAE 1.0 hari). Sistem ini berhasil mengubah proses manual menjadi otomatis dan preventif, secara signifikan memitigasi risiko *human error* dan menjamin keandalan layanan digital.

**Kata kunci:** domain; grafana; pemantauan; prometheus; sertifikat SSL.

## INTRODUCTION

The security of digital assets such as domains and Secure Sockets Layer (SSL) certificates is now fundamental to the smooth operation of companies that rely on connectivity. Digital certificates play an important role in ensuring the authenticity of digital identities while protecting the security of sensitive data exchanges through encryption, so that data exchanges between users and servers remain safe from cyber threats. If the lifecycle management of these assets, especially their validity period, is not carried out properly, it can pose detrimental risks. The most obvious impact that can occur is the disruption or even cessation of important services such as website access, email communication, web portals, and system integration. This can also undermine public trust and damage the company's image [1]. Expired domains and certificates have the potential to open up dangerous security gaps, which can be exploited for attacks such as man-in-the-middle attacks, certificate forgery, and domain hijacking by irresponsible parties [2]. Management using manual methods is highly prone to human error, especially with the drastic growth of digital assets [3]. This shows that reactive work patterns are no longer effective and can pose significant risks in the modern IT era.

PT XYZ, as one of the largest media companies in Indonesia, focuses on multimedia, entertainment, and communications services. The company operates a number of leading television stations in Indonesia, including both free-to-air and pay TV channels. In addition, the company is also active in film and video production and distribution, content trading, and is highly dependent on the stability and security of digital infrastructure to deliver news and entertainment to millions of viewers [4]. However, digital asset management at PT XYZ still faces significant obstacles. One notable incident occurred when one of PT XYZ's domains slipped through the cracks and expired, resulting in a loss of public access. Although the domain was successfully secured again, this incident highlighted a fundamental weakness in the monitoring system used, which is still reactive and relies entirely on notifications from vendors via email.

To address these issues, an automated, centralized, and preventive monitoring system is needed to provide early warnings before domains and SSL certificates expire. The solution provided in this research is to design and implement a specialized monitoring system using an open-source stack that involves Prometheus as a metric collector, Grafana as a visualization dashboard, and integrated real-time warning notifications via Telegram [5]. A number of studies confirm that the use of Prometheus and Grafana can strengthen IT infrastructure monitoring capabilities [6]. The combination of Prometheus and Grafana can effectively monitor overall system performance metrics, ranging from Central Processing Unit (CPU) usage and Random Access Memory (RAM) usage to disk capacity [7]. Other studies have demonstrated the success of implementing this stack in various environments, such as Kubernetes clusters [8] as well as in large-scale data centers [9]. Several other studies also focus on specific applications for improving performance visibility, such as monitoring load balancing on web servers with HAProxy [10] and monitoring on the MongoDB database system [11].

Meanwhile, integration with real-time notifications via Telegram enables server administrators to immediately respond to any potential disruptions that occur [12]. The use of notifications via Telegram has proven to speed up responses, allowing administrators to take immediate action [13][14]. In addition, the importance of security in data communication has also been emphasized through the implementation of SSL or Transport Layer Security (TLS) to prevent attacks such as Man-in-the-Middle, which confirms that certificate management is an integral part of an organization's cybersecurity posture [15]. Although previous studies have proven the superiority of Prometheus and Grafana in monitoring general infrastructure performance, there is still a gap analysis in applications that focus specifically on preventive security aspects through monitoring the life cycle of digital assets, such as domains and SSL certificates. Most of the literature focuses on performance metrics such as uptime, latency, and resource utilization, while only a few have designed systems to directly anticipate business risks due to the expiration of digital assets, which in fact are often the main cause of downtime. This research fills that gap by not only implementing a standard monitoring stack, but also customizing it specifically for preventive purposes through the use of specialized exporters, namely Blackbox Exporter for SSL validation and Domain Exporter for domain expiration checks. This system is designed to address the real problems faced by PT XYZ, where monitoring failures are not technical in nature, but rather procedural oversights that can be completely prevented through automation.

This research aims to design and build an automated system capable of monitoring and displaying the remaining validity period of domains and SSL certificates within PT XYZ in real time. In addition, this system is also designed to be integrated with early warnings via Telegram, so that the IT team receives warnings well before digital assets expire. With this implementation, PT XYZ is expected to transform from a reactive manual monitoring method to an automated and preventive system, thereby reducing the risk of downtime incidents, improving security posture, and ensuring the reliability of its digital services to the public. The implementation of this proposed system effectively strengthens the company's information technology risk management framework.

## METHOD

This research method is described in its entirety through the flowchart in Image 1.

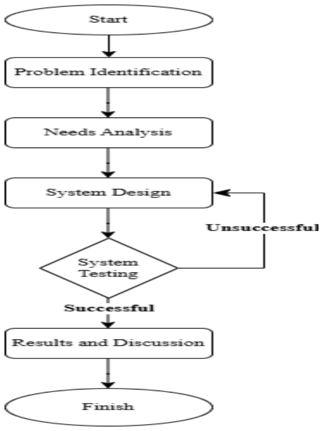


Image 1. Research Process

### Problem Identification

This process was carried out through interviews with the IT team at PT XYZ to obtain a clear picture of the current state of domain and SSL certificate management processes, including any obstacles encountered. The purpose of these interviews was to confirm the occurrence of downtime incidents caused by failures in the manual monitoring process. In addition, ongoing business processes were also observed to identify aspects that could be automated.

### Needs Analysis

After identifying the problem, we formulated functional requirements for a system capable of automatically monitoring the validity period of domains and SSL certificates and providing early warning notifications before expiration via Telegram. Based on previous research, open source stacks such as Prometheus and Grafana have proven to be effective solutions for various infrastructure monitoring scenarios [16]. This is in line with this study, which reinforces the reasons for choosing Prometheus and Grafana as open source tools to support and complement system requirements. Several other supporting software and hardware are also used, as presented in Table 1 and Table 2.

Table 1. Software Specifications

| Tools/Components | Specifications |
|---|---|
| Virtual Box | Version 7.2.0 |
| Operating System | Ubuntu Server 24.04.3 |
| Processor | 4 *Core* |
| Memory (RAM) | 3 GB |
| Prometheus | Version 2.54.1 |
| Grafana | Version 12.1.1 |
| Blackbox Exporter | Version 0.25.0 |
| Domain Exporter | Version 1.24.1 |
| Python venv | Versi 3.12.3 |
| Pyhton pip | Versi 25.2 |
| Telegram | Bot API (*token*) |

Table 2. Hardware Specifications

| Tools/Components | Specifications |
|---|---|
| Laptop | Lenovo ideapad 3 Slim 3 |
| Memory (RAM) | 8 GB |
| Processor | 11th Gen Intel(R) *Core*(TM) i5-1135G7 @ 2.40GHz (2.42 GHz) |
| Storage | 477 GB |
| Operating System | Windows 11 Version 24H2 |

### System Design

This stage involves the installation, configuration, and integration of several key components, namely Blackbox Exporter, Domain Exporter, Prometheus, and Grafana. The overall system workflow is shown in Image 2, which illustrates the interconnection between components in the process of monitoring, metric storage, data visualization, and notification delivery.

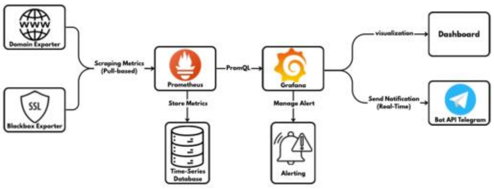
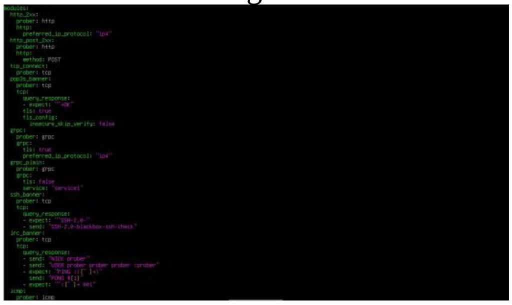


Image 2. System Workflow

The system design stage begins with installing Blackbox Exporter using the following command:

```
wget https://github.com/prometheus/blackbox_exporter/releases/download/v0.25.0/blackbox_exporter-0.25.0.linux-amd64.tar.gz
```

Next, create a configuration file named *blackbox.yml* and configure the file as shown in Image 3.



```
modules:
  http_2xx:
    prober: http
    http:
      preferred_ip_protocol: "ip4"
  http_post_2xx:
    prober: http
    http:
      method: POST
  tcp_connect:
    prober: tcp
  pop3s_banner:
    prober: tcp
    tcp:
      query_response:
      - expect: "^+OK"
      tls: true
      tls_config:
        insecure_skip_verify: false
  grpc:
    prober: grpc
    grpc:
      tls: true
      preferred_ip_protocol: "ip4"
  grpc_plain:
    prober: grpc
    grpc:
      tls: false
      service: "service1"
  ssh_banner:
    prober: tcp
    tcp:
      query_response:
      - expect: "^SSH-2.0-"
      - send: "SSH-2.0-blackbox-ssh-check"
  irc_banner:
    prober: tcp
    tcp:
      query_response:
      - send: "NICK prober"
      - send: "USER prober prober prober :prober"
      - expect: "PING :([^ ]+)"
        send: "PONG ${1}"
      - expect: "^:[^ ]+ 001"
  icmp:
    prober: icmp
```

Image 3. Blackbox Configuration

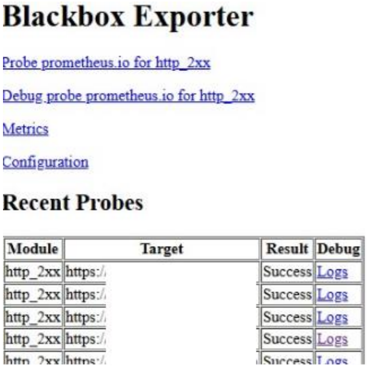
This file contains the probe configuration that will be used to monitor specific targets according to the system design. After configuration is complete, run Blackbox Exporter with nohup so that it can run in the background using the following command:

```
cd ~/blackbox_exporter-0.25.0.linux-amd64
nohup ./blackbox --config.file= blackbox.yml > blackbox.log 2>&1 &
```

Then check port 9115 using the command:

```
sudo ss -tuln | grep 9115
```

If successful, the service can be accessed via the web interface *http://localhost:9115*, as shown in Image 4.



## Blackbox Exporter

Probe prometheus.io for http_2xx

Debug probe prometheus.io for http_2xx

Metrics

Configuration

### Recent Probes

| Module | Target | Result | Debug |
|---|---|---|---|
| http_2xx | https:/ [redacted] | Success | Logs |
| http_2xx | https:/ [redacted] | Success | Logs |
| http_2xx | https:/ [redacted] | Success | Logs |
| http_2xx | https:/ [redacted] | Success | Logs |
| http_2xx | https:/ [redacted] | Success | Logs |

Image 4. Blackbox successfully accessed on the web interface

The next step is to install Domain Exporter, which monitors domain validity periods. Installation is performed by downloading the installation file using the command:

```
wget https://github.com/caarlos0/domain_exporter/releases/download/v1.21.0/domain_exporter_1.21.0_linux-amd64.tar.gz
```

To run Domain Exporter, a Python virtual environment is required. Therefore, install Python venv and pip with the command:

```
sudo apt install -y python3-venv python3-pip
```

Then create a special folder for WHOIS Exporter with the command:

```
sudo mkdir -p /opt/whois_exporter
sudo chown namaserver:namaserver /opt/whois_exporter
cd /opt/whois_exporter
```

After that, a virtual environment was created using the command:

```
pyhton3 -m venv venv
```

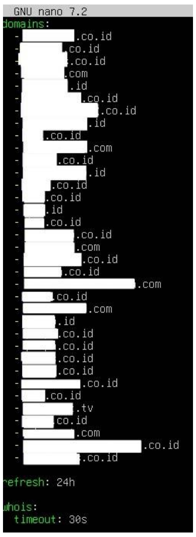
The virtual environment is then activated and the pip update is performed using the command:

```
source venv/bin/activate
pip install –upgrade pip
```

Next, install the required dependencies, namely *python-whois*, *prometheus_client*, and *pytz*, with the command:

```
pip install python-whois prometheus_client pytz
```

Then, in the *domain_exporter* directory, create a *config.yml* configuration file and configure it as shown in Image 5.



```
 GNU nano 7.2
domains:
  - [redacted].co.id
  - [redacted]co.id
  - [redacted].co.id
  - [redacted].com
  - [redacted].id
  - [redacted].co.id
  - [redacted].co.id
  - [redacted].id
  - [redacted].co.id
  - [redacted].com
  - [redacted].co.id
  - [redacted].id
  - [redacted].co.id
  - [redacted].co.id
  - [redacted].id
  - [redacted].co.id
  - [redacted].co.id
  - [redacted].com
  - [redacted].co.id
  - [redacted].co.id
  - [redacted].com
  - [redacted].co.id
  - [redacted].com
  - [redacted].id
  - [redacted].co.id
  - [redacted].co.id
  - [redacted].co.id
  - [redacted].co.id
  - [redacted].co.id
  - [redacted].co.id
  - [redacted].tv
  - [redacted]co.id
  - [redacted].com
  - [redacted].co.id
  - [redacted].co.id

efresh: 24h

hois:
  timeout: 30s
```

Image 5. Domain Exporter Configuration

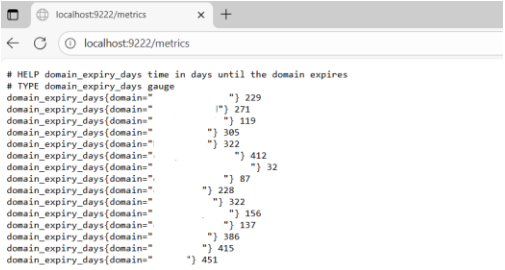
After configuration is complete, Domain Exporter runs in the background using the command:

```
nohup ./domain_exporter --config.file=
/home/(nama server)/domain_config.yml > domain_exporter.log 2>&1 &
```

Then, port 9222 was checked using:

```
sudo ss -tuln | grep 9222
```

After that, the service can be accessed through the web interface *http://localhost:9222*, as shown in Image 6.



```
# HELP domain_expiry_days time in days until the domain expires
# TYPE domain_expiry_days gauge
domain_expiry_days{domain="[redacted]"} 229
domain_expiry_days{domain="[redacted]"} 271
domain_expiry_days{domain="[redacted]"} 119
domain_expiry_days{domain="[redacted]"} 305
domain_expiry_days{domain="[redacted]"} 322
domain_expiry_days{domain="[redacted]"} 412
domain_expiry_days{domain="[redacted]"} 32
domain_expiry_days{domain="[redacted]"} 87
domain_expiry_days{domain="[redacted]"} 228
domain_expiry_days{domain="[redacted]"} 322
domain_expiry_days{domain="[redacted]"} 156
domain_expiry_days{domain="[redacted]"} 137
domain_expiry_days{domain="[redacted]"} 386
domain_expiry_days{domain="[redacted]"} 415
domain_expiry_days{domain="[redacted]"} 451
```

Image 6. Domain Exporter was successfully accessed

Both exporters, Blackbox Exporter and Domain Exporter, present metrics that have been collected through HTTP endpoints so that they can be accessed by Prometheus.

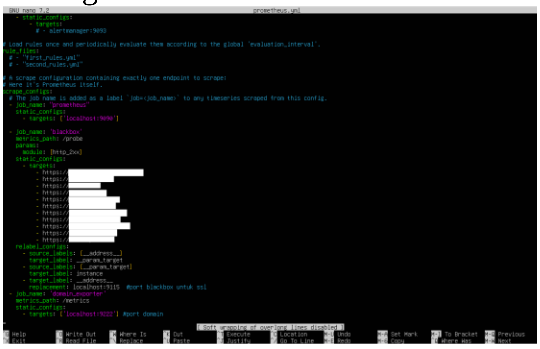
Next, Prometheus was installed as the main component that collects metrics from both exporters. The installation process was carried out by installing the file using the command:

```
wget https://github.com/prometheus/prometheus/releases/download/v2.54.1/
prometheus-2.54.1.linux-amd64.tar.gz
```

Next, the default prometheus.yml configuration file is edited with the command:

```
sudo nano prometheus.yml
```

Then configure the file, as shown in Image 7.



```
    - static_configs:
        - targets:
          # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"
    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'blackbox'
    metrics_path: /probe
    params:
      module: [http_2xx]
    static_configs:
      - targets:
        - https://[redacted]
        - https://[redacted]
        - https://[redacted]
        - https://[redacted]
        - https://[redacted]
        - https://[redacted]
        - https://[redacted]
        - https://[redacted]
        - https://[redacted]
        - https://[redacted]
        - https://[redacted]
    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels: [__param_target]
        target_label: instance
      - target_label: __address__
        replacement: localhost:9115  #port blackbox untuk ssl
  - job_name: 'domain_exporter'
    metrics_path: /metrics
    static_configs:
      - targets: ['localhost:9222'] #port domain

                         [ Soft wrapping of overlong lines disabled ]
^G Help      ^O Write Out  ^W Where Is  ^K Cut    ^T Execute  ^C Location   M-U Undo  M-A Set Mark  M-] To Bracket  M-Q Previous
^X Exit      ^R Read File  ^\ Replace   ^U Paste  ^J Justify  ^/ Go To Line M-E Redo  M-6 Copy      ^Q Where Was    M-W Next
```

Image 7. Prometheus Configuration

After configuration is complete, Prometheus is run with nohup so that it remains active in the background with the command:

```
cd ~/prometheus-2.54.1.linux-amd64
```

```
nohup ./prometheus –config.file=
prometheus.yml –web.listen-address=”0.0.0.0:9090” >
prometheus.log 2>&1 &
```

Then, port 9090 was checked using:

```
sudo ss -tuln | grep 9090
```

If the check is successful, the service can be accessed via the web interface *http://localhost:9090*, as shown in Image 8.

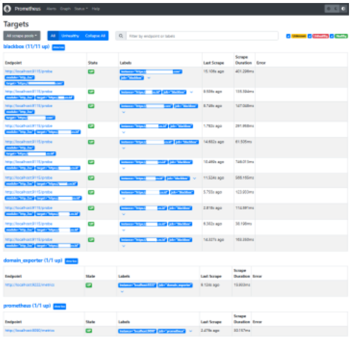


Image 8. Prometheus was successfully accessed on the web interface

The collected data is stored in Prometheus' Time-Series Database. For data processing and access, Prometheus provides a query language called PromQL (Prometheus Query Language) that allows other systems, such as Grafana, to retrieve data as needed. The verification process is done through the Status menu, then selecting the Target option by running the following query:

To check the SSL remaining validity period:

```
(probe_ssl_earliest_cert_expiry -
time()) / 86400
```

To detect the remaining validity period of a domain:

```
domain_expiry_days
```

The next step is to install Grafana, which displays data visualizations from Prometheus. Installation begins by adding the official Grafana repository and supporting dependencies using the command:

```
sudo apt install -y apt-transport-https
software-properties-common wget
wget -q -O - https://packages.grafana
.com/gpg.key | sudo apt-key add -
echo "deb https://packages.grafana.
com/oss/deb stable main" | sudo tee
/etc/apt/sources.list.d/grafana.list
sudo apt install grafana -y
```

After the installation process is complete, the Grafana Server service is run and set to start automatically at boot with the command:

```
sudo systemctl start grafana-server
sudo systemctl status grafana-server
sudo systemctl enable grafana-server
```

Once the service is active, Grafana can be accessed via a browser at *http://localhost:3000* as shown in Image 9, with the initial credentials being Username "admin" and Password "admin".

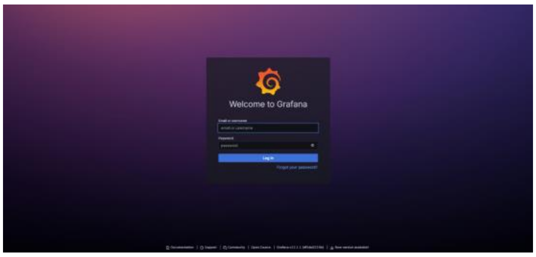


Image 9. Grafana was successfully accessed on the web interface

After all components are installed, all services are restarted so that the system is properly integrated. The process is run with the following command:

For Prometheus:

```
cd ~/prometheus-2.54.1.linux-amd64
nohup ./prometheus –config.file=
prometheus.yml –web.listen-address=
”0.0.0.0:9090” > prometheus.log
2>&1 &
```

For Blackbox Exporter:

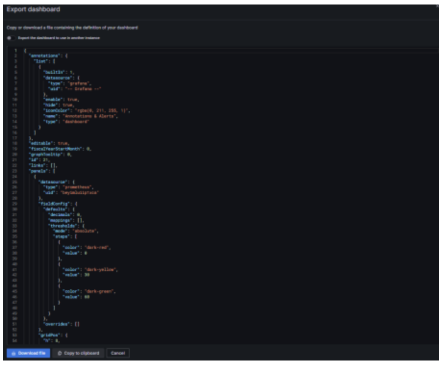
```
cd ~/blackbox_exporter-0.25.0.linux-
amd64
nohup ./blackbox --config.file=
blackbox.yml > blackbox.log 2>&1 &
```

For Domain Exporter:

```
nohup ./domain_exporter --
config.file=
/home/(nama server)/domain_config.
yml > domain_exporter.log 2>&1 &
```

Once all services are active, the integration process between Prometheus and Grafana is carried out. This process is done through the Data Sources menu in Grafana by selecting the Add data source option, then selecting Prometheus, and entering the following URL *http://localhost:9090.*

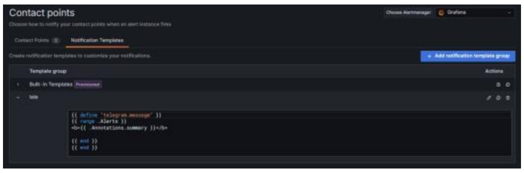
The Grafana dashboard is configured to display monitoring results from Blackbox Exporter and Domain Exporter. Although it can be set up through the user interface, configuration using JSON format is selected so that the layout produces a display that suits monitoring needs.


Export dashboard
Copy or download a file containing the definition of your dashboard
Export the dashboard to use in another instance
Download file
Copy to clipboard
Cancel

```
{
  "annotations": {
    "list": [
      {
        "builtIn": 1,
        "datasource": {
          "type": "grafana",
          "uid": "-- Grafana --"
        },
        "enable": true,
        "hide": true,
        "iconColor": "rgba(0, 211, 255, 1)",
        "name": "Annotations & Alerts",
        "type": "dashboard"
      }
    ]
  },
  "editable": true,
  "fiscalYearStartMonth": 0,
  "graphTooltip": 0,
  "id": 21,
  "links": [],
  "panels": [
    {
      "datasource": {
        "type": "prometheus",
        "uid": "[illegible]"
      },
      "fieldConfig": {
        "defaults": {
          "decimals": 0,
          "mappings": [],
          "thresholds": {
            "mode": "absolute",
            "steps": [
              {
                "color": "dark-red",
                "value": 0
              },
              {
                "color": "dark-yellow",
                "value": 30
              },
              {
                "color": "dark-green",
                "value": 60
              }
            ]
          }
        },
        "overrides": []
      },
      "gridPos": {
        "h": 8,
```

Image 10. JSON Dashboard Format

Then set Alert Rules through the Grafana UI interface. Notification rules are created for two main conditions, namely domain validity period and SSL certificate. These rules are created to provide warnings when the remaining validity period approaches a certain threshold, with a “warning” alert given when the remaining time reaches 60 days, and a “critical” alert given when the remaining time is 30 days, for both domains and SSL certificates. Next, the contact point configuration is done to connect the system with the Telegram Bot API. This integration allows each notification to be sent automatically via the Telegram bot in HTML format, so that alerts can be received in real time. After that, a notification template is created in the contact point configuration by adding a message template as shown in Image 11. This template serves to set the format and content of the notification so that it displays a summary of the warning according to the conditions detected by the system.


Contact points
Choose how to notify your contact points when an alert instance fires
Choose Alertmanager
Grafana
Contact Points 1
Notification Templates
Create notification templates to customize your notifications.
+ Add notification template group
Template group
Actions
Built-in Templates
Provisioned
tele

```
{{ define "telegram.message" }}
{{ range .Alerts }}
<b>{{ .Annotations.summary }}</b>

{{ end }}
{{ end }}
```

Image 11. Notification Templates in Grafana

Then set up Notification Policies to manage notification delivery, such as message delivery frequency and label usage.

## System Testing

The testing was designed through several stages that were carried out systematically to evaluate the accuracy and reliability of the domain validity period and SSL certificate monitoring system. The initial stage began with the selection of samples in the form of several active domains and SSL certificates belonging to PT XYZ, which were used as test objects. The data samples taken were in the form of information on the remaining validity period of the domain and SSL certificate for each sample monitored. Next, the data collection process is carried out by recording the values displayed on the Grafana dashboard, where the data is the result of processing metrics from Prometheus collected through Blackbox Exporter and Domain Exporter. The next stage is manual verification as a comparison to the results from the system. Testing was carried out on each domain and SSL certificate sample. Each sample was compared between the monitoring results from the system and the manual verification results, where domain verification was carried out using an online platform via a browser. System evaluation was carried out by calculating the accuracy rate based on a comparison of the system output results and the manual verification results for each test data using formula (1).

DOI: http://dx.doi.org/ 10.33330/jurteksi.v12i1.4237
Available online at http://jurnal.stmikroyal.ac.id/index.php/jurteksi

$$Akurasi = \frac{Nilai\ Manual\ -\ |Nilai\ Sistem - Nilai\ Manual|}{Nilai\ \ Manual} \times 100\% \quad (1)$$

The equation is used to calculate the accuracy level of each domain sample and SSL certificate based on the difference between the system results and the manual verification results. The accuracy value indicates how close the system results are to the actual values, where a higher accuracy value means that the system is more accurate in displaying the validity period of the domain and SSL certificate. In addition, the difference in values between the system results and the manual verification results is also analyzed using Mean Absolute Error (MAE) with formula (2).

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \qquad (2)$$

In this formula, $y_i$ is the value of the manual verification result for $i$, while $\hat{y}_i$ is the value obtained from the system for data $i$. The value $n$ indicates the total number of test data or samples used in the calculation. A smaller MAE value indicates a lower level of system error, thus describing high accuracy in monitoring the validity period of domains and SSL certificates. The accuracy calculation results for each domain and SSL certificate sample are presented in Table 3.

Table 3. Comparison of System Results and Manual Verification for Each Sample

| Domain SSL | System Value | Manual Value | Accuracy |
|---|---|---|---|
| Domain 1 | 30 | 30 | 100% |
| Domain 2 | 344 | 344 | 100% |
| Domain 3 | 85 | 85 | 100% |
| Domain 4 | 227 | 227 | 100% |
| Domain 5 | 295 | 295 | 100% |
| SSL 1 | 262 | 261 | 99.62% |
| SSL 2 | 144 | 143 | 99.30% |
| SSL 3 | 142 | 141 | 99.29% |
| SSL 4 | 158 | 157 | 99.36% |
| SSL 5 | 297 | 296 | 99.66% |

Based on the test results shown in Table 3, the system achieved an accuracy rate of 100% in the domain test, with an MAE value of 0 days, indicating that the system results were completely identical to manual verification. Meanwhile, in the SSL certificate test, an accuracy of 99.45% was obtained with an MAE value of 1.0 day, indicating that the average difference between the system results and the manual data was only about one day.

## RESULTS AND DISCUSSION

The monitoring system was successfully implemented and tested, proving that all functions work well in detecting and reporting the validity period of domains and SSL certificates. The main result of this research is a centralized dashboard in Grafana, shown in Image 12. The dashboard visually displays the remaining active period of each digital asset in days. The use of thresholds on the dashboard allows for quick identification of asset conditions through color markers. Green indicates a long validity period (more than 60 days), yellow indicates an early warning for assets with a validity period of less than 60 days, while red indicates assets with a critical validity period of less than 30 days.

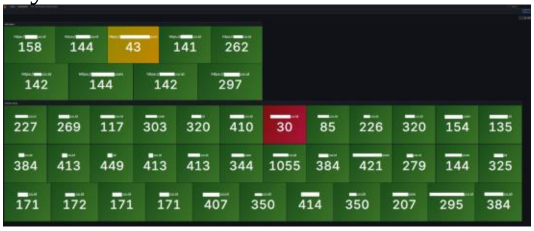
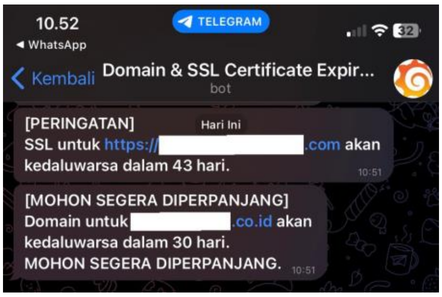


Image 12. Domain Dashboard and SSL Certificates

In addition to displaying visualizations, this system has also been successfully integrated with the Telegram Bot API to provide real-time notifications. When a domain or SSL certificate enters a predefined warning period, the system automatically sends a message to Telegram. When the remaining validity period is less than 60 days, the system provides an early warning so that the renewal process can be scheduled sooner. Meanwhile, when the remaining validity period reaches below 30 days, the system sends a high-priority warning as a sign that the domain or SSL certificate must be renewed immediately to avoid service disruption. Image 13 shows the warning notification received, containing complete information about the asset type (Domain or SSL Certificate), asset name, and remaining validity period.



[PERINGATAN]
SSL untuk https://[redacted].com akan kedaluwarsa dalam 43 hari. 10:51

[MOHON SEGERA DIPERPANJANG]
Domain untuk [redacted].co.id akan kedaluwarsa dalam 30 hari.
MOHON SEGERA DIPERPANJANG. 10:51

Image 13. Alert notifications via Telegram

The results of this study clearly show that the monitoring process, which was originally carried out manually and reactively, has become an automated and preventive system. The success of this automation, as shown in Image 12 and reinforced by the early warning notification in Image 13, directly addresses the challenge of managing the certificate lifecycle, which is prone to human error [3]. The implementation of the Prometheus and Grafana technology stack has proven to be effective not only as a performance monitoring tool, but also as a foundation for preventive security [1]. The results of this study reinforce the findings revealed in previous studies, which show that the combination of these two tools is capable of providing comprehensive visibility into IT infrastructure intuitively and in real time [5]. While many previous studies have focused on monitoring performance metrics such as CPU utilization, memory, and throughput [6], This study emphasizes metrics that directly impact public availability and trust, namely the validity of digital assets. The use of exporter to monitor HTTPS endpoints and Domain Exporter is considered more appropriate and relevant to the objectives of this study than the use of Node Exporter, which has a more general scope [12]. In addition, the successful integration of real-time notifications into Telegram, as shown in Image 13, supports other research findings that confirm that rapid alert mechanisms are an important factor in accelerating responses to incidents [4]. Through the implementation of this system, PT XYZ can proactively manage its digital assets, prevent the recurrence of incidents due to negligence, and improve the security and reliability of its services in the public eye.

## CONCLUSION

This research successfully designed and implemented an automated monitoring system using Grafana and Prometheus to proactively prevent service disruptions from expired domains and SSL certificates, transforming the previous reactive, manual process into a preventive security framework. The primary contribution to knowledge is the practical application of a common monitoring stack for the often-overlooked operational risk of digital asset lifecycle management, moving beyond simple performance monitoring. The system's scientific justification stems from its use of specialized exporters to gather validity data accurately, which fundamentally eliminates the human error inherent in manual checks. This system has broad application potential for any organization seeking to maintain operational continuity and secure its digital services.

## BIBLIOGRAPHY

[1] A. Setiawan, M. A. Satrio, I. Madani, R. D. Rachmat, and S. H. Sukma, "Meningkatkan Keamanan Sertifikat Digital dengan Pengaktifan HTTPS," *J. Internet Softw. Eng.*, vol. 1, no. 4, p. 9, Aug. 2024, doi: 10.47134/pjise.v1i4.3170.

[2] X. Wang, B. Wan, W. Zhou, H. Niu, and S. Feng, "Research on technical scheme for multi type load resource information access," *J. Phys. Conf. Ser.*, vol. 2189, no. 1, 2022, doi: 10.1088/1742-6596/2189/1/012030.

[3] P. S. Yadav, "Automation of Digital Certificate Lifecycle: Improving Efficiency and Security in IT Systems," *J. Math. Comput. Appl.*, vol. 2023, no. October 2023, pp. 1–4, 2022, doi: 10.47363/jmca/2023(2)e107.

[4] Rahmatia, N. Huda, and Nurhayati, "Analisis Rasio Likuiditas pada PT. Surya Citra Media, Tbk," *Ekopedia J. Ilm. Ekon.*, vol. 1, no. 2, pp. 210–221, 2025, [Online]. Available: https://doi.org/10.63822/mmj6cp9 5

[5] D. Rahman, H. Amnur, and I. Rahmayuni, "Monitoring Server dengan Prometheus dan Grafana serta Notifikasi Telegram," *JITSI J. Ilm. Teknol. Sist. Inf.*, vol. 1, no. 4, pp. 133–138, 2020, doi: 10.62527/jitsi.1.4.19.

[6] A. Lubis, A. M. Ghiffari, and S. Wahyuni, "Enhancing Network Performance Visibility with Grafana and Prometheus: A Case Study at P.T. Nata Digital Solution," *Int. Conf. Artif. Intell. Navig. Eng. Aviat. Technol.*, vol. 1, no. 1, pp. 305–309, 2024.

[7] M. D. Elradi, "Prometheus & Grafana: A Metrics-focused Monitoring Stack," *J. Comput. Allied Intell.*, vol. 3, no. 3, pp. 28–39, 2025, doi: 10.69996/jcai.2025015.

[8] P. B.C., H. Maddirala, and S. M., "Implementing an effective Infrastructure Monitoring Solution with Prometheus and Grafana," *Int. J. Comput. Appl.*, vol. 186, no. 38, pp. 7–15, 2024, doi: 10.5120/ijca2024923873.

[9] P. Fajar Malik and B. Parulian Josaphat, "Desain dan Implementasi Sistem Monitoring Jaringan Menggunakan Zabbix dan Telegram (Studi Kasus di Data Center BPS) (Design and Implementation of Network Monitoring System Using Zabbix and Telegram (Case Study at BPS Data Center))," *Semin. Nas. Off. Stat. 2024*, pp. 711–722, 2024.

[10] E. Pentanugraha, A. S. Saragih, and E. Christian, "Analisis Kinerja Load Balancing Webserver Menggunakan Haproxy Terintegrasi Dengan Grafana Sebagai Monitoring Dan Notifikasi Telegram," *J. Inf. Technol. Comput. Sci.*, vol. 4, no.

1, pp. 68–80, 2024, doi: 10.47111/jointecoms.v4i1.13191.

[11] Ramadoni, M. Z. Amirudin, Rifki Fahmi, Ema Utami, and M. S. Mustafa, "Evaluasi Penggunaan Prometheus dan Grafana Untuk Monitoring Database Mongodb," *J. Inform. Polinema*, vol. 7, no. 2, pp. 43–50, 2021, doi: 10.33795/jip.v7i2.530.

[12] E. Marpanaji, M. F. Rafi, and S. Kusumawardhani, "Experimental Study of High Availability Cloud Learning Management System and Monitoring System Based on Grafana , Prometheus and Telegram," vol. 1, no. 1, pp. 66–78, 2026.

[13] J. P. Masyarakat, "Sultra," vol. 2, pp. 114–122, 2025.

[14] B. Rasyidi and F. Pratama, "Sistem Monitoring Server di PT. XYZ Media Indonesia Berbasis Grafana dan Prometheus," *MALCOM Indones. J. Mach. Learn. Comput. Sci.*, vol. 4, no. 4, pp. 1456–1465, 2024, doi: 10.57152/malcom.v4i4.1546.

[15] S. STMIK Royal, "Implementasi Ssl Untuk Pencegahan Man in the Middle Attack Pada Ftp Server," *J. Sci. Soc. Res.*, vol. 4307, no. 1, pp. 28–33, 2021, [Online]. Available: http://jurnal.goretanpena.com/index.php/JSSR

[16] M. Bajpai, "Automating Monitoring and Incident Management with Prometheus, Grafana, and Google Cloud Pub/Sub," *Int. J. Sci. Res.*, vol. 11, no. 1, pp. 1673–1675, 2022, doi: 10.21275/sr24829151754.