**STANDAR ARSITEKTUR KEAMANAN APLIKASI BERBASIS WEB**

**{{nama organisasi}}**

**{{logo organisasi}}**

|  |  |
| --- | --- |
| No Dokumen | : 0.1 (Draft) |
| Revisi | : 00 |
| Tanggal Terbit | : |

|  |  |  |
| --- | --- | --- |
| Disusun  oleh : | Diketetahui  oleh: | Disetujui / Disahkan  oleh : |
|  |  |  |
| **NN**  NN | **NN**  NN | **NN**  NN |

**Riwayat Dokumen**

|  |  |  |
| --- | --- | --- |
| **Versi** | **Tanggal** | **Perubahan** |
| 0.1 (Draft) | 28/02/2023 | Rilis draft awal dengan hanya menyalin sepenuhnya konten dari **PEDOMAN TATA KELOLA KEAMANAN APLIKASI BERBASIS WEB - BSSN** |
| … | … | … |
|  |  |  |

1. **TUJUAN**
   1. Untuk memberikan arahan/petunjuk dan memastikan bahwa:
      1. Setiap perancangan sistem Aplikasi Berbasis Web memperhatikan kaidah – kaidah aristektur sistem
      2. Adanya pemisahan tiap komponen/layer aplikasi Aplikasi Berbasis Web sehingga:
         1. Dicapai optimalisasi kinerja yang memungkinkan tiap komponen bekerja independen dan memudahkan mitigasi masalah.
         2. Ketika terjadi insiden, roll back dapat dilakukan dengan cepat.
         3. Menciptakan *service staging perimeter protection* yang didukung *access control management*

agar mencegah pihak yang tidak berhak untuk masuk ke dalam sistem.

* 1. Untuk memastikan tingkat ketersediaan (*availability*) sistem ketika terjadi gangguan sesuai dengan tingkat kompleksitas Aplikasi Berbasis Web

1. **RUANG LINGKUP**
   1. Arsitektur sistem web untuk: web statis, web dinamis, dan web dinamis dengan aplikasi transaksional
   2. Penentuan Kebutuhan Sumber Daya Komputasi Web
2. **ISTILAH DAN DEFINISI**

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | Open Web Server | : | Web Server berbasis free (Terbuka) untuk OS Unix, Linux, BSD, dan Windows. Open Web Server merupakan alternatif pilihan disamping webserver yang sudah terkenal semacam apache / httpd. Open Web Server didesain untuk keamanan, kecepatan, fleksibel dan memenuhi Standar international serta bisa  diaplikasikan kedalam mesin produksi. |
| 2. | Cache/Accelerator WAF/Proxy | : | Cache adalah proses penyimpanan sementara data atau halaman HTML dan gambar sebuah Aplikasi Berbasis Web untuk mengurangi penggunaan bandwidth dan loading server. Secara sederhana, cache adalah teknologi yang membantu menampilkan  halaman Aplikasi Berbasis Web lebih cepat. |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | *Web Accelerator* adalah proxy server yang bertujuan untuk mempercepat akses pada Aplikasi Berbasis Web. Web Accelerator bisa diinstall di disisi klien atau disisi server  Teknik yang digunakan pada *web accelerator* adalah   * Cache & Prefect yaitu data yang baru atau sering diakses disimpan pada tempat penyimpanan sementara baik berupa RAM atau disk kemudian disajikan kembali tanpa perlu mengambil dari server utama . * Kompresi data dimana ukuran data diperkecil agar bisa lebih cepat. * Optimize yaitu kode dioptimasi sehingga size atau ukuran menjadi lebih kecil.   [Proxy](https://www.niagahoster.co.id/blog/apa-itu-proxy/) adalah suatu sistem yang memungkinkan kita untuk bisa mengakses jaringan internet menggunakan IP yang berbeda dengan yang diterima oleh perangkat.  Reverse proxy adalah salah satu jenis dari proxy, yang digunakan sebagai perantara antara client dengan web server. Reverse proxy dapat menghandle beberapa web server. Adapun cara kerjanya adalah client akan melakukan akses terhadap sebuah URL maka secara otomatis client akan melakukan request terlebih dahulu ke reverse proxy akan tetapi seolah - olah client melakukan request langsung ke web server. Setelah menerima request dari client, maka reverse proxy akan  meneruskan request tersebut ke web server yang dituju |
| 3. | Instansi | : | Kementerian/Lembaga, Instansi pusat atau daerah. |
| 4. | *Password* | : | Kata sandi yang digunakan bersamaan dengan *username* (*sign on/sign in/log-on/log-in*) oleh pemilik yang sah sebelum melakukan koneksi/akses ke sistem  komputer. |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | *Patch* | : | Rutin program atau sekumpulan kecil instruksi yang biasanya dibuat sebagai solusi sementara untuk mengatasi atau memperbaiki permasalahan (*bugs*) pada program komputer dan sering dibuat dalam bentuk *’object code’* yang disisipkan ke dalam program yang  akan dieksekusi. |
| 6. | Beban kerja suatu sistem |  | Himpunan semua input yang diterima sistem dari lingkungannya, selama periode waktu tertentu. Atau banyaknya sumber daya yang harus disediakan untuk melayani setiap request (proses) dan akses. Setiap sistem adalah unik dan akan memiliki karakteristik  beban kerja yang berbeda |
| 7. | *Content Delivery Network* (CDN) |  | adalah kumpulan dari server global yang terletak di beberapa data center dan tersebar di berbagai negara. Jaringan ini berfungsi untuk mengirimkan konten dari server ke suatu Aplikasi Berbasis Web.  Yang dilakukan oleh CDN adalah meningkatkan kecepatan pengiriman data melalui jaringan server kepada visitor dari lokasi terdekat yang paling  memungkinkan |
| 8. | *Content Management System (CMS)* |  | adalah sebuah perangkat lunak atau sistem yang mengatur konten pada situs web |
| 9. | *Virtual Private Server/Virtual Dedicated Server* |  | adalah sebuah server yang dibagi menjadi beberapa virtual server yang dapat diinstall OS dan berbagai aplikasi nya sendiri. VPS atau VDS itu sendiri merupakan teknologi yang memungkinkan sebuah komputer (server) dengan kapasitas sumber daya hardware yang sangat besar dapat dibagi-bagi menjadi beberapa virtual komputer yang mandiri.  VPS dapat berjalan layaknya sebuah Dedicated Server dan juga dapat diinstall sistem operasi (OS) tersendiri serta dapat mengatur virtual komputernya tanpa  mengganggu virtual komputer yang lain |

|  |  |  |
| --- | --- | --- |
| 10. | *Load Balancer* | adalah perangkat/tools yang berfungsi untuk mendistribusikan beban trafik pada dua atau lebih jalur koneksi secara seimbang, agar trafik dapat berjalan optimal, memaksimalkan throughput, memperkecil waktu tanggap dan menghindari overload pada salah satu jalur koneksi. Load balancer digunakan pada saat sebuah server telah memiliki jumlah user yang telah melebihi maksimal kapasitasnya. Load balancer juga mendistribusikan beban kerja secara merata di dua atau lebih komputer, link jaringan, CPU, hard drive, atau sumber daya lainnya, untuk mendapatkan pemanfaatan  sumber daya yang optimal. |

1. **REFERENSI**
   1. OWASP: Map Application Architecture (OTG-INFO-010)
   2. Standar Keamanan Informasi (ISO 27001:2013 – ISMS)
2. **STANDAR**
   1. **Arsitektur Aplikasi Web**
      1. Dalam merancang Arsitektur Web harus melihat jenis web yang akan dibangun yaitu:
         1. Web statis (non user generated content – non UGC)
         2. Web Dinamik
         3. Web Dinamik dengan aplikasi transaksi
      2. Rancangan Arsitektur Web Statis minimal menggunakan *Virtual Private Server* dengan persyaratan keamanannya sebagai berikut:
         1. Menggunakan Secure Shell (SSH) untuk masuk ke dalam server
         2. Mengubah port untuk login ke SSH
         3. Menggunakan password yang kompleks dan mengubahnya secara rutin
         4. Menonaktifkan akun root
         5. Menjaga update keamanan terbaru
         6. Menghindari mengunduh perangkat lunak kecuali dari sumber yang terpercaya
         7. Menonaktifkan port network yang tidak terpakai
         8. Menggunakan enkripsi GnuPG
         9. Mengkonfigurasi firewall
         10. Menggunakan SFTP di samping FTP
         11. Membuat folder/boot menjadi read-only
         12. Mengaktifkan update otomatis CMS
         13. Menginstall anti-malware/antivirus
         14. Memblokir akses anonymous ke FTP
         15. Menginstall rootkit scanner
      3. Standar arsitektur diatas secara teknis dapat menggunakan tools yang berbasis open source atau komersial.
      4. Rancangan Arsitektur Web Statis & Dinamis berbasis open source harus tetap memperhatikan tingkat skalabilitas, fleksibilitas, dan mudah dikembangkan tingkat keamanannya dengan arsitektur minimal sebagai berikut:
         1. Redundansi Web Server dan virtualisasi serta load balancer open source (Linux Virtual Server)
         2. Web Server Utama menggunakan User Management Server, Log Server dan Software VPN yang open source

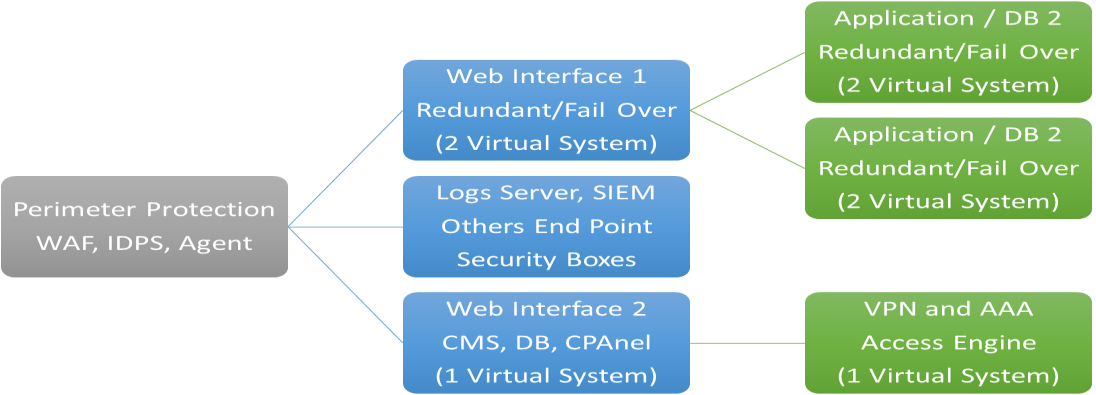
**Gambar 3.1 Desain Arsitektur Web Server**

Diagram

Description automatically generated

1. Rancangan Web dengan Aplikasi Transaksi harus menerapkan model pemisahan tugas di setiap komponen sistem atau fitur yang disediakan yaitu:
   1. Front End – UI/UX,
   2. Proxy/Accelerator/Load Balancer
   3. Web Services,
   4. Application,
   5. Database.
2. Gambaran rancangan arsitektur Aplikasi Berbasis Web dengan aplikasi dapat digambarkan sebagai berikut:

**Gambar 3.2 Desain Arsitektur Layanan Aplikasi Berbasis Web**



1. Arsitektur Layanan Aplikasi Berbasis Web meningkatkan kinerja HTTP reverse proxy, selain sebagai penyekat keamanan juga dapat berfungsi untuk mempercepat pengiriman konten Aplikasi Berbasis Web melalui *caching* terintegrasi (*accelerator),* melalui teknik *stream splitting* atau agregasi dengan kerjasama sistem *load balancer* dan kontrol bandwidth.
2. Konfigurasi fine tuning akan meningkatkan layanan Aplikasi Berbasis Web dan keamanannya dengan menerapkan otentikasi pengguna dan SSL tunnel. Jika diperlukan dapat ditambahkan modul kompresi konten Aplikasi Berbasis Websehingga beban kerja server dapat berkurang.
3. Arsitektur Model Anycast dengan menerapkan Content Delivery Network (CDN)

Bagi instansi-instansi tertentu yang aplikasi Aplikasi Berbasis Webnya dalam satu waktu membutuhkan tingkat ketersediaan yang sangat tinggi (Ditjen Pajak: e-filing di bulan Maret & April, Komisi Pemilihan

Umum (KPU) ketika masa pemilu, BKN ketika penerimaan CPNS, Kemendikbud & Dinas Pendidikan Daerah ketika penerimaan siswa baru dengan zonasi) dapat menerapkan:

1. Desain Arsitektur *Web Anycast/ High Availability Architecture* (HA). Suatu CDN (*Content Delivery Network*) dibangun menggunakan jaringan Anycast. Satu IP yang sama diterapkan pada semua host (mirror) CDN yang berada pada sejumlah network yang berbeda. Kemudian keberadaan IP ini di-broadcast melalui tabel BGP routing di masing-masing upstream network di bawah ASN yang sama. Sehingga IP tersebut akan dapat dikenali aktif berada di sejumlah network yang berbeda sekaligus.
   1. Setiap pengguna di wilayah yang berbeda memungkinkan dilayani oleh server CDN terdekat melalui jalur rute yang terbaik. Misalnya, pengguna di AS atau UE atau Asia akan dilayani oleh server CDN terdekat atau yang paling cepat.
   2. Jika salah satu server CDN tidak tersedia atau penuh – pengguna akan diarahkan secara otomatis (fail over) ke CDN lain yang terdekat dan masih tersedia (kapasitasnya), melalui perhitungan jalur rute yang terbaik. Sehingga Aplikasi Berbasis Webtersebut akan selalu aktif dan dapat diakses. Secara tidak langsung akan meningkatkan redundansi, kapasitas layanan dan kecepatan akses.
   3. Pada saat peak, CDN akan mendistribusikan akses secara merata ke banyak host CDN dengan teknik atau menggunakan modul load balancer, sehingga akan lebih efisien. Model Ini juga bermanfaat untuk DDOS karena setiap paket data dipecah ke tempat yang berbeda.
2. Arsitektur ini juga harus mudah untuk direplikasi, sehingga ketika diperlukan pengembangan –menerapkan cluster atau kapabilitas redundansi berbasis anycast, dapat dilakukan dengan cepat. Model Web Anycast memungkinkan efisiensi skenario redundansi dengan menempatkan duplikat sistem (*mirror*) yang identik di sejumlah network yang berbeda sekaligus sebagai pembagi beban (*load balancer*) sehingga dapat menghindari masalah *single point of error/failure (SPOF).*
3. Instansi dapat menggunakan CDN gratis atau berbayar. Berikut gambaran arsitektur model Web Anycast:

**Gambar 3.3. Desain Arsitektur Web Anycast**

Graphical user interface, diagram

Description automatically generated

* 1. **Penentuan Kebutuhan Sumber Daya Komputasi Aplikasi Berbasis Web**
     1. Suatu layanan Aplikasi Berbasis Web harus direncanakan sesuai dengan kebutuhan sumber daya komputasi agar menjamin kualitas proses dan melayani akses di saat beban terus meningkat atau ketika menghadapi berbagai kondisi beban rendah (*low*), tinggi (*high*), puncak (*peak*), berlebih (*over load*).
     2. Dalam menentukan kebutuhan sumber daya komputasi, instansi harus menyusun perencanaan kapasitas, berupa:
        1. Model biaya yang digunakan untuk menentukan biaya konfigurasi server termasuk peripheral, akses, lisensi, perawatan, personel dan pelatihan.
        2. Model beban kerja, yang membagi beban kerja berdasarkan jenis konten, menghitung intensitas tiap request.
        3. Model kinerja digunakan untuk menghitung metrik kinerja seperti *response time, error rate dan throughput serta availability*.
        4. Dan gabungan dari ketiga model ini menghasilkan rasio *cost/performance.*
     3. Tahapan dalam perencanaan kapasitas meliputi:
        1. Memahami Lingkungan:
           + Menentukan jenis situs web yang dijalankan
           + Penggunaan Aplikasi Berbasis Web(untuk konsumsi internal atau publik)
           + Jenis server yang digunakan (Web, Database, Application, User Management (AAA))
           + Jenis konektivitas (dedicated, shared, Metronet, VSAT, dll)
           + ISP beserta SLA yang diberikan
           + Sistem Operasi yang digunakan server
           + Aplikasi – aplikasi pendukung yang diperlukan
           + Tingkat layanan (SLA) yang dijanjikan/diberikan kepada pengguna
           + Ketersediaan fasilitas back up
           + Ketersediaan roll back
           + Ketersediaan SOP keamanan (terutama penanganan insiden)
        2. Karakterisasi Beban Kerja Untuk Aplikasi Berbasis Web:
           + Karakteristik Situs berita statis akan lebih banyak membutuhkan sumber daya komputasi untuk melayani jumlah akses yang banyak sekaligus pada satu saat dan kapasitas jaringan yang dedicated.
           + Karakteristik situs transaksi online bisa membutuhkan sumber daya penyimpanan dan prosesor (CPU) dalam jumlah yang banyak.
           + Beban Kerja Server: semua transaksi (pencarian, jelajah, dan pembaruan) yang diproses oleh server dalam satu periode.

Misalnya server memproses selama 30 menit (atau 1800 detik) dengan 72.000 transaksi (beban kerja) diselesaikan. Karakteristik beban kerja – diwakili intensitas beban (access rate = 72.000 / 1800 = 40 transaksi per detik), jenisnya (query, browsing, dll.) dan permintaan sumber daya dari masing-masing 72.000 transaksi = waktu CPU rata-rata per transaksi dan jumlah rerata operasi Input / Output per transaksi.

* + - * + Beban Kerja Jaringan (Network): kebutuhan kapasitas jaringan, dihitung menggunakan kalkulasi sederhana, seperti perkalian rerata jumlah request per detik dengan rerata ukuran file/dokumen yang diakses. Masih ditambahkan beban overhead protocol sebesar 20 persen. Sementara latency, tidak diperhitungkan. Sehingga bila ada 100 permintaan per detik untuk mengakses file sebesar 10 KB, dibutuhkan bandwidth 100 x (10 x 1024 x 8 bit) = 8.192.000 bps atau 8,192 Mbps.
        + Tingkat latency harus diperhitungkan di dalam analisis karakteristik dan beban kerja. Semakin kompleks interaksi antara client dengan server, pada dasarnya akan menambah parameter perhitungan. Misalnya ketika beban kerja meningkat dan terjadi antrian,
        + Setiap pengembang Aplikasi Berbasis Webharus dapat menentukan berapa ukuran – misalnya file – yang paling efisien untuk diproses dengan menggunakan sumber daya yang paling minimal.
        + Untuk meningkatkan kapasitas komputasi dapat berupa kapsitas server yang perlu ditambah menambahkan mirror load balancing.
    1. Metrik yang digunakan untuk mengukur kinerja antara lain *response time throughput, error rate, availability*. Misalnya, waktu respons untuk halaman statis tidak boleh melebihi 0,5 detik dan waktu query database tidak melebihi 1 detik per 6 HTTP request. Jika beban berlipat gKita menjadi 12 HTTP request per detik, kinerja tidak turun di bawah 30 persen dan maksimal penolakan request hanya 1 persen saat peak, utilisasi harus di bawah 80%.
    2. Untuk mengukur kinerja komputasi dapat menggunakan metodologi alternatif yaitu benchmarking, membandingkan kinerja 2 sistem yang berbeda. Matriks tolok ukur yang populer adalah WebStone dan SPECweb. Kedua aplikasi pengukuran kinerja ini mensimulasikan browser dan melakukan sejumlah request ke server sesuai karakteristik beban kerja yang ditentukan, menerima respons dari server, dan akan mencatat hasil pengukuran (response time, query, latency).
    3. Model untuk Kinerja Aplikasi Berbasis Web:
       1. Untuk membangun model kinerja Aplikasi Berbasis Web, adai dua opsi: model simulasi atau analitik. Model simulasi lebih memakan waktu untuk dijalankan dan memerlukan lebih banyak parameter daripada model analitik. Khusus untuk tujuan perencanaan kapasitas, biasanya lebih baik menggunakan model yang lebih cepat dan tingkat yang lebih tinggi, seperti model analitik.
       2. Model kinerja Aplikasi Berbasis Web, pada dasrnya menggunakan teori antrian yang diterapkan pada CPU, disk, LAN internal, router, dan jaringan. Hasil pengukuran ini bisa digunakan untuk prediksi kinerja setelah memperkirakan evolusi beban kerja. Misalnya, menggunakan model kinerja untuk memprediksi response time ketika request rate meningkat hingga sebesar 60 persen.
       3. Dalam teori antrian proses komputasi ini, salah satu faktor yang menentukan adalah waktu tunggu proses atau yang disebut dengan latency. Yaitu total waktu yang dihabiskan selama menunggu proses di CPU, RAM, disk, bus, LAN, router dan pengiriman data berlangsung. Contoh, satu request file gambar memerlukan waktu proses 320 ms (0,320 detik). Tingkat request adalah 1,5 per detik. Maka Udisk = 1,5 x 0,320 = 0,48 = 48 persen adalah tingkat efisiensi sumber daya harddisk, sedangkan latencynya adalah 0,32 / (1 - 0,48) = 0,615 detik.
       4. Dengan kata lain, proses request file gambar menghabiskan waktu 0,615 detik di disk drive. Karena 0,320 detik dihabiskan untuk proses di disk drive, sisanya 0,295 detik (0,615 - 0,320) dihabiskan untuk antrian disk atau latency. Dengan kata lain, pemanfaatan yang tinggi akan menyebabkan waktu tunggu yang tinggi juga. Namun pertumbuhan latency tidak selalu linier dengan tingkat pemanfaatannya. Misalnya, jika tingkat request meningkat 60 persen menjadi 2,4 request per detik, pemanfaatan disk drive tumbuh menjadi 0,768 (2,4 x 0,320) dan latency menjadi 0,320 / (1 - 0,768) = 1,38 detik, terjadi peningkatan 224 persen.
    4. Memprediksi Beban Kerja Aplikasi Berbasis Web melalui:
       1. Memperkirakan jumlah pengunjung Aplikasi Berbasis Web untuk merencanakan kapasitas yang lebih memadai
       2. Beban kerja yang diharapkan selama musim liburan
       3. Beban kerja ketika ada penambahan fitur dan layanan baru serta peningkatan volume transaksi.
       4. Metoda prediksi dapat kuantitatif atau kualitatif. Metoda kuantitatif sangat bergantung pada keberadaan data historis untuk memperkirakan nilai parameter beban kerja di masa depan. Sedangkan pendekatan kualitatif adalah proses subyektif, berdasarkan penilaian, intuisi, pendapat ahli, analogi sejarah, pengetahuan komersial, dan informasi terkait lainnya. Strategi peramalan harus menggabungkan teknik kuantitatif dan kualitatif.
    5. Basis metrik tingkat utilisasi yaitu request per second/minute (RPS/RPM) dimana 100 adalah peak dan 80 adalah maximum treshold (high) dan diantara 20-60 adalah (normal), sedangkan di bawah 20 dianggap beban rendah (low). Apabila kondisinya sering berada di atas maximum treshold (high), maka terjadi overload.
    6. Jika di bawah beban terendah (low), maka terjadi inefisiensi maka perencanaan kapasitas harus dikoreksi.
    7. Memaksimalkan konfigurasi Database untuk fitur akselerasi proses seperti tools indexing dan cache, sekalipun konsekuensinya, diperlukan tambahan RAM untuk menempatkan tabel index dan cache untuk mengurangi latency– teorinya, transfer rate I/O RAM lebih tinggi (cepat) dibanding bila tabel index dan cache ditempatkan di media penyimpan (disk) walaupun sudah SSD.
    8. Untuk menjaga kinerja dalam menghadapi latency, pengembang aplikasi Aplikasi Berbasis Web harus menggunakan metoda mengubah hasil query ke database – misalnya proses search data tertentu ke dalam bentuk teks statik dan dijadikan buffer. Ketika ada request ke obyek sama berulang, web interface mengarahkan langsung ke buffer bukan melakukan query ke DB. Sehingga, penggunaan CPU, RAM dapat dihemat. Berikut gambarannya:

**Gambar 3.4. Model Query Statik**

Graphical user interface, text, application, chat or text message

Description automatically generated

Untuk menjamin tabel Static Index tetap up to date, maka Web Interface harus dikonfigurasi melakukan request query langsung ke database secara periodik dengan memanfaatkan waktu off peak.