Top 10 Security Architecture Mistakes

(AKA Anti-Patterns)

Harman Singh

www.thecyphere.com info@thecyphere.com

What is an anti-pattern?

- A flawed, repeated solution to a common problem.
- Coined by Andrew Koenig in response to Design Patterns.

Trust Levels: Low vs High-Side Systems

Systems aren't isolated – they're connected to various networks, and trust levels vary.

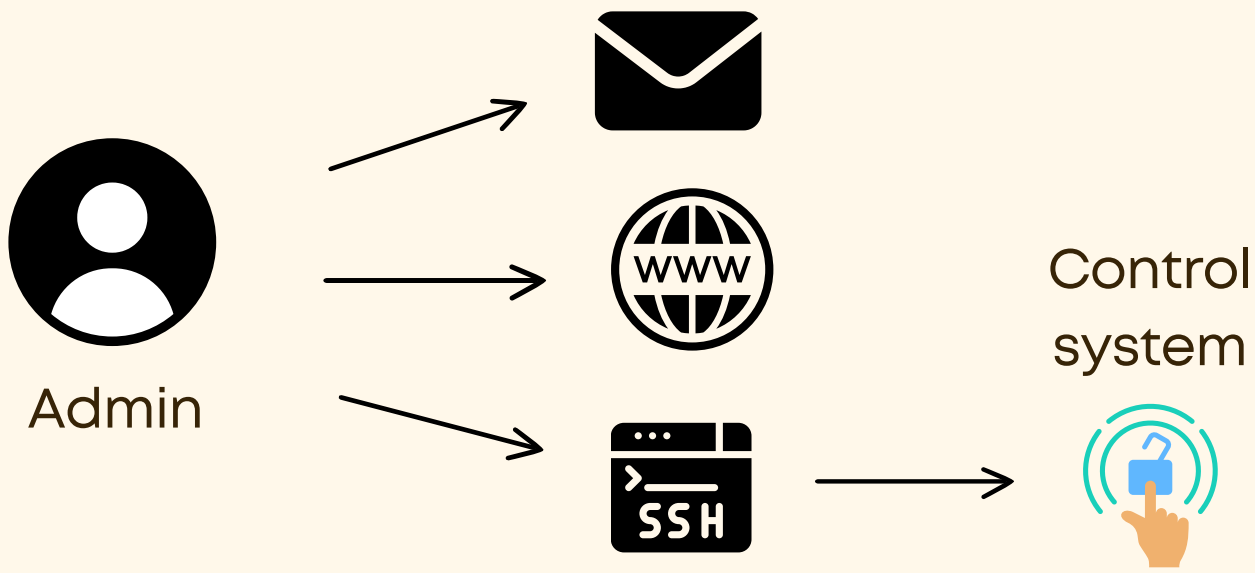
- **Low side (less trusted):** Lower confidence in integrity.
- **High side (more trusted):** Higher confidence in integrity.

Anti-Pattern 1: Browse-up for Administration

www.thecyphere.com info@thecyphere.com

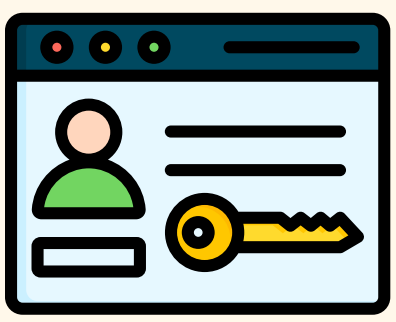
- **What is ‘Browse-Up’ Administration?**
  - Administering a high-trust system from a low-trust device.

- **How to identify?**
  - Look for remote desktop/shell from less trusted devices.
  - Unverified devices in remote support.
  - Admin tasks on web/email devices.



Real-world risks

Session hijacking



Credential theft

Lateral movement

Actionable advice

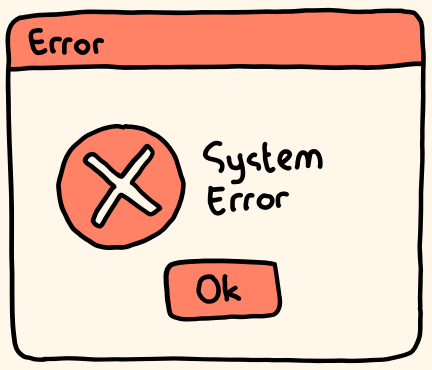
DOS	DON'TS
• Use high-trust devices for admin tasks. • Separate admin tasks from web/email activities. • Implement strict access control for admin accounts. • Regularly update admin credentials.	• Administer from untrusted devices. • Trust insecure jump boxes. • Rely only on 2FA. • Ignore session expiry or cached creds.

Anti-Pattern 2: Single Point of Failure

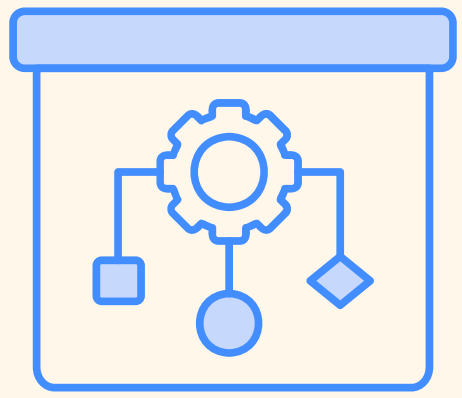
- **What is Single point of failure?**
  - Designing critical systems or components without redundancy, if that one piece breaks, everything else can too.

- **How to identify?**
  - No backups.
  - No failovers.
  - No plan B.
  - One failure takes down the whole system.

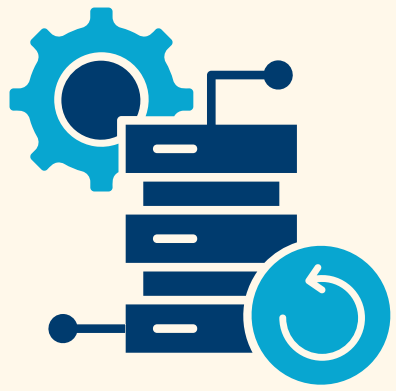
Real-world risks



Total outage/ High downtime



Bottlenecks that limit scalability



Poor disaster recovery

Actionable advice

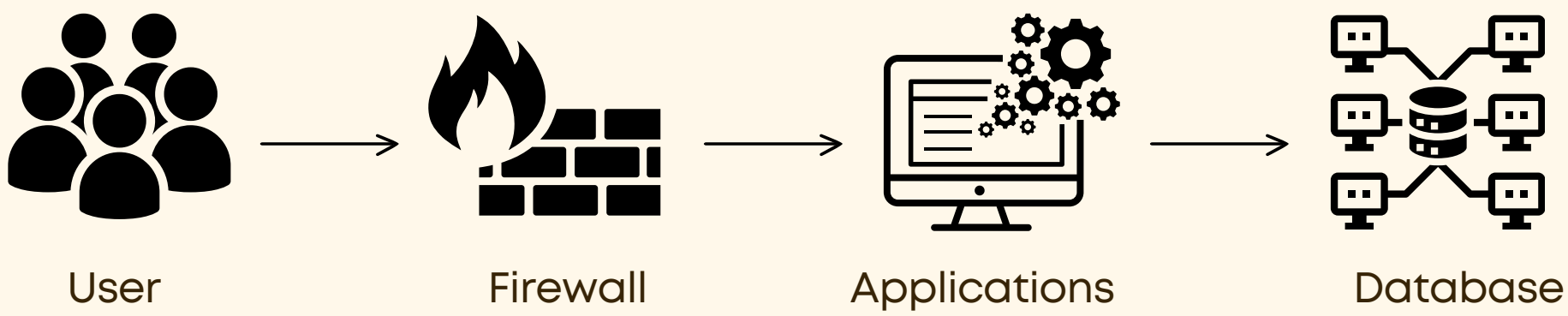
DOS	DON'TS
• Design with redundancy and failover in mind. • Test high-availability setups regularly. • Include resilience in your security architecture.	• Rely on one instance of critical infrastructure. • Assume uptime without validation. • Ignore business continuity plan.

Anti-Pattern 3: The Unpatchable System

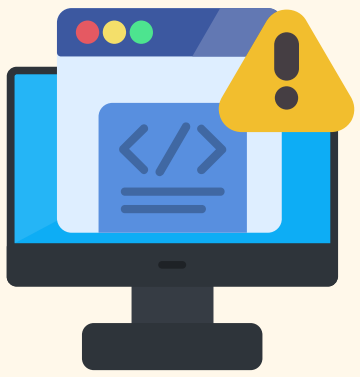
www.thecyphere.com info@thecyphere.com

- **What is an ‘un-patchable’ system?**
  - A system that cannot be patched due to 24/7 operational demands.

- **How to identify?**
  - Look for systems without redundancy that rely on all components being operational, preventing phased upgrades.

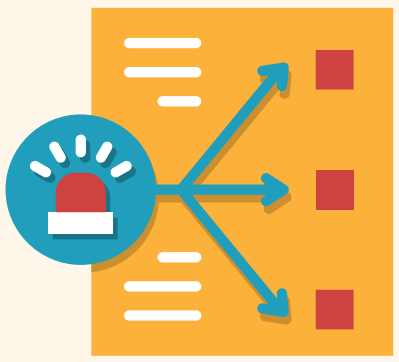


Real-world risks



Exploitable vulnerabilities

System outage



Costly incident response

Actionable advice

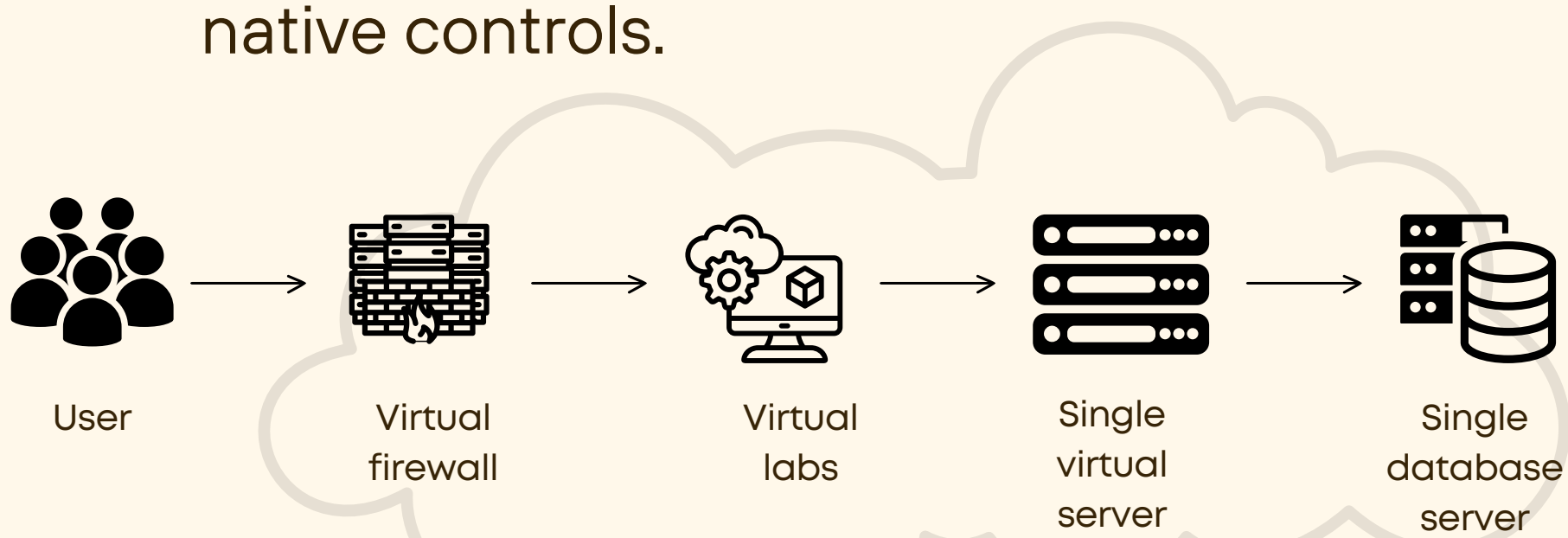
DOS	DON'TS
• Design for phased updates. • Use rolling upgrades. • Test updates in control. • Automate patching to reduce manual errors.	• Build without redundancy. • Delay patching. • Skip patch testing. • Rely on emergency fixes.

Anti-Pattern 4: Building an ‘On-Prem’ Solution in the Cloud

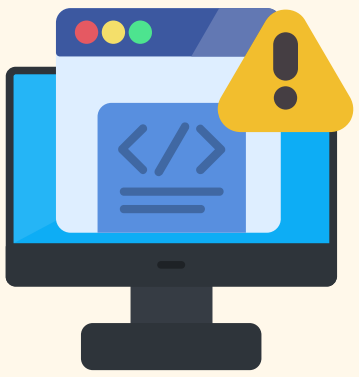
www.thecyphere.com info@thecyphere.com

- **What is building an ‘on-prem’ solution in the Cloud?**
  - Mimicking traditional on-prem infrastructure in the public cloud.

- **How to identify?**
  - Look for databases on compute instances
  - 24/7 running environments
  - Virtual appliances used instead of cloud-native controls.



Real-world risks



Security vulnerabilities

Lack of flexibility

Maintenance overhead

Actionable advice

DOS	DON'TS
• Leverage PaaS for easy management. • Focus on unique tasks. • Use cloud-native tools for security. • Automate patching with managed services.	• Treat IaaS like on-prem. • Ignore cloud-native services. • Keep environments running 24/7. • Over-manage virtual appliances.

Anti-Pattern 5: Security by Obscurity

- **What is security by obscurity??**
  - A flawed security practice aimed at hiding implementation details without secure approach towards design or build

- **How to identify?**
  - Understand whether your security relies on keeping things hidden such as
    - minimal documentations,
    - treating proprietary systems as protection.

Real-world risks

Unaddressed vulnerabilities

Reverse engineering

Lack of defence

Actionable advice

DOS	DON'TS
• Build systems on proven, transparent security principles and best practices. • Use open standards and peer-reviewed mechanisms. • Use zero-trust, access controls, secure software development lifecycle.	• Count on secrecy to keep attackers out. • Hide flaws behind proprietary or undocumented systems. • Rely on secrecy instead of secure coding and proper defences.

Anti-Pattern 6: Ignoring Threat Modelling

www.thecyphere.com info@thecyphere.com

- **What is ignoring threat modelling?**
  - Designing systems without systematically identifying, analysing, and mitigating potential threats during development.

- **How to identify?**
  - No documented threat assessments during SDLC; security is added as an afterthought.

Real-world risks

Missed vulnerablities

Costly last stage fixes

Weak by design

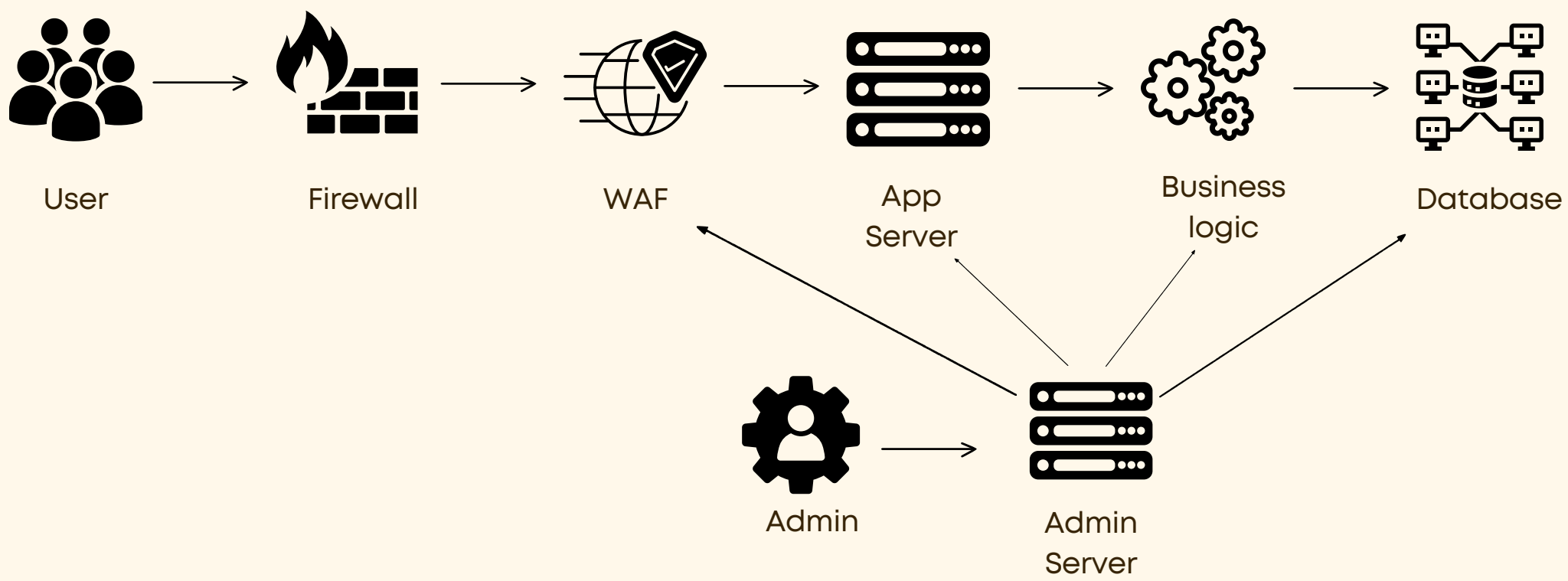
Actionable advice

DOS	DON'TS
• Integrate threat modelling from the start. • Use frameworks like STRIDE or PASTA. • Document and share findings across teams. • Regularly review and update the threat model. • Follow the "Assume Breach" principle.	• Rely only on automated tools for insights. • Ignore human factors (e.g., insider threats, social engineering). • Overcomplicate threat modeling. • Neglect risks from external dependencies. • Apply a one-size-fits-all approach.

Anti-Pattern 7: Management Bypass

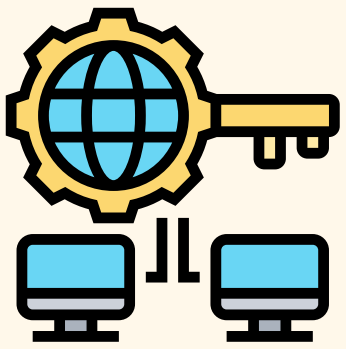
- **What is management bypass?**
  - When defences in the data plane can be bypassed through the management plane.

- **How to identify?**
  - Look for management interfaces from different layers sharing a single switch without separation.

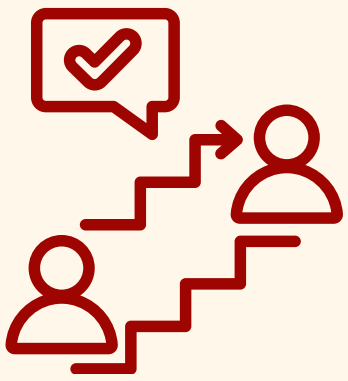


Real-world risks

Lateral movement



Critical system access



Privilege escalation

Actionable advice

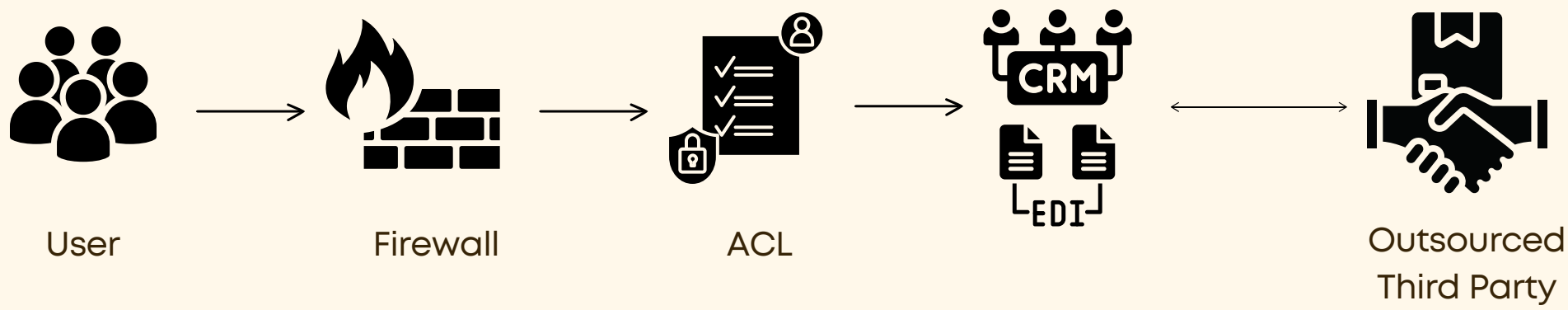
DOS	DON'TS
• Use trusted devices for management. • Separate credentials per trust layer. • Isolate management systems. • Use bastion hosts per trust boundary.	• Expose management interfaces on data plane network. • Overlook defence-in-depth for management plane. • Allow lateral movement between planes. • Use shared credentials across layers.

Anti-Pattern 8: Uncontrolled & Unobserved Third-Party Process

www.thecyphere.com info@thecyphere.com

- **What is uncontrolled third-party access?**
  - When a third party has unrestricted remote access without constraints or monitoring.

- **How to identify?**
  - Look for direct, unbroken connections in network diagrams indicating third-party relationships.



Real-world risks

Third-party breach impact

Supply chain attacks

Undetected malicious activity

Actionable advice

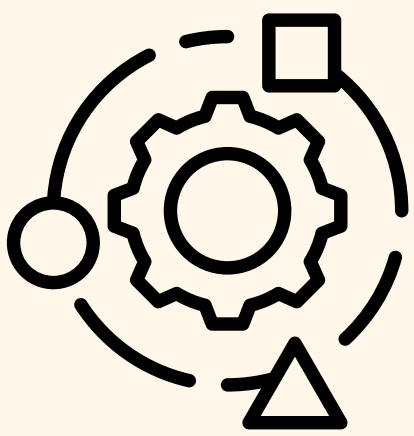
DOS	DON'TS
• Enforce least privilege access. • Track actions with audit logs. • Use MFA for remote users. • Isolate third-party access. • Apply just-in-time access.	• Avoid unrestricted access via bastion hosts. • Prevent shared credentials. • Don't rely on supplier's security alone. • Avoid shared system access for multiple third parties. • Don't keep persistent third-party access.

Anti-Pattern 9: Vendor Lock-In

- **What is vendor lock-in?**
  - Overly dependent on one vendor's proprietary tech.
  - No clear plan exists to switch away.

- **How to identify?**
  - Architecture tied to a single vendor.
  - Limited interoperability.
  - No exit or migration plan.

Real-world risks



Reduced flexibility

Increased cost

Vendor disruption

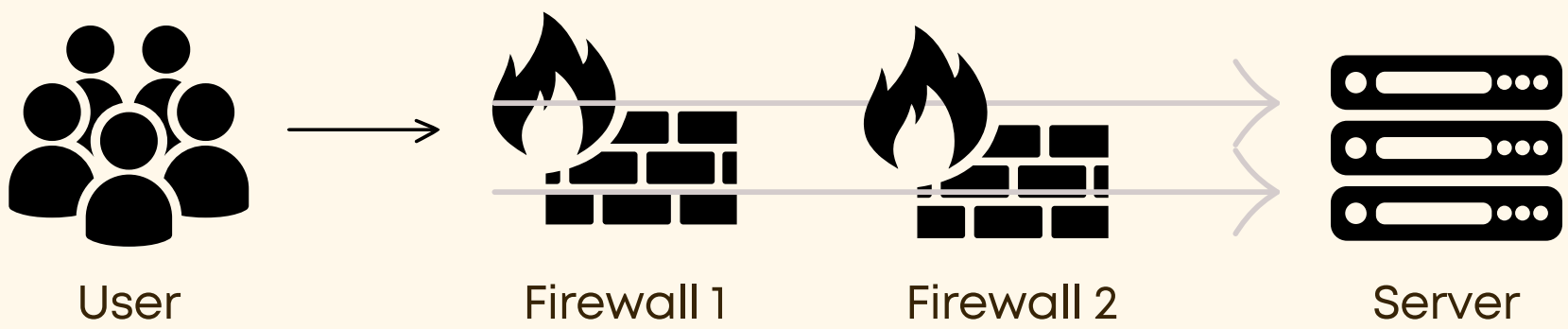
Actionable advice

DOS	DON'TS
• Be vendor-neutral as much as possible (e.g., multi-cloud) • Ensure compatibility with open standards (APIs are key). • Document migration strategies and test vendor-agnostic solutions.	• Rely on proprietary services without considering alternatives • Couple architecture to vendor-specific APIs • Depend on vendor solutions without exit plans.

Anti-Pattern 10: Back-to-Back Firewalls

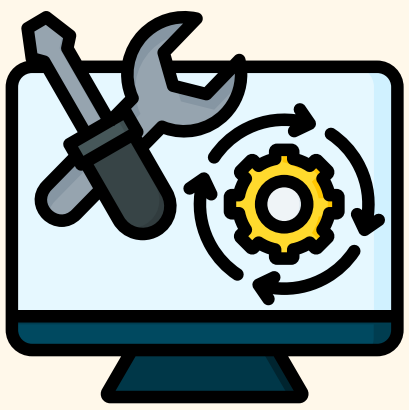
- **What is back-to-back firewalls?**
  - When two firewalls are placed in series, often from different vendors, to apply the same security controls.

- **How to identify?**
  - Look for two firewalls in series in a network architecture diagram.



Real-world risks

Weak exploit defenses



Slow Patching

Operational overhead

Actionable advice

DOS	DON'TS
• Use a single, well-maintained firewall. • Regualr upgrades, patches and maintenance • Restrict access and enforce strong authentication. • Keep firewall rules simple and documented.	• Rely on two firewalls for added security. • Rely on vendor diversification to mitigate firewall vulnerabilities. • Never expose management interfaces to untrusted networks. • Keep policy configurations simple to avoid errors.

Empower others with knowledge - follow & share this post!

www.thecyphere.com
info@thecyphere.com