LOCKHEED MARTIN

Defendable Architectures

Achieving Cyber Security by Designing for Intelligence Driven Defense®

Scott C. Fitch, Michael Muckin
Lockheed Martin Corporation

Abstract

Conventional approaches to security architecture focus on hardening systems against attack. The implicit expectation is that, once deployed, it is the system's responsibility to block all attacks against it. However, systems that rely strictly on hardening cannot learn from, or adapt themselves to, changes in attacker techniques, capabilities, and objectives over their lifecycle. Nor can they address the reality that attackers occasionally succeed. The ongoing efforts of sophisticated adversaries, including the Advanced Persistent Threat (APT), require systems that can be actively defended in an equally sophisticated manner. An intelligence-based model that builds an understanding of attackers from a variety of intelligence sources, including their interactions with the system itself, enables defenders to adapt to and anticipate changes in adversaries' attacks.

Defendable Architectures describe an alternative approach to system architecture by explicitly designing, implementing, and maintaining systems to support Intelligence Driven Defense® practices. The result is a virtuous cycle of greater visibility into systems to gather intelligence, faster translation of intelligence into defensive measures, and more effective deployment of those measures into the system's security controls. Moreover, threat intelligence can be leveraged when building systems to ensure their design is well adapted to current and emerging threats. This concept is also extended to the enterprise as a whole, describing how organizations can plan and deploy their systems and infrastructure with the Intelligence Driven Defense framework in mind.

Applying this approach, organizations can build systems that are resilient to cyber attacks, and create system designs that are resilient to changes in attackers' techniques and objectives. Defendable Architectures provide an approach for creating systems that can be defended against attack, survive compromise, and adapt to adversary changes.

Keywords: architecture, computer network defense, intelligence, APT, resilience

1 Introduction

Classical information security principles focus on hardening systems against potential attack. Traditionally, this is done by specifying security controls that will counter attacks the designer anticipates the system will encounter. Implicitly, once in operations, the system is expected to block any attacks fielded against it.

This approach does not reflect the reality of modern information systems defense. Systems cannot secure themselves. Nor can they be expected to block every attack, based on static controls designed years, months, or even weeks prior; systems cannot learn and adapt themselves to changes in adversary behavior over time. Succinctly stated, if an intelligent human is conducting an attack, intelligent humans must be directing the defense.

This paper describes an alternate approach that more accurately reflects how networked systems must be defended. Based on Defendable Architectures, systems explicitly designed to support Intelligence Driven Defense® techniques are better suited to withstand ongoing changes in adversaries' objectives and their tactics, techniques, and procedures (TTPs). Such systems better support their defense by exhibiting the characteristics of visibility, manageability, and survivability. The organizations that apply these architectures leverage threat intelligence throughout their design, development, and operations.

Furthermore, all systems exist in the context of their enterprise, interfacing systems, and neighboring systems. Understanding a system's context is critical to its design and defense. An enterprise can gather threat intelligence across its entire scope, and use that intelligence to plan and implement common security infrastructure to address adversary objectives and TTPs. In this way, Defendable Architectures are extended to the concept of a Defendable Enterprise.

There is growing recognition among organizations in the public and private sectors of the importance of resilience against network attacks [1] [2] [3] [4]. Cyber resilience is rooted in the acknowledgement that systems must continue to provide their intended services during attack and recovery. In this sense, resilience is the ability of a system to survive attack, detection, and recovery. Resiliency also represents the ability of a system's design to withstand changes in an attacker's approach and objectives; or the introduction of new attackers with different approaches and objectives. Organizations applying the Defendable Architecture approach are able to achieve both types of resiliency as an inherent characteristic of the design: systems are resilient to attacks, and designs are resilient to changes in attackers.

2 Leveraging System Knowledge and Threat Intelligence

Hutchins, Cloppert, and Amin described the essential function that intelligence plays in defending computer networks [5]. Key activities in Intelligence Driven Defense processes, such as analyzing attacks based on the Cyber Kill Chain® model, pivoting through historical data based on new intelligence, and adapting security controls to changing attacks, require human interaction and tradecraft to be successful.

By acknowledging the active role that human intelligence plays in defending systems and enterprises, organizations are able to deliver more effective security. Systems built on Defendable Architectures not only leverage defenders' knowledge and intelligence, but also that of designers, developers, testers, and administrators who support the system throughout its lifecycle. The phases represented by these roles – design, build, run, and defend – all offer unique opportunities to share and incorporate that knowledge into the system itself.

During **design**, engineers specify the concept, requirements, and design of the system. This is when many of the fundamental security characteristics of a system are determined. Defendable Architectures are distinguished from traditional security architectures by focusing not just on trying to design a *hardened* system, but by using threat intelligence and system threat analysis[1] to guide architecture decisions, and designing the system to support the needs of Intelligence Driven Defense practices. As described later in this paper, systems based on Defendable Architectures exhibit the characteristics of visibility, manageability, and survivability. During design, engineers also analyze available threat intelligence to determine which security controls will be most effective.

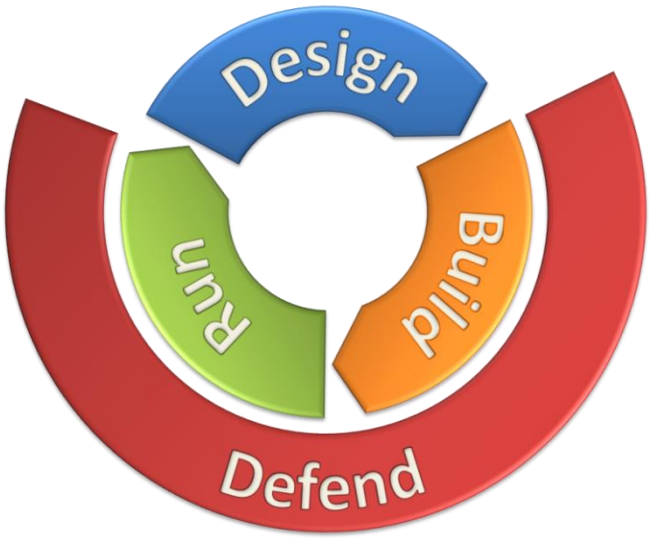

Figure 1: Lifecycle of Defendable Architectures

During the **build** phase, engineers implement system functionality and security controls through code and configuration. Testers verify the inherent security characteristics of the system (e.g., input validation) and efficacy of the security controls selected during design. A variety of test approaches are used, such as inspection of configuration settings, demonstration of security controls, static and dynamic vulnerability testing, and adversarial or penetration testing. The selection of appropriate test mechanisms is driven by the risk and relative importance of functions being tested.

During the **run** phase, administrators manage the system and end users utilize it. Administrators sustain the system over time, including operational needs such as patching and system administration. They also support developer and defender diagnostics of the system, and implement changes to security controls based on new indicators discovered by defenders.

During the **defend** phase, intelligence analysts generate intelligence through visibility into adversary activity, and detect and respond to attacks. This includes defending the production, test, and development environments of the system and its source code. Defenders translate indicators to protection and detection rules, and deploy those rules to actively defend systems. They also assess and communicate changes in attack vectors to designers, developers, and administrators.

Each of these phases, and the roles that support them, have important knowledge and intelligence to share and incorporate. These communications are summarized in the figure below, which indicates information transfer from a row's phase to a column's phase.

[1] In this paper, *threat intelligence* refers to knowledge about adversaries, their campaigns, objectives, and TTPs, typically gathered through Intelligence Driven Defense, intelligence sharing, and similar approaches (e.g., [5]). *System threats* refer to the undesirable events or conditions that can impact a given system or enterprise (e.g., [6]). These are separate but complementary concepts, as described in section 4.1. A means to distinguish them, and to clarify the overloaded term "threat," is necessary.

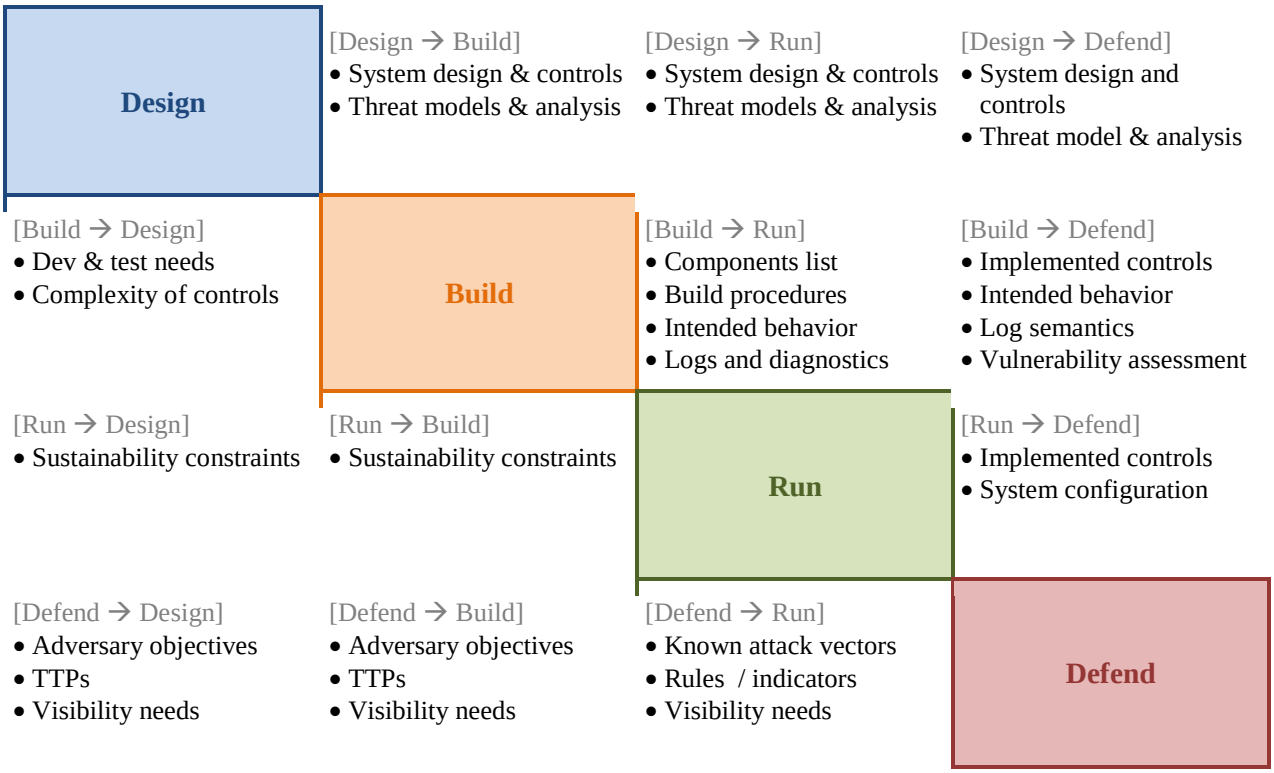

Figure 2: Sample of knowledge flows between lifecycle stages

While many of these communications are recognized (and sometimes practiced) by classical security engineering approaches, several are unique to the Defendable Architecture approach:

- Threat models & analysis
- System configuration and intended behavior
- Indicators of adversary activity
- Tactics, Techniques, and Procedures (TTPs)
- Adversary objectives

Highlighted in Figure 3, these communications enable an organization to continuously incorporate knowledge about a system and the threats against it into the way the system is designed, built, run, and defended.

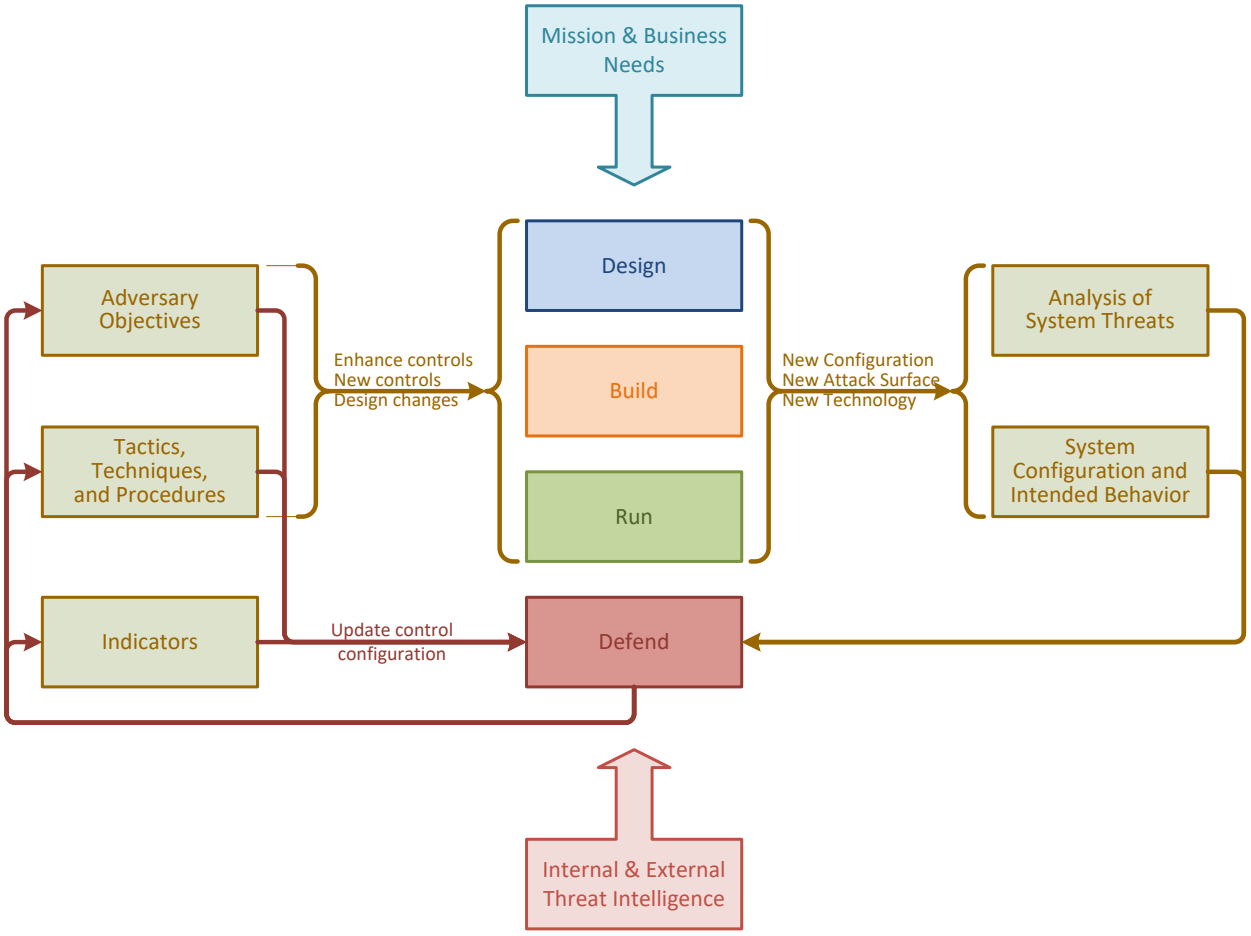

Figure 3: Cyclical knowledge flows of Defendable Architectures

An analysis of the threats against a system is generated while the system is being designed, developed, and modified. This analysis begins with identifying the important assets, such as the data and functionality within the system. System threats are the undesired events or conditions that would impact these assets [6]. As long as the assets exist, so do the threats. For example, a system's transactional data can be subjected to theft, tampering, loss of availability, or repudiation. Each of these threats has different potential impacts to the organization's mission and business needs. Designers and developers balance each of these threats based on their impact to the organization and the controls available to mitigate them. The result of this analysis is an understanding of the system's assets, the threats against them, and the chosen controls. Sharing the results of this analysis with administrators and defenders gives them important insight into which assets are most important to the system, and the purpose of the controls that were selected as a result.

In addition to threat analysis, the designers and developers specify the system configuration and intended behavior. This helps defenders understand the design and implementation of a system, essential during incident triage and recovery. It also helps defenders calibrate their detection mechanisms based on expected system behavior. For example, if administrator access is designed to only be allowed from a particular network path, then defenders know that any administration activity from other networks is cause for further investigation.

Defenders also have important knowledge to share with designers, developers, and administrators. Often collected together under the title *threat intelligence,* we identified three different abstractions. These

different levels of threat intelligence impact the design, implementation, and operations of a system in different ways.

At the most basic level, newly revealed indicators are used to develop rules that are applied to existing controls based on enterprise-level attack analysis and threat intelligence sharing. New indicators usually imply the adversary is changing parts of their toolkits or infrastructure. These changes are typically managed by defenders and added to configuration and rules in existing security controls by the appropriate security infrastructure administrators.

At a higher layer of abstraction, changes in adversaries' TTPs can necessitate implementing new types of controls to address new attack types. These often signal the existence of new adversary campaigns or shifts in the way adversaries conduct their attacks, such as changing from email to web-based attacks. Because an attacker incurs cost to change TTPs, these tend to be more durable detection and protection controls than individual indicators. While some changes in TTPs require infrastructure or system modifications, a flexible infrastructure allows defenders to adapt defenses to many changes in TTPs. Observed changes in TTPs can also result in deemphasizing some existing controls in order to reallocate operations and developer resources to new, more effective mitigations.

The final layer of abstraction of threat intelligence is used to understand the adversaries' objectives. Unlike discovery of indicators and some TTP changes, which can be addressed with greater agility, changes in adversary objectives primarily affects system design and controls selection. Comparing the adversaries' objectives with the confidentiality, integrity, and availability needs of the system allows designers to make appropriate design decisions about the system and to choose security controls that most effectively mitigate anticipated threats to the system. For example, a system that must withstand a distributed denial of service (DDOS) attack will be designed differently than one that is not intended to survive such an attack. It also allows developers, testers, and administrators to implement and sustain the necessary controls and choose appropriate test scenarios. Analyzing the security needs of the system in the context of current threat intelligence gives a more complete picture of a system's risk posture.

For organizations that have already deployed security infrastructure with Defendable Architecture in mind, defenders can respond rapidly and directly to new indicators and changes in TTPs. The fewer steps required to adjust controls based on new intelligence, the quicker an organization can react to its adversaries' changes. This is further described in section 5, The Defendable Enterprise.

These knowledge flows create a closed-loop virtuous cycle where threat intelligence influences system design and implementation, which in turn improves the operations and defense of the systems themselves. By sharing this knowledge across the roles that support a system throughout its lifecycle, the entire team is better able to defend the system. The result is a system that is resilient to attack, and a system architecture that is resilient to changing attackers.

3 Supporting Intelligence Driven Defense®

Leveraging the knowledge flows described above, organizations can field systems that are adapted to their mission needs and adversaries' capabilities and objectives. It also allows organizations to more effectively defend these systems with better insight into how the system is intended to operate under normal conditions. These information exchanges drive the design decisions, implementation tradeoffs, and operational needs of the system.

In addition to this information exchange, Defendable Architectures exhibit certain common characteristics. With these traits, systems are explicitly designed for Intelligence Driven Defense practices, allowing administrators and defenders to better perform active defense of the systems. These characteristics are summarized as visibility, manageability, and survivability.

Visibility is a critical enabler for Intelligence Driven Defense analysis. It allows operators and defenders to see the activity on the network, in the operating system, and at the application layer. It also provides a historical record of activity to investigate based on future intelligence.

Manageability ensures that a system can be sustained over time. Sustaining a system's security posture includes administrative activities such as system patching and configuration. It also entails being able to update the security controls of the system or its environment based on new threat intelligence. The speed and precision with which updates can be deployed greatly affects how well it can be defended.

Survivability allows the system to provide its intended services during attack, compromise, and recovery. Whereas classical security principles focus on hardening a system against initial attack, Defendable Architectures also address the system's ability to withstand lateral movement and support recovery from attack in an assured manner.

4 Creating Defendable Systems

Security is an inherent characteristic of a system, not a feature added afterward. Likewise, a system based on Defendable Architectures incorporates mission needs and threat intelligence, and exhibits the characteristics of visibility, manageability, and survivability as intrinsic parts of the system. The following sections describe how to design, build, run, and defend systems based on Defendable Architectures.

4.1 Designing Defendable Systems

It is the designer's responsibility to build Defendable Architecture characteristics into a system's design. While conventional approaches to security design focus on selecting controls, designers must begin with an understanding of the assets within the system, the threats against them, and the impact of those threats to the organization's mission and business needs. The primary security goal of a system is to protect its assets; therefore threats against those assets are the primary consideration of the design. Unlike vulnerabilities and design flaws, system threats represent a relatively stable set of security objectives for the system. As stated above, as long as the asset exists, so do the threats. This provides a basis for evaluation and selection of controls to incorporate into the system's design.

Following a structured threat analysis methodology helps designers identify the potential threats against the system and its assets. It also provides a means to assess which controls and design alternatives will effectively mitigate the identified threats and attack vectors. For example, encryption of data at rest can mitigate unauthorized disclosure, but the specific implementation determines which types of theft it will be effective against. It this example, disk-level encryption only mitigates physical theft, whereas application-level encryption mitigates system-level attacks. Each option impacts system performance and maintenance needs in different ways. These represent tradeoffs that the designer makes in mitigating threats and managing impacts other system requirements.

Even a mildly complex system has a number of assets, and therefore many potential threats. Designers utilize their understanding of the organization's mission and business needs to help assess and triage the system threats. Organizations with a mature threat intelligence practice are also able to incorporate this information into the system threat analysis process. The IDDIL/ATC methodology [6], for example, provides steps that naturally incorporate mission needs and threat intelligence. Figure 4 depicts these inputs into the system threat analysis process.

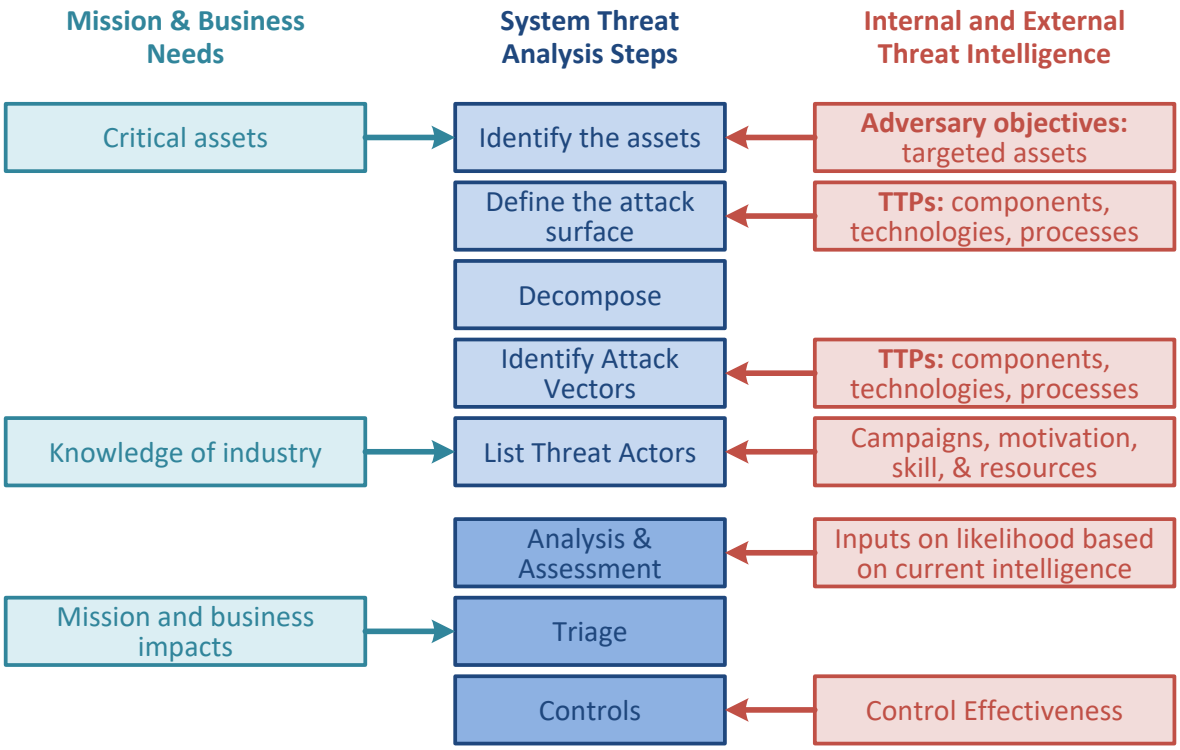

Figure 4: Incorporating mission needs and threat intelligence into system threat analysis

Using this combined analysis, the designer is able to make informed decisions about design tradeoffs and selection of effective security controls. Designers must carefully evaluate the extent to which they build controls to counter each threat. For example, as depicted in Figure 5 some threats against the system must be mitigated even in the absence of existing threat intelligence, particularly where the impact to the system, mission, or organization is severe.

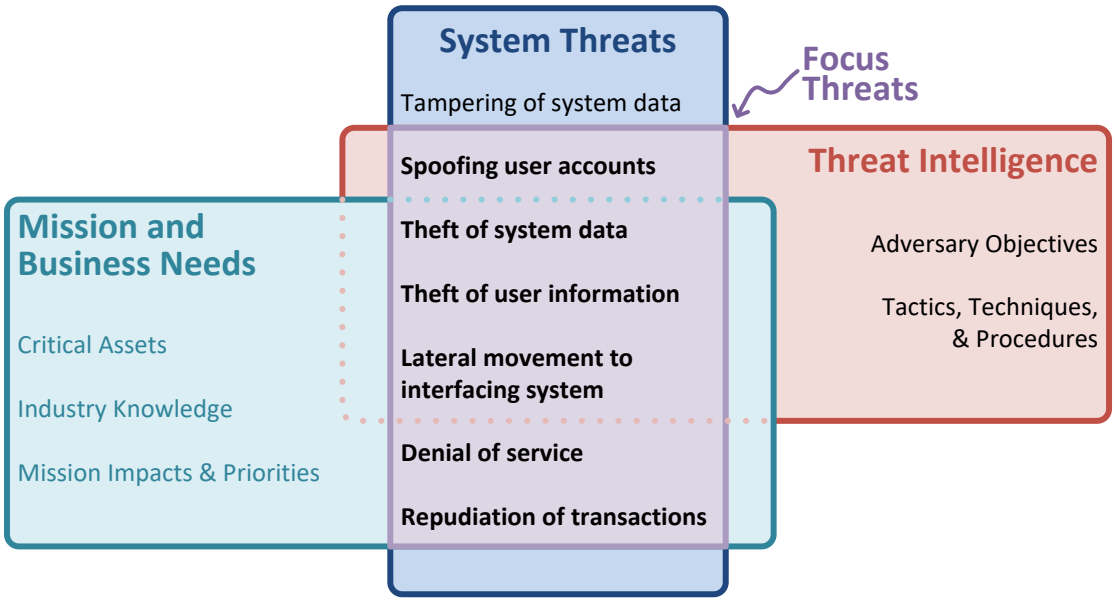

Figure 5: Using system threats, mission needs, and threat intelligence

When selecting controls, designers address both individual system threats and the need to actively defend the system as a whole. A catalog of the organization's security controls that spans the set of functions described in Figure 6 ensure that designers are aware of the security infrastructure and services available to their systems. By defining controls based on the *functions* they deliver, this catalog provides designers with clear guidance on what each control provides to the system, rather than what it prevents. Designers primarily leverage controls in the "Protect" functional group as mitigations to system threats, though other functions can be applied as compensating controls to address gaps in Protect functions.

However, solely mitigating individual threats against the system's assets does not yield a defendable system. As outlined in section 3, designing for Intelligence Driven Defense practices requires *visibility* into system activity and behavior, *manageability* for timely deployment of system updates and rules based on new intelligence, and *survivability* to continue to provide services during attack and recovery. Therefore, designers apply controls from the breadth of functional groups to build the key Defendable Architecture characteristics into the system. The subsections below describe how to incorporate each of these characteristics into a system's design.

Control Function	**Description**	**Visibility**	**Manageability**	**Survivability**
Inventory	Discover, track, and report on IT assets throughout their lifecycle	✓	✓	
Collect	Capture, organize, and retain system activity	✓		
Detect	Identify and alert on system activity			✓
Protect	Prevent attack and unauthorized system behavior			✓
Manage	Deploy system configuration, updates, and rules based on threat intelligence		✓	
Respond	Incident handling, system analysis, and recovery		✓	✓

Figure 6: Top level control functions from the Functional Controls Hierarchy [6]

An organization can deliver its controls at different tiers, as depicted in Figure 7. Security infrastructure consists of the functions built into the system's environment. Examples include full packet capture and network intrusion detection. Security integration services are those that a system can use, but must integrate with. Examples include log collection and authentication services. Reused security controls are deployed into the system from managed libraries, such as operating system images and cryptography libraries. Finally, application-specific security controls are those that an application must implement itself, such as input validation, memory checks, and application event logs. Designers use infrastructure and services where possible, allowing the system's developers and administrators to focus on core functionality of the system. It also leads to more consistent implementation of the controls and improves overall defense by providing visibility at the common infrastructure layer.

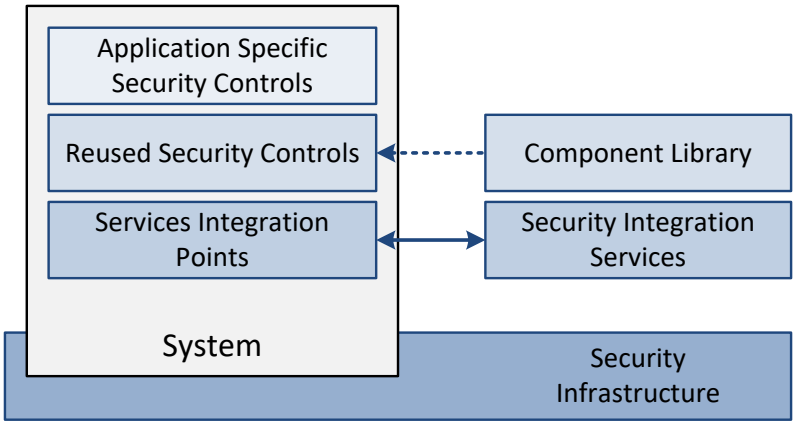

Figure 7: Tiers of available security controls

4.1.1 Designing for Visibility

The goal of designing for visibility is to enable defenders to comprehensively monitor current and historical activity within the system. Visibility controls also allow administrators and defenders to recreate the sequence of events over time and across components within the system. This requires visibility controls to be placed at critical vantage points within the system. It also means that the contents of the recorded events must contain enough information to correlate log entries over time and across components. Figure 8 provides a sample of questions and activities that a designer considers when determining visibility requirements.

Network	What talks to what, when?
	Show me everything that went over the network.
	What if the communications are encrypted?
	Tell me if known malicious traffic goes over the network.
Operating Systems	What's happening on the server / workstation?
	Tell me when something known malicious happens.
	What's installed and patched on each system?
	I need a forensic image of a given system.
Platform & Application	What is happening on the web server, DBMS, application, etc.?
	What is going into and out of the system?
	What did a user try to do in the system?
	What network and OS events correspond to this user action?
Identity & Access Management	What accounts where created, when, and by whom?
	Who authenticated, when, where, and how?
	What authorized actions did the user perform?
	What unauthorized actions did the user attempt?

Figure 8: Sample of visibility objectives at different system layers

The system threat analysis used for overall system architecture is also used in determining where to locate visibility controls in the system. Trust boundaries in a threat model represent important defensive lines to implement in the system. Monitoring activity at trust boundaries and system assets provides administrators with valuable diagnostic information, and gives defenders essential visibility into adversary activity. To the greatest extent possible, designers leverage the visibility infrastructure built into the system's environment and integration services available to the system.

Given both the real-time and historical usage of visibility, it is important to separate the concepts of event collection, detection, storage, and analysis. The processes for collecting network data and system logs can subsequently feed correlation and detection mechanisms, along with processing, storage, and analysis, as

depicted in Figure 9. Designing for visibility holistically across the enterprise ensures that important attack surfaces are monitored and that data collected from different sensors can be correlated and pivoted on. See Section 5 for more information about holistic visibility in the Defendable Enterprise.

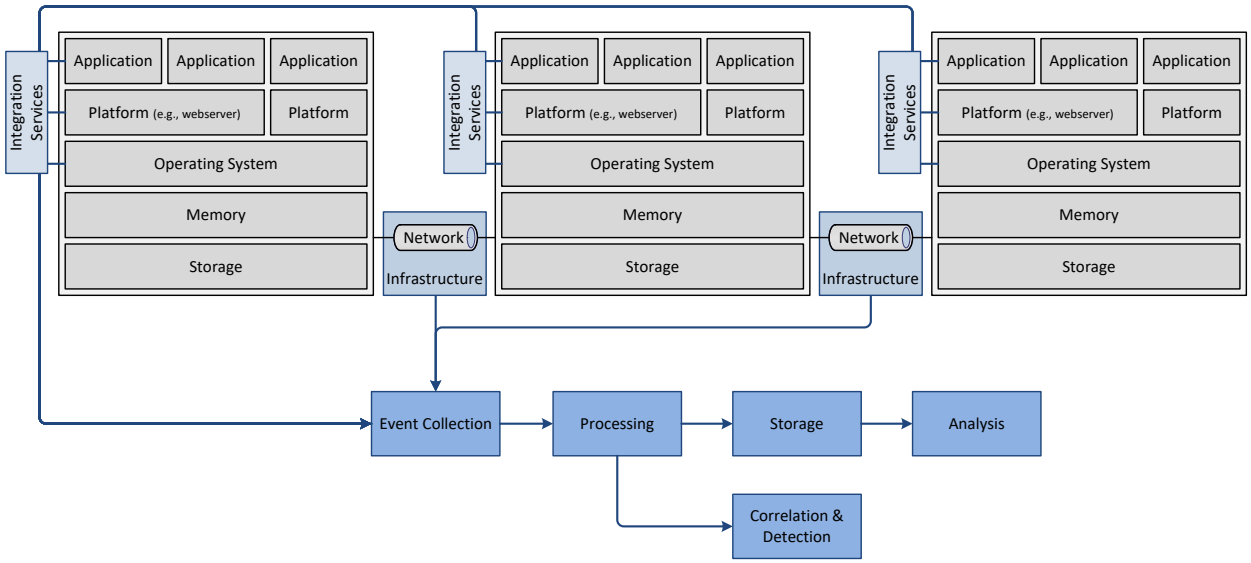

Figure 9: Visibility infrastructure

Collection and storage of visibility data puts demands on an organization's computing, network, and storage capacity. Designers, administrators, and defenders collaborate to prioritize visibility points, collection rates, and retention periods. Typically visibility requirements begin at the network layer, and then move up the stack to operating systems and applications. Each subsequent layer enriches the semantics of events being collected, but also adds volume and challenges in normalizing the data. Highly diverse visibility sources, such as application logs often do not lend themselves well to normalization. Often it is best to leave these logs in a relatively raw format for processing during analysis.

Visibility extends beyond monitoring active attacks. For example, knowing when and how an adversary is performing reconnaissance, and being able to correlate that activity to later stages of an attack, provides critical intelligence about how a particular adversary operates. As a designer, being aware of how the system can be abused in each stage of the attack and ensuring visibility into that activity are important parts of building Defendable Architectures.

4.1.2 Designing for Manageability

A manageable system is one that can be actively defended against current and emerging attacks, requiring a design that supports basic inventory, configuration, and vulnerability management as well as expedient updates to controls based on indicators, TTPs, and adversary objectives derived from threat intelligence.

Vulnerability management requires timely patching and configuration of all components of the system. While patching infrastructure can address much of this need, designers must recognize that some components are not readily supported by an enterprise patching system. These components typically require administrators to manually download and apply patches. By extension, merely patching software does not make it secure. Administrators must also set and maintain its configuration. As with patching, configuration management addresses much, but not all, of this problem set.

Therefore, designing for manageability also includes mechanisms for administrators to manually install, patch, and configure components. The Defendable Architecture includes controlled and monitored paths for administrators to access systems, download updates, and manage system configuration. The

characteristics of visibility and survivability are also applied to administration paths. Monitoring activity on administrator interfaces provides insight into what actions were taken when, by whom, and from where; essential for recreating the sequence of events from an incident. Survivability ensures that administrators will still be able to manage the system, even when it is under attack.

Paradoxically, many manageability functions can increase the system's attack surface by requiring additional external interfaces. The need and means of implementation of each for these functions must be considered in the context of the system and its threats. For instance, in most cases an automated interface to support patching is an appropriate implementation. However, in some scenarios, such as an air gapped system, reducing the patching frequency may in fact decrease overall system risk, assuming compensating controls are in place.

The second portion of designing for manageability is incorporating new threat intelligence into the system's security controls. Organizations that do this efficiently differentiate themselves in their cyber security defense from those based on classical security principles. Systems based on Defendable Architectures can achieve this portion of manageability chiefly though using common security infrastructure and services that apply to each layer in the system's architecture.

That means having controls at the network, operating system, platform, and application layers. This allows defenders to update controls at the appropriate level of the system's stack. For instance, intelligence about an adversary's infrastructure may yield indicators to block certain IP addresses, and be implemented in an IP firewall. Whereas intelligence about command-and-control protocols may yield rules based on HTTP headers and content, and be implemented in a web proxy. This ability to update controls at different layers in the system, and across multiple systems, allows defenders to implement precisely tuned rules, greatly improving the efficacy of deployed controls against current and emerging attacks. A cross-reference between an organization's controls catalog, such as the one outlined in [6], with the types of indicators that each control can effectively mitigate, provides important insight to both designers and defenders.

By aligning visibility vantage points with protection controls, defenders are able to test and tune rules based on historical network traffic and system logs. This allows defenders to confidently deploy new and updated rules with low risk of false positives or unintentional business impact. Furthermore, infrastructure based of Defendable Architectures can be adapted to changes in TTPs and even adversary objectives. Building protection on top of existing visibility collection capabilities provides a flexible architecture that streamlines adding new types of rules and blocking techniques based on an existing data collection pipeline, increasing defensive agility.

In this sense, Defendable Architectures are distinct from, though not mutually exclusive of, the classical approach of defense in depth. Traditionally, defense in depth promotes the idea that multiple controls with the similar purpose will offset each other's weaknesses, treating controls as a static cascade of filters in anticipation that at least one will have the correct configuration to block an attack. However, the heterogeneity imposed by defense in depth degrades a system's manageability by requiring administrators to be proficient in multiple tools and spend time maintaining them. Defendable Architectures focus creating systems that can implement actively tuned controls across the system's stack based on threat intelligence and system threat analysis.

4.1.3 Designing for Survivability

Survivability is the ability of the system to provide its intended service during attack, compromise, and recovery. The objectives of an adversary affect the types of survival mechanisms a system should have in place. Attackers whose objective is computer network exploitation (CNE) aim to exfiltrate information from the targeted organization. In this case, compromise of a system rarely causes the system to fail; part of the intruder's objective is to remain undetected, and many attackers go to great length to evade

detection. In the case of CNE, it is the discovery and recovery process – not initial compromise – that is most disruptive to the system. Organizations often induce failure once a successful attack is detected, by taking the system offline to perform analysis and recovery.

In contrast to CNE, attackers whose goal is computer network attack (CNA) are intent on disrupting or destroying portions of an organization's assets. In some cases, the system under attack is the target, such as in a denial of service (DOS) attack. However, often the CNA sequence begins with gaining undetected access to systems that provide subsequent access to the target through lateral movement. The systems on this attack vector are subject to the same survivability needs as those targeted for CNE.

In the case of both CNE and CNA, system designers must compare system threat analysis artifacts with available threat intelligence about the adversaries' objectives, as described in Section 4.1. This allows the designer to choose defensive measures matching needs of the system with the anticipated attack objectives and business criticality of the system. Changing adversary objectives may necessitate changes to the design of the system or supporting controls. For example, new threat intelligence indicating that a system may be targeted for data tampering would prioritize implementation of fine-grained auditing and integrity controls in the system. These needs can be anticipated during the design phase using threat analysis techniques. Even where the controls aren't built into the initial implementation of a system, they can be designed for future integration.

The most common response to incident recovery is to take the entire system offline, perform forensic analysis on the system to understand when, how, and which parts have been compromised, then restore the system configuration and data from backups or build procedures. While this design and operational approach is acceptable for many systems, a much more sophisticated approach is required for mission-critical systems.

A more advanced survivable design will allow the system to continue operating during analysis and recovery. The ability to remain in operation during recovery is dependent on an architecture that (1) provides thorough visibility and analysis into the current and historical state of the systems components, (2) allows components to be taken offline and put back online without impacting normal operation.

Visibility in this case is an extension of the characteristic described above. In order to support discovery and analysis of attacker activity, visibility must cover the depth of the system stack, and breadth across the components in the application. It must also span the time domain, from on-demand forensic capture to historical activity in the system. These will include network traffic capture, logging changes to file systems, operating systems, application configuration, and application activity. Finally, it requires the ability to perform full forensic capture of system state, including memory and network traffic. These are necessary for intrusion analysts to assess current state of the system and correlate with indicators and campaign intelligence from other sources. Based on this analysis, it is then possible to determine which components have been compromised and for how long. An assured recovery can then be performed based on system backups, build procedures, and test plans.

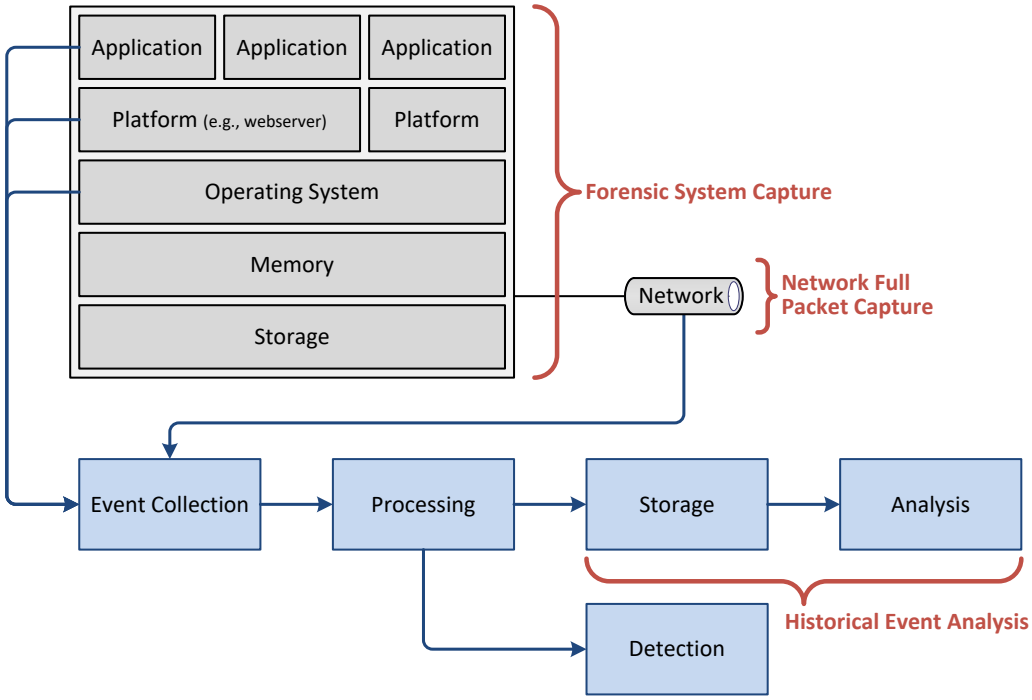

Figure 10: Visibility for intrusion analysis and recovery

The second requirement of advanced survivable designs – allowing components to be taken offline and put back online – necessitates designing for component failure. Several design patterns exist for supporting component failure, and many modern system architectures are designed to gracefully withstand the failure of individual components. This allows systems to survive loss of availability against parts of the system, whether accidental (e.g., system crash) or intentional (e.g., denial of service attack). Chaos Monkey [7] is an excellent example of explicitly designing and building for component failure. By intentionally designing components for – and being continuously subjecting to – failure, an overall system can be highly resilient to availability faults. Designers evaluate the system for single points of failure or compromise to understand the aggregate survivability of the system as a whole.

Designing for component failure only addresses loss of availability; it does not adequately address compromise of a system's components due to attack. In this case, designers also consider the trustworthiness of a component. In both CNE and CNA, individual components may continue to run, but may not be trusted to perform their function correctly. The system is designed to allow administrators to selectively change components' trusted status in the system. For some designs, this can be accomplished using infrastructure controls such as disabling service accounts or network isolation. Other systems require application-level controls to achieve the necessary degree of granularity.

4.1.3.1 Lateral movement

Many attackers use an initially compromised system as a foothold to move laterally within and between systems. This allows an attacker to both broaden his span of access within an enterprise and also to choose weaker targets for initial entry.

In addition to being able to gracefully recover from intrusion, a survivable system is designed to contain the initial compromise. The implication of lateral movement is that designers further scrutinize trust boundaries within the system. Very few interfaces between components should be considered fully trusted. A clear understanding of what interfacing components are *capable* of doing to each other –

instead of merely what they are *intended* to do – is necessary in order to properly mitigate lateral movement.

When evaluating lateral movement, designers consider components that do not normally interface with each other, but are also not *prevented* from interfacing. This is a common problem with neighboring systems in a hosting environment. Even if they are not designed to interface, implicit trust of adjacent systems can allow an attacker to establish direct connections, or via common infrastructure such as authentication or backup services. Threat intelligence can provide insight into commonly used lateral movement TTPs, which help designers prioritize compensating controls.

Resistance to lateral movement begins with traditional system hardening and least privilege configuration to limit the attack surface between systems. However, Defendable Architectures also ensure visibility into lateral movement through logging and network monitoring.

Design decisions on system segmentation can drastically affect survivability. The foundational design principles of coupling and cohesion have a strong influence over a system's resistance to lateral movement. Tightly coupled components tend to share a high degree of trust, which is easier to exploit for lateral movement. Consider the design alternatives, depicted in Figure 11, for running two software services. A simplified set of design options balance sharing accounts and servers. In configuration A, the services are tightly coupled and share trust via the operating system and service account. The service itself is the only defensive boundary against an attacker who has gained control over the other service. At the other extreme, configuration D uses separate accounts and separate servers, forcing the attacker to compromise another account and another server.

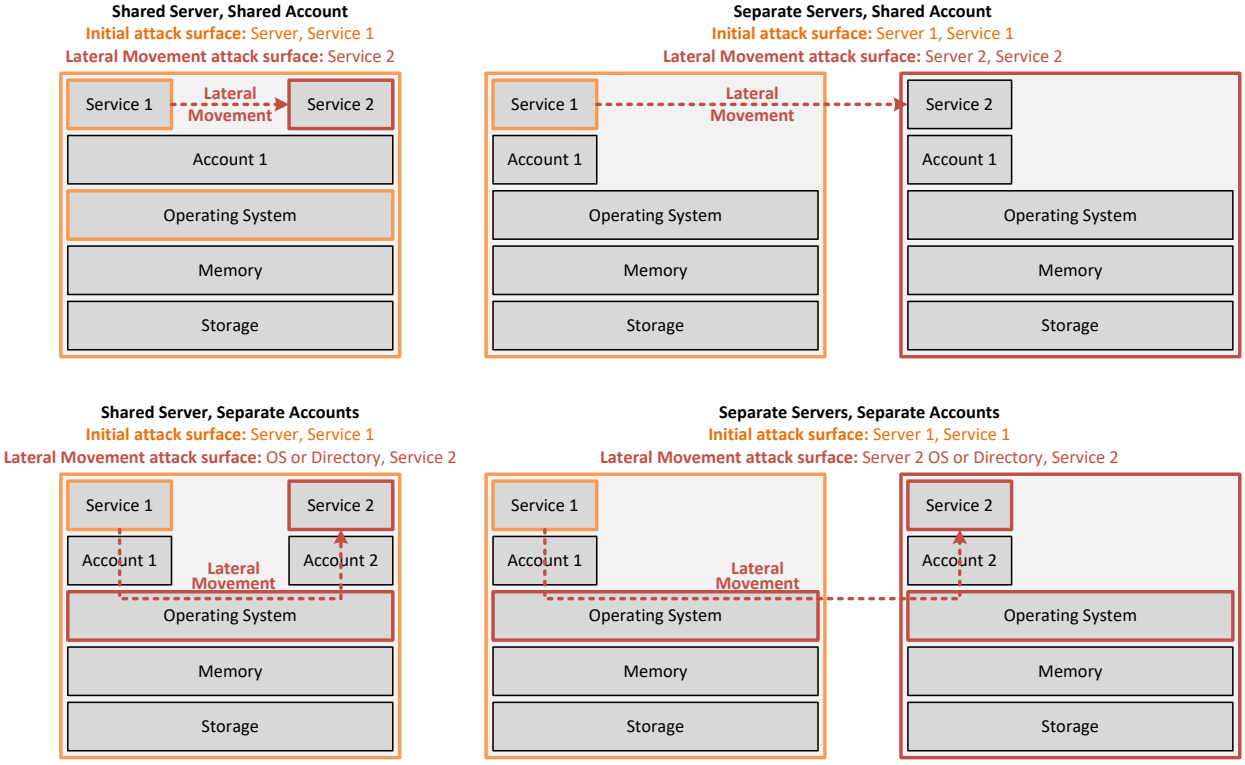

Figure 11: Comparison of apparent attack surfaces in different component segmentation options

Analyzing how, and to what degree, an attack surface can be defended helps designers make informed decisions about system configuration. By considering how to provide visibility and survivability on attack surfaces, designers can determine the appropriate forms of segmentation within the system and the degree

to which individual components can be trusted. The designer weighs these options against the threats against the system, threat intelligence about lateral movement, and the manageability constraints on the system. As with most design decisions, these tradeoffs must be taken holistically to consider impacts to the system functionality, security, and operational costs.

4.2 Building Defendable Systems

Developers, engineers, and testers are responsible for implementing and integrating defensive measures in systems. Leveraging system threat analysis from designers and threat intelligence from defenders, developers select the correct control implementations, and testers prioritize verification methods based on real-world attacks.

As with designers, the system's developers and testers must be aware of available security infrastructure and integration services. Reusing these existing implementations reduces the work required to implement the system, allowing the build team to focus on core system functionality. Reuse also provides greater consistency for administrators and defenders in interacting with systems across the enterprise.

4.2.1 Building for Visibility

The most effective action that developers and engineers take in improving the visibility of their system is implementing robust application logging. Application logs record both events relevant to the mission of the application and to the application's security. Rich security logs provide insight beyond traditional allow/deny and success/failure events, to show how the application reacted to various inputs and conditions.

For example, user inputs that are parameters for an SQL statement should be logged and the resulting application action recorded. Using this, a defender has insight into which SQL injection attempts the application successfully blocked and which ones it failed to block. Furthermore, the contents of event logs allow system activity to be reconstructed across the software stack and over time. This means including the attributes such as event IDs, user IDs, and timestamps necessary to correlate events. Defenders provide guidance to developers on the log contents necessary to efficiently perform this reconstruction.

4.2.2 Building for Manageability

To implement the manageability features in a Defendable Architecture, developers clearly define and segregate all administration functions. Differentiating administration traffic from standard end user traffic allows defenders to more readily identify attacks and elevation of privilege attempts. It also enables stronger control over administrative interfaces, such as two-factor authentication and limiting access to trusted network paths.

Defending a system requires more than just protecting the production servers. Source code, compiled executables, configuration files, and operating system images are all parts of the system as a whole. An adversary who is able to embed malicious software in a system's source code is able to operate with a degree of impunity that most system designers don't anticipate.

Therefore, defendable architecture principles are applied to source code control components, system build scripts, and development and test environments. Close attention is also paid to the procedures for migrating code, configuration, and data into the production environment. This form of manageability must account for the steps required to deliver and validate all components deployed into a production environment.

4.2.3 Building for Survivability

Building survivability into a system begins with identifying its essential data, functions, and services, typically identified as part of system threat analysis. Based on this, a development team applies the appropriate implementation patterns and principles such as redundancy, encapsulation, and decoupling to

ensure that a system can operate when individual components fail. Developing a system that provides graceful degradation of services when key components are unavailable allows operators to continue to use the system, even during intrusion analysis and recovery.

Developers also scrutinize internal trust boundaries for lateral movement opportunities. Few interfaces require the participating components to fully trust one another. Instead, developers define the *purpose* of the trust, and constrain the interface to that purpose. For example, instead of a database server broadly trusting a web server, the DBMS trusts the web server's service account to query specific tables in a given database instance.

Testing contributes enormously to a system's survivability; perhaps more than any of the other characteristics of Defendable Architectures. As with most systems' availability requirements, testers verify the system's ability to failover or provide acceptable functionality in the cases of component failure. In addition, testers also verify that defenders can perform the necessary intrusion analysis while the system is online. Examples include collecting forensic system images, retrieving files and backups, and recovering detailed system logs. Testing also confirms the ability to put components back online without disruption.

Testing is essential in understanding how a system performs under attack. As with Defendable Architectures as a whole, the test design must also leverage available threat intelligence. Test plans that simulate the types of attacks the system will be subject to are built using a combination of system threat analysis and threat intelligence about TTPs and adversary objectives. These test scenarios address both initial attack attempts and lateral movement within the system. The purpose is to test the system's ability to be actively *defended*. This requires testing not just how well-hardened a system is; but also how well it provides visibility, survivability, and manageability. As stated in [1], testing can't prove that a system is secure. But it can verify that the controls support actively defending it.

4.3 Running Defendable Systems

Operators and administrators are responsible for maintaining the integrity of the system while in operations. They also have the most first-hand experience with the system, and are therefore in the best position to assist defenders with investigations and analysis. Initial detection alerts typically require further analysis before a compromise can be confirmed. Administrators' knowledge about a system help defenders better understand the configuration and "normal" activity within the system, in order to determine which components may be compromised. In this sense, the inventory of systems, their design, and configuration is a form of visibility, essential in quickly analyzing a potentially compromised system.

Administrators are also charged with vulnerability management within the system. Based on the design and implementation, some of these activities, such as patching, may be automated. However, in most systems, a subset of the software will need to be manually managed. This requires administrators to perform software updates and patches in a timely basis. It also requires them to have processes to verify the authenticity of all executables brought into the system. Code signing is a common technique; however other more manual processes may be necessary where delivered code is not signed. The requirement to authenticate executables applies equally to commercial, open source, and custom software. All represent potential attack vectors against the system, though mitigations may vary between them.

Finally, administrators of systems based on Defendable Architectures must adhere to the defined administration paths designed into the system. Deviation from these network paths, software interfaces, or authentication mechanisms, will likely trigger false positive detections by defenders.

4.4 Defending Defendable Systems

This paper does not seek to explain how to conduct Intelligence Driven Defense analysis, as several texts and entire curricula already describe approaches to defending systems. Instead, this section describes how

defenders can support the design, development, and operations of Defendable Architectures. It also describes the ways that these systems are more effectively defended.

Defendable systems feature pragmatic designs driven by a working knowledge of adversaries and the need to actively defend systems while in operations. Defenders' most important contribution to Defendable Architecture is threat intelligence. Sharing their knowledge gained from cyber intelligence analysis, defenders help organizations make pragmatic choices in selecting and implementing security controls. These choices allow teams to focus their efforts on implementing the most effective controls well. Strong intelligence management processes allow defenders to aggregate information about TTPs and adversary objectives to share with designers and developers. More importantly, organizations with advanced intelligence analysis capabilities are able to recognize trends and changes in TTPs, adversary objectives, and existence of new adversaries. Significant changes in these trigger changes in infrastructure and design decisions of Defendable Architectures.

Defenders communicate their visibility needs in both content and vantage points. By specifying the basic contents of application logs, for example, defenders can readily pivot across historical logs to analyze intrusion attempts. This also supports correlating activity across the system stack, including network, OS, platform, and application.

Using this greater visibility, defenders are able to use indicators to test and deploy precisely tuned rules to the security controls supporting a system. Historical data allows defenders to find relationships between new attacks and past activity, revealing new intelligence about changes in adversaries and campaigns. Defenders also test new detection and blocking rules against this historical data, lending confidence to the accuracy of the rules.

The manageability of Defendable Architectures allows defenders to quickly and accurately deploy these new rules to the appropriate controls. Most systems and enterprises have a broad set of security controls with overlapping capabilities. Defendable Architectures help provide clarity into the most effective place to deploy new rules. These decisions are informed by the alignment of controls with visibility points, and defenders' insight into which visibility sources most closely align with the required indicators.

Of course, defenders' most identifiable role is intrusion detection and response. Defendable Architectures improve the analysis and response to an intrusion by giving defenders deep forensic visibility into the system. This enables online analysis of systems and shortens the timeline for offline forensic analysis. For example, being able to immediately capture and transfer a forensic image over the network, instead of installing agent software and shipping physical media, takes hours or days out of the process. Additionally, systems designed for survivability are able to continue providing their services through analysis and recovery while containing the intrusion, thus minimizing business impact and disruption to end user tasks.

5 The Defendable Enterprise

Individual systems cannot be secured on their own. Intelligence Driven Defense solutions are most effective when performed across an entire organization, and enriched with intelligence sharing within an industry. Likewise, Defendable Architectures exist in the context of their environment and enterprise at large. Many of the visibility, manageability, and survivability controls that differentiate Defendable Architectures from traditional information security approaches can be applied to a Defendable Enterprise. A Defendable Enterprise's infrastructure simultaneously increases the span of visibility that defenders have at their disposal, and reduces the cost and complexity of systems fielded in the environment.

As with building individual systems, managing a Defendable Enterprise begins with the intersection of threat analysis and threat intelligence. Threat modeling and analysis can be adapted to enterprises in a

similar manner as it can be applied to individual systems. The following table shows how to apply the IDDIL/ATC methodology from [6] to enterprise analysis.

Threat analysis step	Enterprise analysis step
Identify assets	Identify major enterprise assets (e.g., ERP) and classes of assets (e.g., web servers)
Define attack surface	Define enterprise boundaries (e.g., internet gateways, mail gateways, offsite backups, off-network workstations)
Decompose	Decompose aspects of the enterprise based on risk and complexity
Identify attack vectors	Utilize threat intelligence to articulate current and emerging attack vectors. Brainstorm unexploited or undiscovered attack vectors.
List threat actors	Utilize threat intelligence to articulate known classes of threat actors and their objectives. Brainstorm undiscovered or future threat actors.
Analysis & Assessment	Determine potential business and technical impacts based on identified threats
Triage	Prioritize threats and mitigations based on impacts, threat intelligence, and available mitigations
Controls	Deploy new and update existing control infrastructure

Figure 12: Threat analysis at the enterprise level

This analysis allows security infrastructure investments to be prioritized based on risks in an objective fashion. New trends in attack types can yield the need for new types of infrastructure services. It also allows enterprises to deemphasize or retire existing security services based on deprecated value. Some security functionality that was highly effective (and expensive) at the time of initial deployment recedes to commodity status. Services that provide commodity-level security controls should be afforded a commodity-level portion of the organization's budget. Threat intelligence also informs the timing of infrastructure deployment. For example, the cyclic nature of some adversaries' activity can be used to prioritize the timing of infrastructure implementation.

Just as the overall architecture of a system must account for the system's context, a system's security controls must also account for the defensive infrastructure of the enterprise. Using available infrastructure ensures consistent implementation of controls across systems, and provides defenders with comprehensive visibility and manageability across the enterprise. Significantly, common infrastructure allows a large number of systems to immediately benefit from new indicators and resulting rules as they are deployed to the infrastructure controls. The most effective Defendable Enterprises can translate newly revealed indicators into updated controls in the fewest steps. This is particularly important for tactical mitigations, such as in the case of disclosure of widespread vulnerabilities, such as Heartbleed and Shellshock in 2014.

The value of Defendable Enterprise infrastructure is driven in large part by the rich forms of visibility provided by enterprise-wide systems leveraging the infrastructure. A Defendable Enterprise ensures that designers, developers, and administrators understand the available infrastructure and services, their functionality, interfaces, and how to position systems to best take advantage of them. For instance, a system benefits from a network intrusion detection system if the NIDS is able to inspect all traffic in a decrypted form. Several design alternatives can achieve this goal; designers and developers must understand the choices and tradeoffs for their system. A catalog of security infrastructure and services

helps promote their uniform understanding and use across systems in the enterprise. The functional control hierarchy in [6] outlines one such catalog.

The same catalog of security services used by designers, developers, and administrators is also used by defenders to assess which types of controls are best suited for which indicators. This consistency between design, build, run, and defend phases has the additional benefit of being able to track the efficacy of security controls against adversary actions. Defenders analyze which controls detected or blocked intrusion attempts for each adversary campaign, providing insight into adversary activity, effectiveness of controls against them, and missing controls, allowing enterprises to assess the value of their security controls and plan future enhancements. As a result, manageability at the enterprise level is measured in both the speed and precision for deployment of new rules and in defender's ability to assess the efficacy of those rules.

Breadth of visibility is the foundation of the Defendable Enterprise. Collection and retention of network and system activity provides the insight necessary to prioritize detection controls and build the business cases to transition from detection to blocking. This transition is enabled through confidence the fidelity of detection and blocking rules based on practical experience with live data and assessment against historical data. A flexible architecture that builds active blocking controls on top of collection and detection streams, as depicted in Figure 13, lets the Defendable Enterprise make incremental investments in its infrastructure. Detection rules can be deployed inline or centrally, depending on the rule complexity and the capability of distributed sensors. Aligning blocking and detection rules into the same pipeline also enables the enterprise to readily adapt to changes in adversary TTPs through deployment of new detection and blocking modules on existing pipelines.

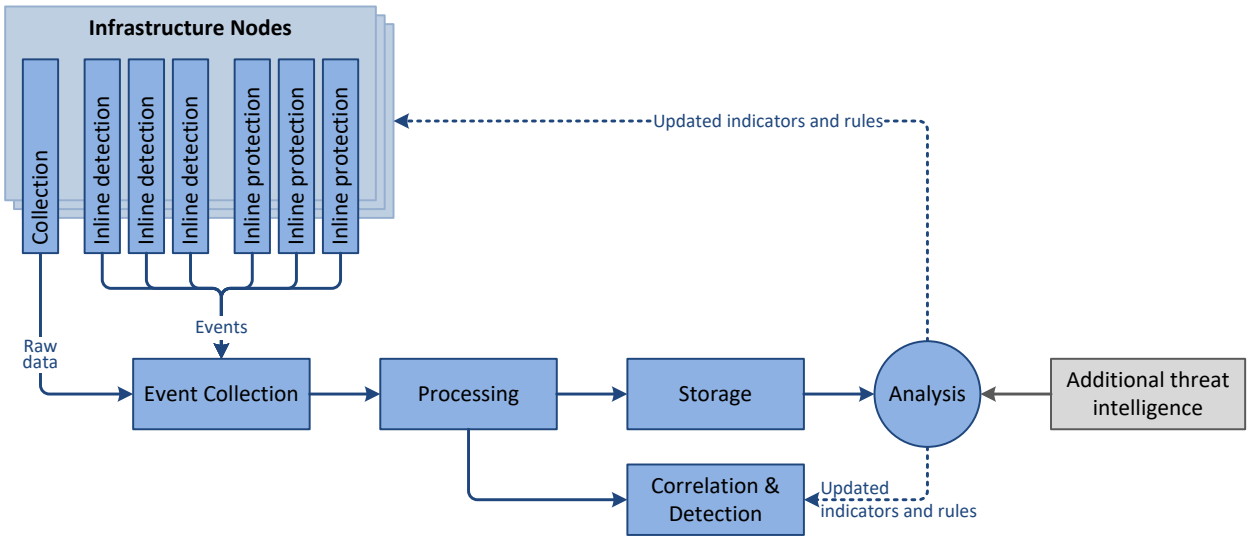

Figure 13: Combined collection, detection, and blocking infrastructure for enterprise visibility and manageability

Enterprise survivability applies many of the same system-level principles to the enterprise as a whole. Cohesion, coupling, and segmentation are essential design considerations, as lateral movement introduces a degree of shared risk that individual system designers may not readily account for. Segmentation across the enterprise is aligned with the most stable aspects of the business structure, to mitigate impacts during organization changes, mergers, acquisitions, and divestitures. Segmentation also leverages technology across the stack, including network, virtualization, operating system, identity and access management, and application functionality.

The overall security of an enterprise depends on its resilience to attacks, and also its resilience to changes in attackers. Both of these have the potential to overwhelm an organization, whether responding to individual attacks, or investing in infrastructure to address changes in adversaries and their objectives. The Defendable Enterprise leverages its threat intelligence and enterprise threat analysis to build flexible security infrastructure and a survivable IT infrastructure. This enables organizations to make effective decisions in the design, development, operations, and defense of their systems and assets.

6 Summary

Classical security engineering and architecture has been trying to solve the wrong problem. It is not sufficient to try to build *hardened* systems; instead we must build systems that are *defendable*. A system's requirements, design, or test results can't be declared as "secure." Rather, it is a combination of how the system is designed, built, operated, and defended that ultimately protects the system and its assets over time. Because adversaries adapt their own techniques based on changing objectives and opportunities, systems and enterprises must be actively defended.

Defendable Architectures represent systems that are designed for Intelligence Driven Defense practices. They leverage the knowledge of designers, developers, administrators, and defenders throughout the system's lifecycle. In this way, organizations make informed and pragmatic decisions about the security controls implemented in their systems. To effectively support their active defense, systems build on Defendable Architectures express the characteristics of visibility, manageability, and survivability.

By extension, these concepts are applied at a larger scale to build a Defendable Enterprise. These organizations are able to use their threat analysis and threat intelligence to design, deploy, and operate security infrastructure that addresses their defensive needs. A Defendable Enterprise is able to efficiently translate new threat intelligence into updated signatures, infrastructure changes, and system design patterns, enabling them to effectively respond to their adversaries. Such organizations are also able to make informed decisions about the deployment of new security infrastructure based on enterprise-level threat intelligence and threat analysis.

Organizations seeking to achieve cyber resilience must consider both the need to survive attacks and to have architectures that are resilient to changes in attacker's techniques and objectives. By leveraging threat intelligence and designing for visibility, manageability, and survivability, organizations and the systems they implement are positioned be actively defended and to adapt to new types of attacks.

References

[1] Department of Defense, Defense Science Board, "Resilient Military Systems and the Advanced Cyber Threat," Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, Washington, DC, 2013.

[2] World Economic Forum; McKinsey & Company, "Risk and Responsibility in a Hyperconnected World," World Economic Forum, Geneva, 2014.

[3] R. A. Caralli, J. H. Allen, P. D. Curtis, D. W. White and L. R. Young, "CERT Resilence Management Model, Version 1.0," Software Engineering Institute, 2010.

[4] "Presidential Policy Directive / PPD-21 -- Critical Infrastructure Security and Resilience".

[5] E. Hutchins, M. Cloppert and R. Amin, "Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains," in *Proceedings of the 6th International Conference on Information Warfare and Security*, 2011.

[6] M. Muckin and S. C. Fitch, "A Threat-Driven Approach to Cyber Security," 2014.

[7] C. Bennett and A. Tseitlin, "Chaos Monkey Released Into The Wild," 30 July 2012. [Online]. Available: http://techblog.netflix.com/2012/07/chaos-monkey-released-into-wild.html.

