splunk>

WHITE PAPER

SPLUNK FOR THE MSSP
TECHNICAL ARCHITECTURE

How to Deploy Splunk as a Managed Security Services Provider

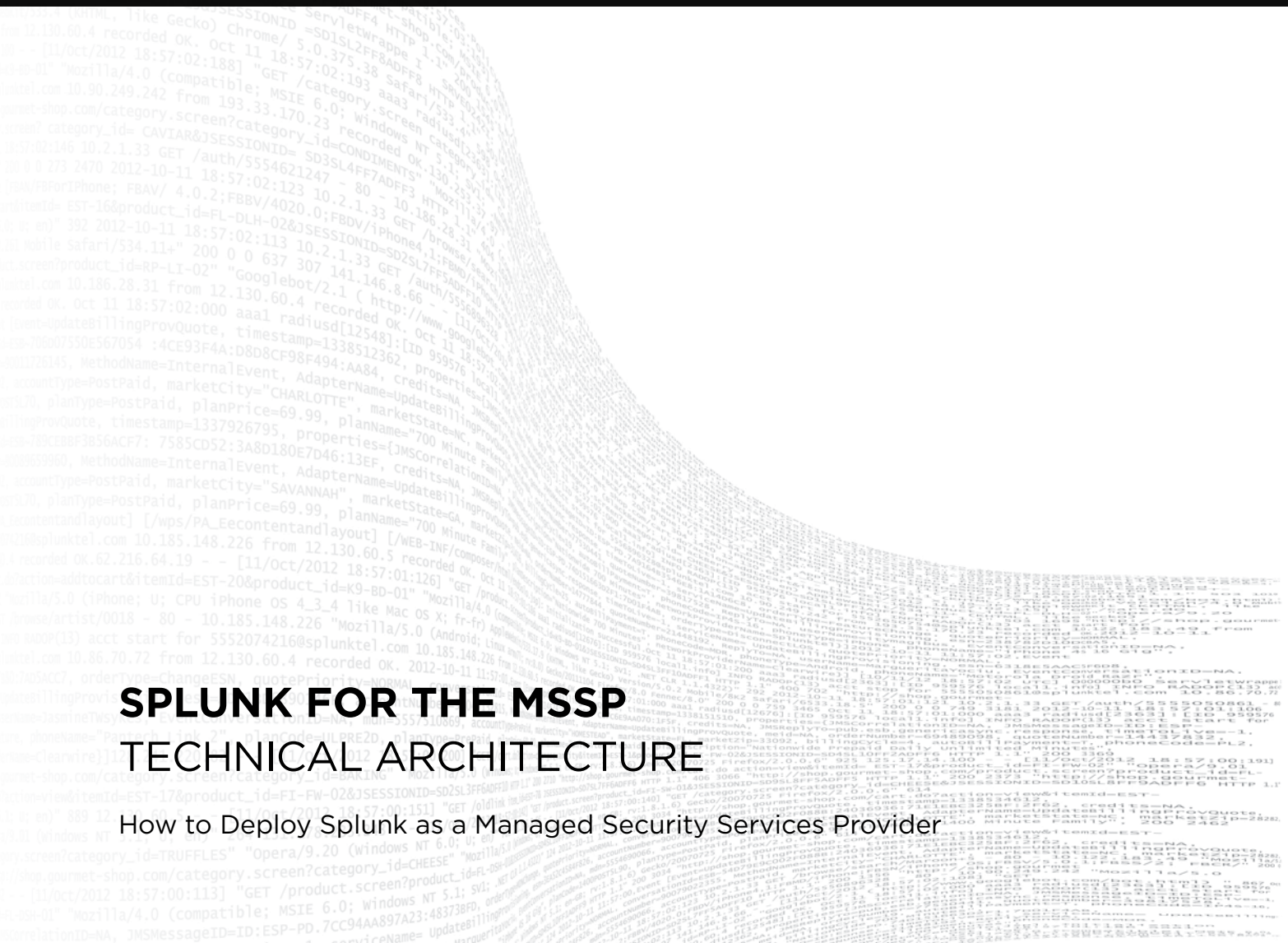

TABLE OF CONTENTS

Who should read this?

This document is intended for managed security service providers (MSSPs) or managed SIEM providers interested in deploying Splunk as the monitoring, SIEM and security analytics platform at the heart of their managed security service. It describes how to architect a Splunk deployment to service multiple customers at a time.

The audience is assumed to have a basic technical understanding of Splunk components, such as forwarders, indexers and search heads. If you are not familiar with those yet, the free Splunk Infrastructure Overview course from Splunk Education is an excellent introduction.

We also assume that the MSSP's intended use cases cover full SIEM scenarios and more, so they are interested in both ***Splunk Enterprise*** and ***Splunk Enterprise Security*** (Splunk ES).

Aside from MSSPs, this document is also relevant to anyone interested in multi-tenant-like Splunk Enterprise and Splunk Enterprise Security environments. Multiple Splunk environments, aimed at different internal customers. Large corporations, governmental entities and universities often fit this bill, since they sometimes run Splunk "as a service" internally, serving multiple clients with varying needs. The architecture for these scenarios may look a little different than the MSSP's, as architecture is always dependent on the requirements, but this paper should still be useful.

What are some of an MSSP's requirements?

Here are a few things that many MSSPs will care about. This is not an exhaustive list, naturally, but a good place to start. These requirements guide the architecture that follows. Splunk is highly flexible, so the basic design here can be tweaked to accommodate other requirements.

Lower the cost of delivery per customer

When managing large numbers of customers (hundreds in some cases), the cost of delivering their service is important to an MSSP. Costs can be of various types:

Customer onboarding costs

Avoiding manual, custom, non-repeatable work is key to eliminating unnecessary costs, so MSSPs find it useful to be able to spin up a new customer in an easy, automated fashion. This could include any setup and teardown done for testing purposes before the customer is officially onboarded and under management. The work here includes technical work like hardware deployment, OS configuration, software configuration, testing and selecting initial use cases. It also includes non-technical, transactional costs associated with interacting with the vendor and customer.

Ongoing system administration costs

Lowering the cost of managing the SIEM solution on an ongoing basis is also important. The more economically the operations can be performed and scaled, the better the MSSP's margins are. The system administration being referred to here refers not only to tuning the SIEM configuration, but to the surrounding infrastructure needed for it to work. Hosts, networks, storage systems and more. Configuration, monitoring uptime, upgrades, backups and patching all take cycles. These costs remain in some fashion as long as the customer is under management, all the way until their contract ends and they are decommissioned.

Ongoing security operations costs

This refers to the day to the day work of the SOC, and all the work that entails. With millions of events needing to be triaged every day, SOC analysts spend a great deal of time wading through noise, figuring out what to focus on, investigating what seems more suspicious than normal, communicating with other tiers and the client, and working through a feedback loop to make their work easier in the future. To ensure that the customer perceives them as a useful security partner, not just a source of un-actionable alerts, competitive MSSPs will invest a significant amount of time in tuning detections, figuring out easy or low-risk remedial actions to automate, fine-tuning the collaboration with the customer's existing SOC (if any), showcasing the value they provide, and analyzing their internal operations constantly to be able to run them better.

Sharing resources

This is an important concern that overlaps with the concerns mentioned before but deserves to be called out on its own. Leveraging investments by sharing them among multiple customers is attractive. Whether it is hardware, software, workflows, best practices or human expertise, MSSPs are interested in extracting the most out of every resource they have. Virtualized or cloud-native deployments start to look appealing. So does knowledge sharing and collaboration across tiers so that work is not wasted or lost. Reusing things across similar clients is also appealing. Anything that requires significant human effort or technology investments to create or use- detection rules, investigative patterns post detection, remedial measures, client recommendation mechanisms, escalation and communication best practices - is a candidate for reuse.

Staff retention

MSSPs are not immune to the staffing challenges that in-house SOCs face. Staffing a large-scale SOC like an MSSP's is hard, and attrition affects the business. Freeing analysts up from mundane tasks is important not just for operational efficiency, but to keep good employees by increasing their job satisfaction and offering space for learning and working on higher-value tasks. Automating common workflows and responses to things they see every day is a key part of this.

Data segregation and leakage

MSSPs are required to keep each customer's data separated. This is an important concern, and usually top of mind. The requirement comes under various names - data hygiene, data segregation, data separation or data leakage. The appropriate level of separation at the technical level depends on the end customer and the MSSP's own legal and security requirements. Some customers demand entirely separate physical hardware and isolated networks, others are satisfied with VM-level segregation, some are open to separation via security groups (in AWS parlance) in the cloud, others are happy with application-level separation and tagging of their data.

Performance and configuration impacts on other customers

Similar to the data segregation problem, the workload of one customer should not impact the performance of another, nor should configuration changes to one customer upset another. Each customer has an individual agreement with the MSSP and, regardless of what their Service Level Agreement (SLA) with the MSSP entitles them to, generally does not expect to be inconvenienced or have their service quality lowered on account of other customers. Understanding the impact of things that are "shared" among different customers, such as virtualized infrastructure, is important here.

Ways to manage multiple customer profiles or types

Grouping customers into a limited number of types or buckets helps the MSSP manage fewer variants or "snowflakes". This lowers costs, increases reliability and offers greater agility. Using templates for each of these buckets to quickly configure a customer is a plus. The templates here include setting up the OS/VM environment in addition to the configuration of the SIEM platform itself, including queries, reports, and integrations with other pieces of the security and operational stack. Different customers have different needs, governmental customers often have unique concerns, and the MSSP business landscape changes, so many different templates could be used in practice. Flexibility of the SIEM is an important asset.

High-Availability (HA) and Disaster Recovery (DR)

High-Availability and Disaster Recovery are often required to keep the managed service running. The requirements here vary. Generally, they cover the SIEM platform's availability, the ability to cope with spikes in load, and backing up the data. These are tied to the level of service being offered to the customer and include concerns such as of lookback for incidents or access to the data after they leave the managed service. The complete HA or DR solution for the MSSP will extend to the other systems beyond the SIEM/analytics platform itself.

Latency Requirements

This is a more "detailed" requirement that occasionally comes up. Latency is obviously important for DR as far as the time taken to roll over to the backup site. Another aspect to it is how quickly an MSSP is able to spot a threat. To gauge the MSSP's SLA compliance, customers may be interested in seeing how long it took the MSSP to triage or escalate an incident, based on the initial event's timestamp – each customer may be synchronized to a different Stratum-1 time source than the MSSP. For an MSSP, dealing with hundreds of customer time scales is not a practical option. Because of this, they may settle on a single way to interpret times for all customers for SLA purposes. This is easiest by considering relative time. The MSSP's time would be considered canonical, and the clock starts when the data makes its way into the MSSP's environment, as shown at the MSSP itself. This can be further defined as the time at which the MSSP can reasonably make use of the data, which in Splunk terms, for example, would be the time at which the data is indexed. This is not necessarily a business concern for all MSSPs, so is usually not a focus. Many MSSP offerings are focused on individual deployments at individual customers (not the MSSP's premises), so there is only one time scale to manage at a time.

Alert Triaging and Escalation

MSSP analysts need to triage alerts efficiently, investigate the extent of the possible incident quickly and escalate as appropriate. They are usually governed by SLAs that guarantee a certain response time at different service levels. SLAs are important because they included in the customer contract, and SLA breaches can be grounds for penalties or service termination.

In general, higher quality, relevant, contextual escalations take more work. This does not have to mean more *time*, but often does when tools are limited, or processes are complicated and/or manual. This results in a natural tradeoff. If the service provider feels obligated to escalate something they are not quite sure about because of a looming SLA, they are not really offloading work from their customer, and the customer gets less value out of the service than they hoped for.

Transparency and visibility – "What are you doing for me?"

At higher service levels, customers are keen to understand the value that their MSSP provides. Metrics involved may include things like threats blocked, actors involved, business services protected, efficiencies provided, and responsiveness metrics like time to triage, investigate and escalate.

The level of reporting varies with service levels and customer maturity. Customers of lower maturity who intentionally offload all security work to the MSSP may not care as much about the mechanics of the MSSP's work. With their smaller budgets, thye are always under pressure to justify their investments in security, however, so they still care about the business value the MSSP provides.

Custom vs scaled-out offerings also matters. For standardized offerings with many customers, the reporting will feature standard content and distribution intervals. It may also be simpler, depending on the maturity and desire of the customers. Reporting affects margins, as does everything in a scaled-out offering. In general, an MSSP's business model hinges on repeatably, reliably, and quickly deploying and running services for many customers.

Architeture Overview

If you are not quite an MSSP

Reminder again that the architecture below is tailored to the general needs of an MSSP, balancing concerns that are seen most often. Every business is unique, so it is possible that something else would work better. It is always a good idea to work closely with Splunk to build out something that addresses particular needs.

MSPs and MSSPs are one piece of the "Splunk as a service" space, and they focus on serving *external* customers. MSSP-like scenarios also arise when in large organizations that offer Splunk "as a service" *internally*. For example:

- Large corporations with multiple business units or geographical sites, each with an individual SOC, and perhaps a higher level SOC that has central visibility of some sort.
- Universities with multiple departments.
- Governmental agencies with sub-agencies focused on particular missions.

All of these might need to provide Splunk as a shared service to others. The requirements will differ. Different use cases (even beyond security), legal or policy-driven constraints, segregation requirements, size and scale of internal customers, top-level visibility needs, management processes, analyst workflows, etc. The correct architecture will always be subject to these. The below may still be useful, but keep in mind that it may be balancing different requirements.

Where can Splunk be deployed?

Splunk is software-only and widely deployed on every type of platform – on-premise, private cloud and public cloud. As with all customers, MSSPs have complete flexibility in where to deploy it. The location tends to be driven by the service offering (at-customer, at-MSSP or hybrid), and then internal decisions to flesh that out – what at-MSSP actually means in terms datacenter location and private/public cloud approaches). The architecture below does not require any particular deployment platform or location. As a working model, it assumes that Splunk is deployed in the MSSP's datacenter, and receives data shipped across the WAN from the customer's on-premise IT or OT infrastructure. This does not in any way rule out cloud variants of this – either cloud-hosted customer systems, or cloud-based deployments of Splunk. MSSPs have deployed Splunk in all of these variants, including all major public cloud vendors (AWS, Azure, etc.). The unique ability of Splunk to offer hybrid search across on-prem and cloud-hosted Splunk deployments also makes it an attractive option for hybrid service offerings involving a partial Splunk deployment at the customer premises, communicating with one in the cloud or at Splunk Cloud (the SaaS option for Splunk).

The classic MSSP model involves taking the customer's data out of its original environment and into the MSSP's for long-term storage and ongoing security monitoring. This allows the MSSP to centralize operations within one or more SOCs, and to deploy as little as possible at the customer's premises to ease remote management, monitoring and reduce truck rolls. This model is the basis for the working model here. MSSPs who prefer a cloud environment can take advantage of very detailed guidance from Splunk on selecting instance types, storage, etc. For AWS-specific recommendations, **go here**. This document should still be used as the overall blueprint for how to deploy Splunk in the cloud or within private data centers. The cloud-specific references offer additional details on the specifics on those platforms.

An alternative location for the MSSP's Splunk environment is Splunk Cloud. Many MSSPs take advantage of the robust, SOC 2-type 2 certified SaaS variant of Splunk as a way to speed up deployments, free themselves from ongoing management of the infrastructure and software of the Splunk systems and focus resources on delivering additional security expertise to their customer. With many large customers already running in Splunk Cloud, it is a proven way to use Splunk at scale. Splunk Cloud is a key option for fully managed SIEM scenarios where the MSSP operates it entirely on behalf of the customer. It is also common in co-managed SIEM scenarios, where customers and the MSSP both use the Splunk Cloud system for different business purposes – the customer might focus on

IT Operations Analytics or Business Analytics use cases, while the MSSP takes on most of the security responsibilities. When Splunk Cloud is the chosen platform, not all of the below will apply.

The major reason is that the architecting will be left to Splunk Cloud, not the MSSP, as is normal for a SaaS offering. Another is that Splunk Cloud environments will be single-customer ones (managed or co-managed SIEM, not "shared" multi-customer MSSP), so most guidelines below for setting up a multi-customer environment are not needed.

Which products do I need?

Splunk Enterprise will always be required, since it is the foundational machine data analytics platform providing all data ingestion, indexing, searching, reporting, and alerting capabilities. When full analytical SIEM functionality is desired, *Splunk Enterprise Security* will usually be deployed on top of Splunk Enterprise. Splunk Enterprise Security offers a great deal of out of the box security reporting and analytics, investigation workflow support, threat intelligence handling, and risk-oriented event analysis capabilities that are useful to a high-performing SOC. A full discussion of the scope of Splunk Enterprise Security's capabilities is beyond the scope of this paper. Note that Splunk Enterprise Security is not required to use Splunk as an analytical SIEM within the MSSP. Depending on the situation, Splunk Enterprise alone can fit the bill. Typically, advanced users will build a lot of the workflow they need using the facilities offered by Splunk Enterprise. This is an area worth discussing with Splunk.

Overall Pattern – With Splunk Enterprise Security (most common)

It is recommended that each customer have their own deployment. Each deployment includes all the software and hardware needed to run Splunk Enterprise, Splunk Enterprise Security and any other apps needed in addition to Splunk Enterprise Security. Logically, this looks like this:

Figure 1 - Splunk Reference Architecture with Splunk ES – Default Recommendation

Each vertical stack above represents a complete physical deployment - forwarders, indexers, search heads, and anything else that would exist in any single distributed Splunk implementation. Each stack may build in capabilities for HA and DR using Search Head Clustering and/or Indexer Clustering, as required. Individual stacks do not have to mean different or manually configured stacks, however. It is very common to apply common policies and configurations to each stack using templates, as will be discussed a little later.

From the MSSP's perspective, this is a multi-tenant *service offering* that hosts multiple customers (tenants). Individual stacks can be as similar or dissimilar as needed. Architecting the service as separate stacks, however, mitigates many concerns with software that is notionally multi-tenant and separates or tags customers logically, but shares resources under the hood. It maximizes the possibility of achieving **all** of the following goals:

1. **Ensuring adequate data separation between customers** – This is key. It is possible to achieve in a shared environment but is more difficult when customers demand more than the data being tagged and identified uniquely.

2 **Performance guarantees** – Customers, of course, do not want to be impacted by performance problems in another's environment. Slowdowns can impact their perception of the MSSP's value and affect the MSSP's ability to impact their security posture.

3. **Specific customer requirements and lifecycles** – Customers may have unique business requirements that need to be taken into account, requiring unique configuration. They also follow their own upgrade lifecycles, so their environments may not match each other's. And finally, they often need to be offered unique value – advanced or vertical-specific use cases, for example.

4. **Protection during maintenance** - Safely maintaining each customer's environment through configuration changes, patches, software upgrades etc. without fear of impacting another.

If a **central view** across the different environments is needed, this can be done in a variety of ways, using either Splunk itself or external systems. These approaches are described further below.

Central configuration of the different environments will be common when the number of deployments increases. This has frequently been done by large Splunk customers using common IT automation tools like Ansible, Chef and Puppet. Splunk is designed to be highly configurable, and to allow that configuration in many ways besides the UI – on the command line, via text-based configuration files, or by using the extensive REST APIs. All of these lend themselves to configuration at scale via automation, which brings all the advantages of reliability, predictability, speed and lower costs to the MSSP. This is described further below as well.

Alternative architectures – Not Recommended

These are mentioned for completeness since they are sometimes proposed by longtime Splunk users.

Splunk does not recommend using the approach below, which uses separate ES search heads per customer, but shares the indexer tier, separating data into indexes. This can be attempted by parties with extremely strong Splunk skills, but is generally deemed an unsuitable approach for most.

Figure 2 - Not recommended – separate Splunk ES search heads, shared indexer tier

The approach below shares a single Splunk deployment for all customers, with a single ES search head for all customers and a shared indexing tier that separates data into indexes. It is similar to a normal Splunk deployment, but when this is an ES search head, it is not recommended.

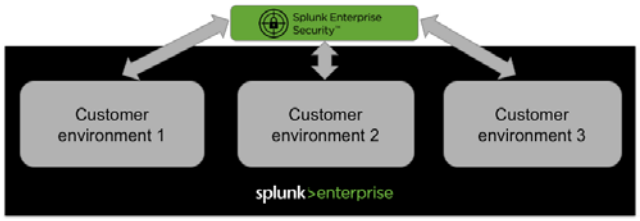

Figure 3 – Not recommended – single Splunk ES search head, shared indexer tier

Overall Pattern – With Splunk Enterprise alone (no Splunk Enterprise Security)

Default Recommendation

The basic pattern of separate stacks is also the most usual recommendation for Splunk Enterprise-only deployments, as it preserves the 4 fundamental advantages mentioned above. This looks like below:

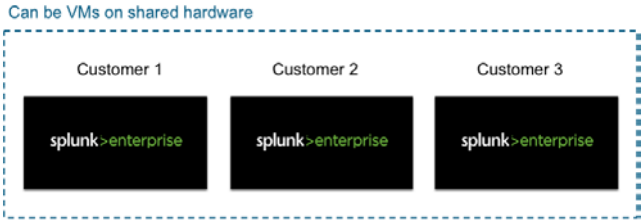

Figure 4 - Default architecture recommendation for Splunk Enterprise-only deployments

Splunk Enterprise by itself is a good solution for MSSPs whose customers have relatively simpler needs, say in the SMB arena. It provides all the core analytical capabilities inherited by Splunk Enterprise Security, has extensive, easy to use dashboarding and reporting capabilities, and provides the complete stack that covers data ingestion, log retention and management, ad hoc querying, machine learning to aid in detection, very customizable alerting, and the entire ecosystem of Splunk apps for quicker time to value. Where the advanced features of Splunk Enterprise Security are not needed yet – say the Risk, Threat Intel, Investigation data and workflow management, or Adaptive Response frameworks - Splunk Enterprise is an excellent option. A discussion of the requirements with Splunk would be welcomed so we can advise on one path vs the other.

Alternative Architecture – suitable in some circumstances

Assuming that only Splunk Enterprise is being deployed (i.e. without Splunk Enterprise Security), an alternative architecture is possible in some circumstances. This trades off some of the 4 advantages above to gain the management advantage of a single distributed Splunk deployment that is multi-tenant. This looks like below:

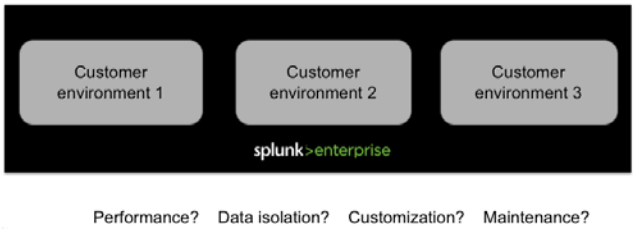

Figure 5 – Alternative architecture for Splunk Enterprise-only deployments – suitable on occasion

This is a single Splunk deployment. It shares resources at the intermediate forwarder, indexer and search head tiers, with the forwarding tier responsible for gathering data from individual customer premises (or clouds) being customer-specific.

That covered the overall patterns. Next, we will take a look at the details of the implementation at each layer of the Splunk deployment, going in order of data flow:

1. Forwarding Tier – Collecting data from the end customer's environment
2. Indexing Tier – Storing the customer's data and making it searchable
3. Searching, Reporting and Analytics Tier

Forwarding Tier – Collecting data from the customer

Collection mechanisms

Splunk has a vast number of ways to collect data from conceivably any kind of system including network devices, detailed endpoint data, custom applications, cloud data sources, or message buses like Kafka. It is also very flexible in how it lets you deploy them. This ability to ingest data is one of the platform's unique strengths, and describing it fully is beyond the scope of this paper. We will cover a few major categories of data ingestion here for example purposes. Collecting additional data and fine-tuning it to reduce noise for the analysts is very possible in Splunk.

The most common way to collect data in Splunk is via the Splunk forwarder. For those platforms supported by the *Splunk Universal Forwarder (UF),* such as Linux and Windows servers. workstations, and domain controllers, simply deploy the UF to those systems. Splunk offers a component called the Deployment Server to make mass configuration of forwarders possible. In some situations, *a Heavy Forwarder* may be called for, such as when pulling data from a database, but the Universal Forwarder is usually the best option, and should be the first thing to consider.

UFs are small, high-performance pieces of software that sit on a system and pull information from there. Where a UF may not be installed directly on the source of the data, such as with network security appliances that offer only the ability to send logs out via syslog, Splunk best practice is to aggregate all syslog data with a syslog server like `syslog-ng` or `rsyslog`, have it write each data source out to a separate file, and then use a UF to pick up these files and send them on to the Splunk indexers. A syslog server provides resilience and a large on-disk buffer in case the network link from the customer's premises to the MSSP goes down. While the UF can receive network data and send it on natively, the syslog server and file approach is a more robust solution to unreliable connectivity.

In some cases, high-value security data may reside in systms incapable of sending data out over syslog or incapable of allowing a Splunk forwarder to be installed, such as mainframes or ICS devices. Mechanisms for these exist as well. When more sophisticated pulling mechanisms are required, such as when a custom python-based Splunk modular input is needed to poll a data source, a Splunk Heavy Forwarder is needed instead of a UF. Heavy Forwarders provide additional capabilities, such as a natively running Python environment. Note that a Heavy Forwarder is not a "more powerful" forwarder, capable of higher throughput. This is a common misconception. It is so-called because it offers more *functionality* than a UF.

It is also now common for useful data to be emitted in HTTP format, not over syslog. This is increasingly so with custom applications that are useful sources of authentication data and user activity, and which log natively in JSON to an HTTP port. It is also common with IoT devices, which have little processing power to run anything like a forwarder and make the assumption that they will send their logs outward to a sink over HTTP. Collecting this kind of data in Splunk is easy via the HTTP Event Collector. This is a feature that can be enabled either on a Heavy Forwarder (should a centralized collection point be desired), or directly on the Splunk indexers. Generally, MSSPs will have a central collector to aggregate all data at the customer's premises, so it will be more common to enable the HEC there.

Splunk supports natively collecting Netflow (and cousins like IPFIX), Layer 7 metadata, and targeted full packet captures as well, which are quite useful for forensic investigations. This is done via the free Splunk Stream. Splunk Stream can be deployed standalone in many places in a network, or enabled on a forwarder, which means it can easily collect data from the Cloud. When enabled on a forwarder, it becomes possible to collect endpoint data, custom applications logs, syslog data, and now wire data all from the same place.

Splunk also has extensive, native support for gathering security or operational data from Cloud environments. Office 365, AWS service logs, Azure logs and more can be easily gathered. These are becoming increasingly important as infrastructure migrates to the cloud. The AWS Add-On can be used to gather data natively from many AWS services, for example. Event sources can also push data to Splunk via Lambda functions or Kinesis firehose to the HEC. This whitepaper is a useful resource on gathering AWS data for security purposes **here**. Cloud data sources are frequently pulled in when customer infrastructure and MSSP Splunk instances are both in the Cloud.

Getting data from the customer to the MSSP

Regardless of the collection method, one key requirement is that the data be sent to the MSSP via secure transport. Although there are many ways to implement this, there are three methods discussed here:

1. A *customer-premise* system containing a Splunk UF and syslog server, sending data to the MSSP.
2. An *MSSP-premise* Splunk UF using a VPN to pull data from the customer site.
3. A *combination*: MSSP-premise machine running a Splunk UF and syslog server that receives data from a customer-premise syslog server.

The implementation chosen, being one of the above, a combination, or something else entirely, matters only in how well it integrates into the MSSP and customer operational models. This is important to understand. As long as all relevant and needed data is sent to the MSSP securely and reliably, and the received data is distributed evenly across the indexing tier, the specific choice does not matter.

Technique 1: Customer-premise aggregator running Splunk UF and syslog server

In many cases, the preferred approach is to place an MSSP-controlled system at the edge of the customer site to serve as an *aggregator*. This aggregates all data from within the customer site, adds any metadata or enrichment the MSSP needs (note that Splunk does *not* require data enrichment or normalization at ingest time due to its unique schema-on-read architecture), then sends it to the MSSP SOC. A Splunk UF installed on a commodity server can serve as this central aggregator. This is a fairly standard configuration for Splunk, also seen in Splunk Cloud deployments.

Many common data sources for MSSPs send their logs out over syslog. For these cases, Splunk best practice is to use a separate syslog server installed on the same machine as the UF to serve as the aggregator for all of these. This may be syslog-ng, rsyslog, or something similar. The syslog server should write each source to a separate file. The UF on the same machine can then pick those files up and transmit them to the Splunk indexers at the MSSP's premises. The same UF can also receive data from other UFs installed further within the customer's endpoints (servers or workstations).

The advantage of encrypting the data at this central aggregation point is that it allows data to be collected from the internal systems in its original unencrypted format, minimizing complexity, then encrypting it all at once as it goes out across the WAN to the

MSSP. It also allows granular control over specifying sources, designating sourcetypes, and other setting other metadata on the data using syslog filters and Splunk forwarder configurations. Care should be taken to transmit the raw data to the indexers as far as possible, as that provides maximum fidelity for security investigations, and allows the security analysts to manipulate that data and extract relevant fields at any time using Splunk's schema-at-read functionality, which is extremely useful. Sending raw data also enables technical add-ons installed at the indexer or search head level to receive the data they expect, then extract fields that they are configured to extract. Splunk will accept any text-based data, so field extraction at ingest time is rarely necessary, contrary to what older data systems require, and which most MSSPs will be used to. Extracting fields later (at search time) unlocks a great deal of flexibility unexpected questions in the heat of a serious investigation.

In some cases, there may be *multiple layers of syslog* coming from largely isolated devices and networks within the customer site, requiring many local syslog servers. These servers would all send their data to the MSSP-controlled customer-edge system. This approach takes advantage of existing collection points within the end customer site and adds a new one at the end, rather than implementing an entirely new way to get data to the MSSP.

Aside from forwarders and syslog servers, the collection infrastructure within the customer premises may also require heavy forwarders, such as when pulling data from a database using Splunk **DB Connect**. All of this data will be forwarded to the aggregator before going outward to the MSSP.

The UF in the aggregator will then forward the data to another forwarder (discussed **below**) in the MSSP's DMZ. This is referred to as an intermediate forwarder. It is no different from any other forwarder; "intermediate" just refers to where it sits, between the original forwarder and the indexers. This forwarder-to-forwarder communication happens over TLS, thus meeting encryption in motion requirements.

This customer-premise aggregator approach has a notable advantage in the case of WAN outages between customer and MSSP sites. It provides a large local buffer that can collect data during the outage, which can be sent on once the outage is resolved. During customer onboarding and before the service rollout, it is highly recommended that latency, outage scenarios and corresponding buffering needs be tested. For many implementations, the default queue-size-related settings (see outputs.conf) should suffice. These are configurable. Further guarantees against loss of in-flight data can be provided by using the Indexer Acknowledgement capability, which makes the forwarder wait for an acknowledgement from the indexer that it has received data, and will cause it to resend it if no ack is received.

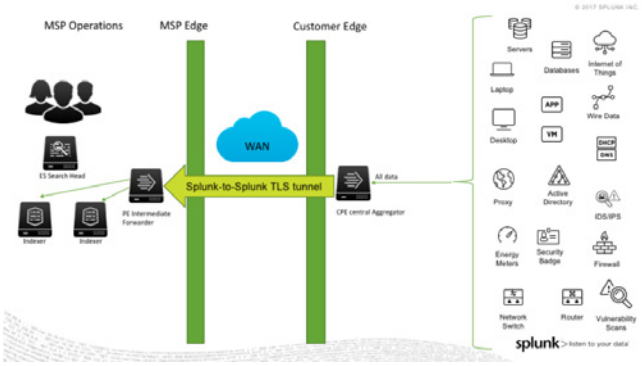

Figure 6 – Data collection technique 1: Customer-premise aggregator running Splunk UF and syslog server

Technique 2: No customer-premise aggregator; send directly over VPN to MSSP

An alternative to an MSSP controlled aggregator at the customer edge is to eliminate that system and establish a VPN between the MSSP SOC and the customer across which all data is sent over syslog. The difference from the previous architecture is that all data from the customer site routes *directly via the VPN* to a system hosted by the MSSP, rather than routing via an aggregator. This adds the complexity of maintaining a site-to-site VPN, but removes the need for an MSSP to manage the aggregator. This approach eliminates those in favor of taking on VPN maintenance instead. It is also an option when customers already own Splunk in some capacity, so supporting this new destination - by adding a new destination to the existing forwarders or telling the existing indexers to forward out a copy of the data they have indexed - is within their own Splunk administrative capabilities.

This has to be balanced against the higher potential for data loss (if the forwarders are adding a destination). Should the VPN or network layer become unavailable, thus stopping all traffic from the customer site to the MSSP, the data intended for the MSSP would have to be buffered at the customer edge until connectivity returns. With the removal of the syslog server + forwarder system at the edge, this buffer no longer exists. Where the data is forwarded out of existing indexers, the original still remains in place within them, so another attempt can be made. Where the original data is coming from endpoints or servers with universal/heavy forwarders installed, those forwarders can do some buffering at that level. Where the original data is sent from the original network sources over syslog, on the other hand (when there is no UF installed), there is a problem; syslog over UDP (the norm) is generally is not designed as anything other than fire and forget. The buffers for sending syslog via TCP are minimal at best. The result is that some data loss will likely occur if the WAN link fails for more than a few seconds. The VPN connections themselves may prove less stable than the underlying WAN transport, so care must be taken for this particular approach.

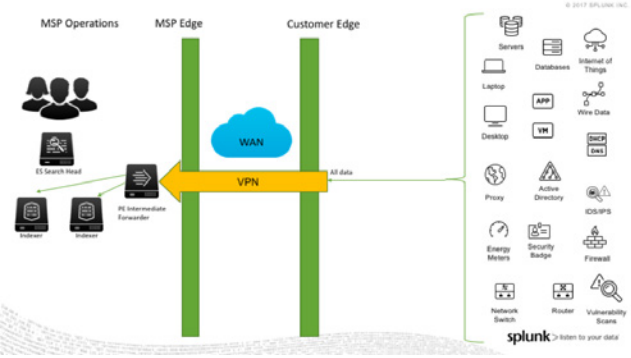

Figure 7 - Technique 3: Combination of MSSP-premise Splunk HF + syslog server AND a customer-premise syslog server

Technique 3: Customer premise syslog aggregator AND MSSP-premise UF + syslog server

The third option is a hybrid design:

1. A customer-premise syslog server AND
2. An MSSP-premise Splunk UF + syslog server

Under this scheme, different types of data are sent to the MSSP in different ways:

- All Splunk UF/HF-supporting devices send to the MSSP-premise UF directly across the WAN, using encrypted Splunk-to-Splunk communication.
- All syslog-supporting devices send first to an on-premise syslog server at the customer site, which serves as the central aggregator for all syslog data. This forwards on to another syslog server hosted at the MSSP. The primary advantage to this arrangement is that the traffic between the two syslog servers can use TLS for secure transmission of data, even when the bulk of the original syslog sources are only capable of sending syslog in clear-text via UDP.

This reduces some level of management overhead and complexity for the MSSP by freeing them from having to manage a customer-premise Splunk UF, while gaining the encryption and buffering advantages of a customer-premise syslog server. Syslog servers are mature technology, so customers may be able to maintain this themselves if needed.

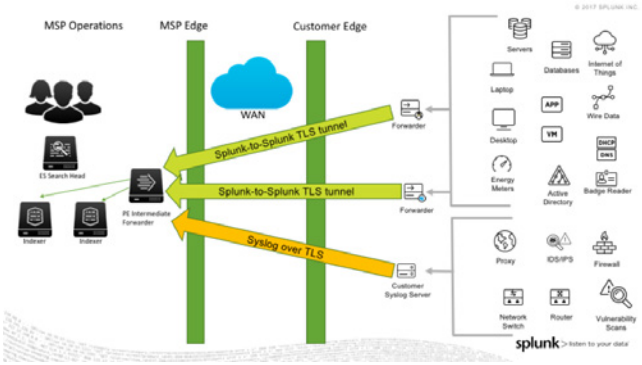

Figure 7 - Technique 3: Combination of MSSP-premise Splunk HF + syslog server AND a customer-premise syslog server

At the MSSP - Intermediate Forwarders

The first landing point for data received by the MSSP across the WAN is an Intermediate Forwarder hosted in the MSSP's DMZ. This can be a Universal or Heavy forwarder depending on routing requirements. The UF can do simpler routing, while the HF can do routing based on the *contents* of events. There are other considerations for UF vs HF decisions, some of which are described here.

These intermediate forwarders can be either dedicated to single customers or shared among multiple, depending on the indexer tier implementation. It is recommended that the architecture here match that of the indexing tier (described shortly).

Dedicated Splunk Stacks – with Splunk Enterprise Security or Splunk Enterprise
In an architecture with dedicated full stacks per customer, each customer should have at least two intermediate forwarders for high availability and redundancy, as well as to ensure even data distribution from the forwarders spraying data to them. (See below for more on event distribution). This reduces network- or system load-related bottlenecks.

Shared Indexer Tier – an option for Splunk Enterprise
When the indexer tier is shared, the MSSP can either dedicate per-customer intermediate forwarders or share intermediate forwarders across customers. The recommendation is to match intermediate forwarder architecture to the indexer tier architecture that it feeds, i.e. if the indexer tier is shared, then the intermediate forwarder tier ought to be shared as well to reduce complexity and maximize hardware utilization. This also provides for easier calculations for matching intermediate forwarders to indexers in the recommended ratios to ensure even data distribution.

How intermediate forwarders affect data distribution
When thinking about intermediate forwarders, it is important to ensure that they spray the data they receive evenly among the receiving indexers. Why? When searches are issued to the indexers, they return partial results based on the data they hold. When data is distributed evenly among the indexers, they can participate evenly in the search process, making maximum use of Splunk's MapReduce capabilities. Conversely, uneven data distribution means that certain indexers will participate disproportionately in the search process, which affects the speed with which results are retuned. Even data distribution also ensures that storage is utilized uniformly across indexers, making maximum use of the storage investment in each. The general rule of thumb for the forwarder to indexer ratio is to have twice as many forwarders as receiving indexers. This can be accomplished via extra forwarders but is more simply done by enabling additional forwarding pipelines on the forwarder, as many as the CPU can accommodate.

The forwarder's `outputs.conf` should be set up to spray data evenly across the receiving systems. This has the added benefit of reducing the impact of network connectivity or other related outages on one or more of the intermediate forwarders. To set up load-balanced forwarding to the indexers:

- Set `EVENT_BREAKER_ENABLE` to true
- Configure the `EVENT_BREAKER` to a regex that represents the event boundary.
- ensure `autoLBFrequency` is set to an acceptable value, such as the suggested 5 seconds. This minimizes the time the intermediate forwarder sends to any receiving indexers, thus maximizing the even spread of data.
- Set `maxKBps` to 0 (unlimited)
- Increase the number of pipelines as much as possible by configuring `parallelIngestionPipelines`, depending on the number of CPU cores available.

Indexing Tier

There are two ways to implement the indexing tier. As discussed in the **overall pattern section**, we recommend dedicated full stacks per customer as a starting point for all implementations, which includes the indexing tier. An alternative when Splunk Enterprise is deployed by itself is to share the indexing tier for all customers. This is an important decision to get right and depends heavily on the business needs of the MSSP. Each option is described below.

Option 1: Dedicated Full Stacks (Recommended for Splunk ES or Core-only deployments)

This is easy to understand – each customer gets their own set of *indexers*, as many as are needed to handle the ingestion volume and searching needs (since indexers do most of the work in searching). The indexers need not be physical machines – virtual machines are common and fully supported, as are cloud-deployed indexers. There are some considerations for deploying Splunk in virtual environments that are covered **here**, which are mainly around ensuring that resources for Splunk are reserved, and ensuring that shared storage has adequate performance. Storage performance is critical for indexers.

The separate indexers hold data for a particular customer. This ensures that the deployment contains only their data, satisfying the most stringent data segregation requirements. These indexers will only be used to provide managed services for this customer.

Recommendation: Even with dedicated full stacks, it is a good idea to follow the customer-specific naming conventions described in the Shared Indexer Tier. This makes it easy to identify and back up data at the filesystem layer if needed, and easier to migrate to the shared indexer tier in the case of a Splunk Enterprise-only deployment.

Option 2: Shared Indexer Tier (Only for Splunk Enterprise)

This option is only for Splunk Enterprise. Instead of separate indexers per customer, this design uses a common indexer cluster for all of them. The indexing tier is therefore shared. Within the cluster, each customer's data is separated into *indexes* instead of than completely separate indexERS.

An *index* is a logical collection of data that belongs together for one business reason or another. It can contain different types of data, so the question is about how this data will be used. Each customer should get their own set of indexes, as many as make sense. An index is physically a directory in Splunk. It is made up of *buckets* on disk, which are the primary unit of data storage in Splunk. Buckets contain raw data, from a particular time frame, plus additional metadata that Splunk creates to make this data searchable very fast. The index is also the level at which RBAC is applied in Splunk, which makes it a good way to segment each customer's data. Splunk has sophisticated RBAC capabilities that enable you to specify exactly who can do what with data in any given index.

IndexERS contain many indexes. They are a physical system running Splunk, serving the function of indexing, and contain the OS in addition to Splunk software. Indexers receive data from forwarders, store it on disk in indexes, and return relevant results from the data they have when they receive search requests from the Search Head.

Dedicated or Shared indexers?

The shared indexer tier option can be considered whenever Splunk Enterprise is deployed on its own. The needs of the SOC and appetite for building added functionality on top are what drive the product decision. The deeper choice of whether to share the indexer tier once Splunk Enterprise by itself has been selected is driven by other aspects of the service – data segregation, performance guarantees etc. – described in the **rationale behind the multi-instance approach** to Splunk Enterprise Security.

As a *rough* rule of thumb, the shared indexer tier tends to be attractive to MSSPs whose customers are of relatively lower maturity (don't need everything in ES), when the MSSP has an existing workflow that they would like to retain completely or can build the small delta they need themselves (would like to take advantage of the workflow-enhancing features in ES in the future), and where customers tend to be smaller, with daily ingestion volumes in the tens of GB/day, rather than hundreds, so the combined ingestion volume of multiple customers can be handled by a single reference indexer (capable of indexing 300 GB/day assuming a certain search load). Consulting with Splunk is recommended here. In the meantime, here are some things to consider when opting for a shared indexer tier.

Advantages

The indexing tier of Splunk is sized in terms of a reference Splunk indexer, which is capable of about 300 GB/day of data ingestion (as of time of writing), assuming a certain amount of search load. Search load is important to consider for the indexers as well, since they do most of the work in searching, and most of the work they do is searching (not data ingestion). The search head accepts searches from the user, sends them outward to the indexers, and combines the partial results they send back for presentation to the user. If a typical customer falls below what a reference indexer is capable of, multiple customers could be handled by it.

Another advantage is avoiding the performance overhead of many virtual machines, each running a separate Splunk indexer, which would be another way to squeeze multiple customers into a single reference indexer while maintaining the approach of separate stacks per customer. These savings gained by eliminating VMs can matter in cases where margins are low enough.

The use of a single Splunk indexer also cuts down on configuration and ongoing management overhead, since only a single cluster and a smaller set of virtual machines is being maintained.

Disadvantages

Since data is segmented into indexes, not *indexERS*, this approach may not provide the data segmentation required. Indexes do segment the data physically – they are separate directories. Certain legal requirements or clients with very sensitive needs may demand machine-level segmentation, however, which calls for the separate stacks approach.

This approach also sacrifices another guarantee – protecting one customer environment from the performance impacts of another's workloads. The same cores and storage on the indexers are servicing everyone, after all. Recall that indexers participate heavily in searching, so that factors in here.

Similarly, this approach can also make it harder to safely maintain a small set of customer instances, while isolating the others. If changing one customer's environment requires restarting the indexers or changing global configurations, then all customers will be impacted. This is a common tradeoff with "multi-tenant" systems that typically ignore this aspect, relying simply on tagging the data separately.

Another reality that customers have independent lifecycles at the MSSP. They are acquired, upgraded to higher tiers, and decommissioned separately, so each customer may in practice be configured differently. The similarity is tied to the service being delivered. When all customers get the same thing, the shared indexer tier becomes more attractive. The acquisition/upsell/upgrade aspects for these essentially-identical customers may still make it a good idea to maintain separate environments, however, particularly as customers grow large.

A final consideration is that setting up the shared indexer option requires more attention to the details on an ongoing basis and calls for some other tasks that might be considered optional in a separate stacks approach.

1. Each customer's data must be uniquely **identified**. Hostnames, IP addresses etc. can overlap between customers, so it must be clear which customer (or business unit within a customer) the data belongs to.
2. Each customer's data must be **routed** correctly to indexes created and reserved solely for that customer. Data cannot be mingled, as this is a core requirement of the MSSP world, and reversing such mistakes is non-trivial.
3. Access to these indexes must also be **protected** in accordance with least-privilege practices. That applies to the dedicated stacks approach as well, but is relatively simpler, since data is not commingled.
4. The system must continue to **perform** well. Splunk is an advanced big data platform that can scale to ingest petabytes of data on a daily basis, and support searches across exabytes of data stored. The patterns for doing so are well-understood. As you might expect, maintaining large environments does require more knowledge and experience than normal. (The assumption here is that is that this shared environment would end up being fairly large – larger than an average dedicated environment one dedicated environment).

The dedicated stacks approach can be used for many small customer environments as well. It all depends on the tradeoffs deemed acceptable. In such cases, units of hardware smaller than the reference hardware sizes can be considered. Splunk tests all functionality on hardware that meets or exceeds the reference hardware recommendations, so no official recommendations are available on how small an environment can be made. Please consult Splunk for guidance here.

Designing a shared indexer tier

The following guidelines should be followed.

Dedicated Indexes per customer

The same indexer cluster will be used for all customers. Each customer's data should land in one or more **dedicated indexes** within the cluster. You can have as many indexes per customer as needed. There are many reasons for this.

1. Indexes are a logical container of data. They are the natural unit of data separation in Splunk. Segmenting data by indexes ensures no data cross contamination or leakage between customers. Note that this does not in any way compromise

search ability or data type commingling ability, since you can search across indexes, and different types of data can be stored in the same index.

2. RBAC in Splunk applies to indexes. You can easily specify who can access the data, and what access privileges they have. More on this in the RBAC section below.
3. Indexes can have their own data retention periods.
4. You can specify which indexes to send data to at the forwarder level, which simplifies management. This includes universal and heavy forwarders, and the HTTP event collector.
5. When indexes contain "like" data – say a category of data for this customer – searching across that category of data is faster.
6. Better performance compared to alternatives like search-time filters, which let you specify what a particular role can search for.

Adopt a naming convention for customers
For ease of data management (say if files are backed up manually) and easy visual identification and searching, a naming convention should be adopted to uniquely identify each customer's data. For example, there might be a customer ID or customer billing code, such as `CUST4893` or `CS4AB2Z`. If no such identifiers exist, the MSSP should create a unique nomenclature for use within Splunk. The naming convention has other advantages – it allows the MSSP to manage its business operations by obtaining easy statistics about customers (say trending over time, spotting the need for additional services etc.), or by comparing customers with each other.

Name the indexes per the convention
Every index's name should clearly reflect which customer's data it contains. This results in indexes with names such as `CUST4893_linux`, `CUST4893_windows`, or `CUST4893_cisco`.

Name the customer in each raw event
Each raw event should reflect which customer it belongs to in a way that cannot be missed by the analyst at search time. This affords the analyst flexibility by allowing them to search single customers or across customers as necessary, while easing the task of understanding the results presented. The recommendation is to add an *index-time field* called customer to every event at the forwarder level, whose value is the customer ID for that customer. This will be added alongside the index-time metadata Splunk adds to every event by default - source, sourcetype and host. Adding one field does not increase the storage requirements much. Search-time fields can be easily overwritten at search time, so are not recommended here. Filtering based on index-time fields is also faster.

Name the customer in lookup data
Raw data can be enriched with data looked up from other databases to make the analyst's task easier, and also offer some unique analytical abilities. Like the raw events, lookups used to enrich the raw data should also specify which customer they belong to. Lookups could include a customer field, but it may be simpler to have separate lookups.

Create catch-all indexes
Create a per-customer catch-all index that serves as a canary to catch any data that is not correctly routed to the customer-specific indexes. When everything is working well, this should never contain any data. If anything appears in here, it indicates either misconfiguration at the forwarder/ingestion level, or a mistake in the index setup.

Searching Tier

The search head provides the user interface to interact with. All queries, reports and dashboards are invoked and presented here. The architecture of this tier depends on which product is being deployed.

With Splunk Enterprise Security

High availability
When Splunk Enterprise Security is being used, each customer requires a dedicated Splunk ES search head. Each customer also has their own indexers when ES is deployed, so the entire stack will be per-customer.

For high-availability of the Splunk ES search head, there are a couple options. Splunk natively offers search head clustering to provide HA of the search tier. If one search head goes down, the user can automatically be redirected to another functioning one. Search head clustering is supported with Splunk Enterprise Security. This is an option for advanced customers only since the process of deploying ES

on a search head cluster (SHC) is more involved than deploying Splunk Enterprise itself. Consider this if you have strong Splunk skills in-house and your HA requirements are stringent enough that this is the only way to go. For most situations, we do not recommend running ES on a search head cluster due to the administrative overhead.

The easier to maintain option is to have a cold backup search head running Splunk Enterprise Security. By cold, this means that the search head does not participate in the searches, and does not accelerate data on its own. It can be synced up with the active "hot" Splunk ES search head via normal VM backup mechanisms. Should the primary search head go down, this can be brought online quite quickly. This is the option we recommend for most MSSPs running ES.

Per-customer views

Since each customer has their own Splunk ES search head, it will only display data for that customer. No additional work is necessary to gain per customer views.

Views across multiple customers

There are two major use cases that make a combined view across multiple ES instances – a multi-tenant *experience* for the SOC – useful to an MSSP.

1. Centralized visibility of notable events and health of individual environments
2. Centrally searching across multiple Splunk installations to see if a threat is affecting other customers.

Analysts can log into each customer's instance and keep an eye on them in the normal way – multiple windows or monitors. This is fine for a few instances. Beyond that, the recommended approach is to create a custom multi-customer view to show exactly what the MSSP needs. These requirements will differ from MSSP to MSSP, so a single layout is unlikely to satisfy everyone.

The overall view may choose to pull in notable events from all customers, additional contextual information present in Splunk aside from notables, SOC analyst notes from the investigator journal, metrics data from Splunk ES, detailed information on investigation steps followed (perhaps from the Splunk ES Investigation Timeline), and other things. The choice depends on what tasks the analyst intends to perform in the overall view. It is rare for this to be the primary investigative point for all depths of an investigation, for example. It often serves as a way to look at trends across multiple customers to identify outbreaks, or to look at all customers at a basic level, however the MSSP defines that. After seeing a customer worth paying more attention to, analysts may jump into the individual instance to follow the trail. Depending on the workflow, they may instead file a ticket in a workflow system for another analyst to take a look at it. Other times, the goal is different – ensuring that expected data is flowing in from the customer without issue so the SOC has what it needs to provide their service. It is also common to need a higher-level health indicator of each customer – how healthy their Splunk environment is. Splunk is highly flexible in the types of questions it lets you ask, so all of these are common scenarios for MSSPs. Everyone's requirements are different, however. It a good idea to work with Splunk on designing the overall view so it integrates smoothly into the SOC's workflow, taking advantage of the collaboration features in Splunk ES and the remediation possibilities of the Adaptive Response framework and perhaps Splunk Phantom, if needed. This ensures that Splunk is able to bring its experience and best practices across similar environments to bear.

The SOC staff will also need access to systems beyond Splunk, such as ticketing and workflow management tools, threat intelligence platforms, internal wikis, external visualization tools, customer information databases, or custom runbooks or other internal tools commonly used by the SOC.

Approach 1 – Design a custom web portal to provide the overall view

A custom web application providing a view that meets all these needs is often the right solution here. Splunk offers excellent API and SDK support to facilitate this, so the central web portal can be developed in any web toolkit of the MSSP's choice. Splunk professional services can also partner with the MSSP on this effort.

Approach 2 – Use a central Splunk instance to provide the overall view

An alternative to a custom web portal is to set up a separate Splunk instance to serve as a central view. This is a normal Splunk Enterprise instance, intended solely for this purpose. It is not another ES instance. Splunk is distributed in nature, so setting up this kind of higher-tier instance to look across many Splunk instances is very common.

This *central view* will aggregate data from the individual ES instances in the precise manner that the SOC finds useful. It can be used to provide overall visibility into notable events for all customers or depending on how it communicates with the individual ES deployments, to conduct cross-customer investigations. Either pull- or push-based approaches can be used to populate the data it needs. In the pull approach, it will need network connectivity to the indexers on the individual instances, as well as the appropriate level of access for the data it needs. In the push approach, each Splunk ES installation will grab the data the overall view needs, using scheduled searches and similar, and periodically push the data up either using native Splunk mechanisms or via standard means like HTTP.

When all data is available in a single view, it becomes very important for the analyst to easily be able to tell whose data they are looking at in the moment. All the recommendations in **designing a shared indexer tier for Splunk Enterprise-only deployments** are useful here. Things like the index-time `customer` field that is present in every event, for example, is extremely useful when the overall view contains raw data as well, not just high-level metrics or notables.

Conceptually, the central Splunk instance approach looks like this:

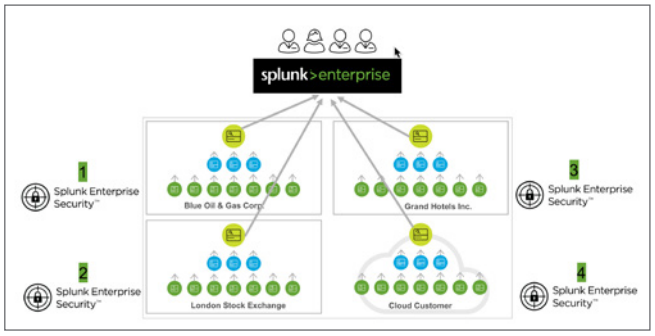

Figure 9 – Central Splunk Enterprise instance to look across multiple customers

With Splunk Enterprise only (no ES)

High Availability

When Splunk ES is not being used, search head clustering is a perfectly valid option for HA of the search tier. We recommend a search head cluster in this case.

Per-customer views

With **dedicated stacks**, each customer has their own search head (cluster), so no additional work is needed to get per-customer views.

With a **shared environment**, the search head (cluster) has access to all customer data. Restricting the view to a single customer is achieved by RBAC – what the user sees depends on the user they are logged in as. We recommend taking this a step further by creating a separate app for each customer, to display on that customer's data. It can display only their machine data, plus enrichment data pulled in from other systems as necessary. This pattern is required for MSSPs who choose to offer their customer access to the Splunk UI. Splunk dashboards are very customizable, so there is no requirement that all apps be identical. Customers can easily have different views of their data if the MSSP so desires. This is also useful to roll enhancements out in a phased manner, to experiment with "beta" views while maintaining the old one, and to offer SOC analysts a different view from the customer (they can have their own app).

Views across multiple customers

With a dedicated stacks approach, the approach is

similar to that recommended for Splunk Enterprise Security. A central Splunk instance or an external portal that leverages the Splunk API can both work.

With a shared environment, since all data is in one place, no additional system is necessary. The default search view can provide access to all data as long as the user has the right permissions. An overall view app could also be created to tailor the presentation to the most common tasks that need to be performed.

Role Based Access Control (RBAC)

MSSPs may have restrictions on which SOC analysts can view a customer's data. Analysts could be restricted to only accessing certain customers, for example. Customers may require analysts to hold certain security clearances, possess citizenship of a certain country or be located in a particular region. Splunk's strong hierarchical RBAC facilities can accommodate these flexibly accommodate these requirements (see docs here).

The below are **required** for a shared Splunk Enterprise deployment since all data is being held in the same cluster. Deploying RBAC carefully can segregate each customer's data, while allowing granular access by admins and analysts to preserve least-privilege concerns. In a dedicated stacks approach, data is completely segregated and not commingled, but to control and audit access by personnel, these recommendations should still be followed. This also has the advantage of making it easier to migrate to a shared indexer tier in the future if needed.

Users

A **separate user** should be created for each customer and used every time that user's data is accessed. This user should have a single customer-specific role assigned to it (described next). should be a corresponding **customer-specific user** with only the single customer role, per customer. This user should be used for all human or automated logins to the Splunk installation. As with indexes, adopt a clear naming convention that includes the customer identifier to ease ongoing privilege management and later auditing of access used anytime the SOC staff or any automated solution needs to safely interact with a particular customer's data. Using clear naming conventions for the user accounts and roles is crucial for proper management in a shared environment and for clarity in any automation code in systems outside Splunk.

Roles

A **separate role** should also be created for each customer to govern access to that customer's data. Each indexer in either a dedicated or shared environment should have a **customer-specific role** created, intended to control access to that particular customer's data. As with users and indexes, is recommended that roles be named with a unique customer identifier., such as the customer ID mentioned earlier. Each customer-specific role must be restricted to only searching the customer's indexes. For example, the role `cust4893` would be restricted to searching the indexes **`CUST4893_linux`**, **`CUST4893_windows`**, **`CUST4893_cisco`**, and any other index whose name begins with **`CUST4893`**.

In the **dedicated full stacks** approach, the indexers are dedicated to a customer, and all indexes within contain only that customer's data. The customer role can therefore be allowed to access any all indexes, though it is good practice to restrict access to the internal indexes used by Splunk itself. Access to those should be reserved for administrators.

In a **shared indexer tier** environment, the indexers are not dedicated to a customer, and each customer's data is segmented into a set of indexes that are reserved for them.

It is good practice in both indexer architectures, however, to restrict access to the internal indexes used by Splunk itself - _internal, _audit etc. Access to those should be reserved for administrators. In most SOCs, Splunk administrator and SOC analyst duties are separated, so SOC analysts should not be given access to the internal logs and indexes. This prevents routine investigative work from inadvertently quashing the logs in the internal indexes. To accommodate this requirement, each customer should have a separate administrative role as well.

Deployment, Management and Integration With SOC Workflow

Deployment, Orchestration and Management

When multiple Splunk environments exist (the norm), each is operated largely as any independent Splunk installation. "Largely", because as the number goes up, it is more likely that configuration management and automation tools will be deployed to configure multiple environments at scale, rather than configuring each individually using Splunk's built-in mechanisms.

Third-party IT orchestration tools are the best way to ensure consistency, reliability, predictability and rollback procedures for multiple Splunk environments. Any tool the MSSP is comfortable with is fine. These tools provide many of the advantages an MSSP seeks – lower cost of delivery, faster onboarding and time to values and revenue, greater customer satisfaction when changes are smooth, the ability to build once and deploy many times.

The basic tools required in such a deployment and orchestration system are:

- A version control system (VCS, like git or subversion) to store configurations and other artifacts.
- A deployment and orchestration infrastructure (Chef, Ansible, Puppet etc.) to configure multiple Splunk systems at scale, support rollback, and support per-client customizations.

The third-party tools will augment the many central management tools that Splunk provides. The **Monitoring Console**, for example, serves as a central health monitor for a Splunk deployment. The **Deployment Server** can configure forwarders remotely at scale. The **Deployer** is used to configure all the members of a Search Head Cluster, and so on. These are all thoroughly documented in the **Distributed Deployment Manual**, among places, and have extensive support from Splunk and the extraordinarily vibrant **Splunk Community** of thousands of users. All of these tools are designed to configure a single single Splunk environment, however, and require that Splunk itself be running. A fully automated deployment will need to handle multiple Splunk systems and bootstrap things before Splunk is running. For example, spinning up the VM or cloud instance (e.g. EC2), installing Splunk, starting it up, pulling configurations from version control to populate things like the Deployment Server, and then invoking the built-in Splunk tools on each deployment to configure the cluster fully.

It should be noted that the best place for the orchestration tools is to augment what exists in Splunk, not duplicate it entirely. The built-in tools understand the underlying workings of a Splunk deployment much better and are fully supported, so should be leveraged as far as possible.

To start the process, the MSSP must design a gold standard customer environment, then copy all the configuration files and other artifacts needed for it, such as logos and other custom art, into the VCS. Then a mechanism must be devised within it to automate the creation of branches, copies, or other means for providing varied configurations for different customers. All customer deployment configurations (say, correlation rules) and operational management changes should be stored within the VCS. This has many advantages:

- Rapidly recreating them for disaster recovery purposes.
- Easier testing and rollback of changes.
- Traceability for change control purposes. This helps the SOC narrow down which version of a search affected what they are seeing today, should they suddenly stop seeing things they used to.

The simplest way to reduce errors and understand exactly what went into a customer environment is to push all changes from the VCS and not make any changes directly on the individual Splunk systems or environments. Changing individual systems makes change management and control much harder at scale.

Creating a new customer environment requires forking, branching or similar from the gold standard, followed by any changes to those files to accommodate the new customer. For example, the customer ID used for naming roles, accounts, and indexes, as well as possibly systems, must be changed from the generic defaults. The changes must also **size each Splunk environment appropriately** for the expected data ingest volume and search load. They must also install licenses, though Not-For-Resale licenses can be provided for R&D purposes.

The whole operation then must be integrated into

the MSSP's custom workflows and other systems, so the final system can be used by the SOC efficiently. The SOC must have the new customer environment information added to ticket systems, workflows, analysts' displays, and any other systems developed to maintain operations.

This list of systems to integrate Splunk with will likely will include a reporting portal of some sort, when Splunk itself is not serving this purpose. Integration with this portal or other portals used by the SOC team may require a **gateway** of some sort. This is depicted in the diagram below. It is important to understand that the gateway is not a Splunk component or even required. It's more of a conceptual idea that some kind of authentication or pass-through may be required before the portals can make API calls to Splunk.

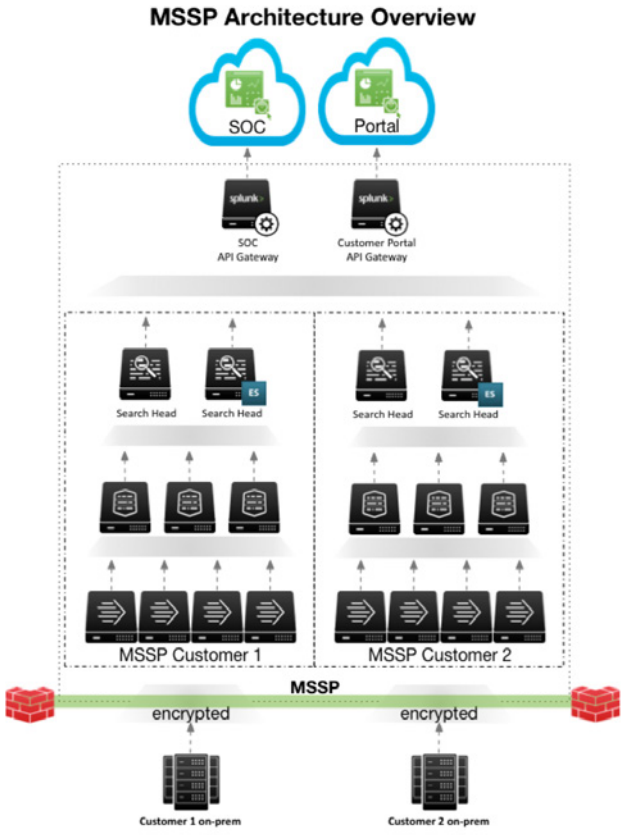

Figure 10 - Communication with portals and SOC

Summing up, spinning up new customer environments can include:

- Creating a server, either virtual or physical
- Installing the OS
- Installing the deployment and orchestration packages
- Installing the Splunk packages
- Using the deployment and orchestration tool to install the customer-specific configurations from the VCS
- Integrating this Splunk environment into the SOC's existing workflows and systems

As with any engineering effort, it is highly recommended to create a pre-production or development environment to test out the entire process. This should mimic the final production environment as far as possible, including realistic data ingest from customers.

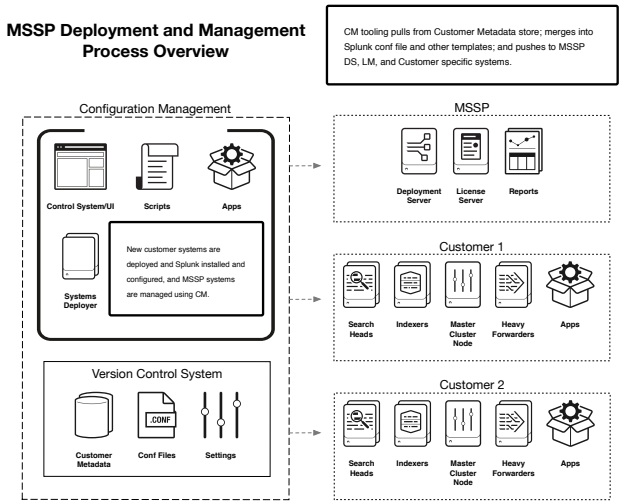

Figure 11 - Deployment and Management Process

Integration With SOC Workflow

Properly deploying Splunk to be the nerve center of a SOC involves tying it into the SOC's overall workflow – systems outside Splunk. Splunk is intentionally designed to be an open, extensible platform that can work alongside many, many others. Ticketing systems, threat intelligence platforms, all manner of security and IT products, single sign-on systems, and more can be integrated with either via longstanding native Splunk capabilities, Splunk Enterprise Security's Adaptive Response framework, or full-fledged orchestration and automation platforms like Splunk Phantom.

Programmatically, this is also very possible via Splunk's extensive APIs. **The Splunk Developer site** is an excellent source of information on this.

More questions?

For questions, reach out to Splunk at **sales@splunk.com** or +1.866.GET.SPLUNK (1.866.438.7758)