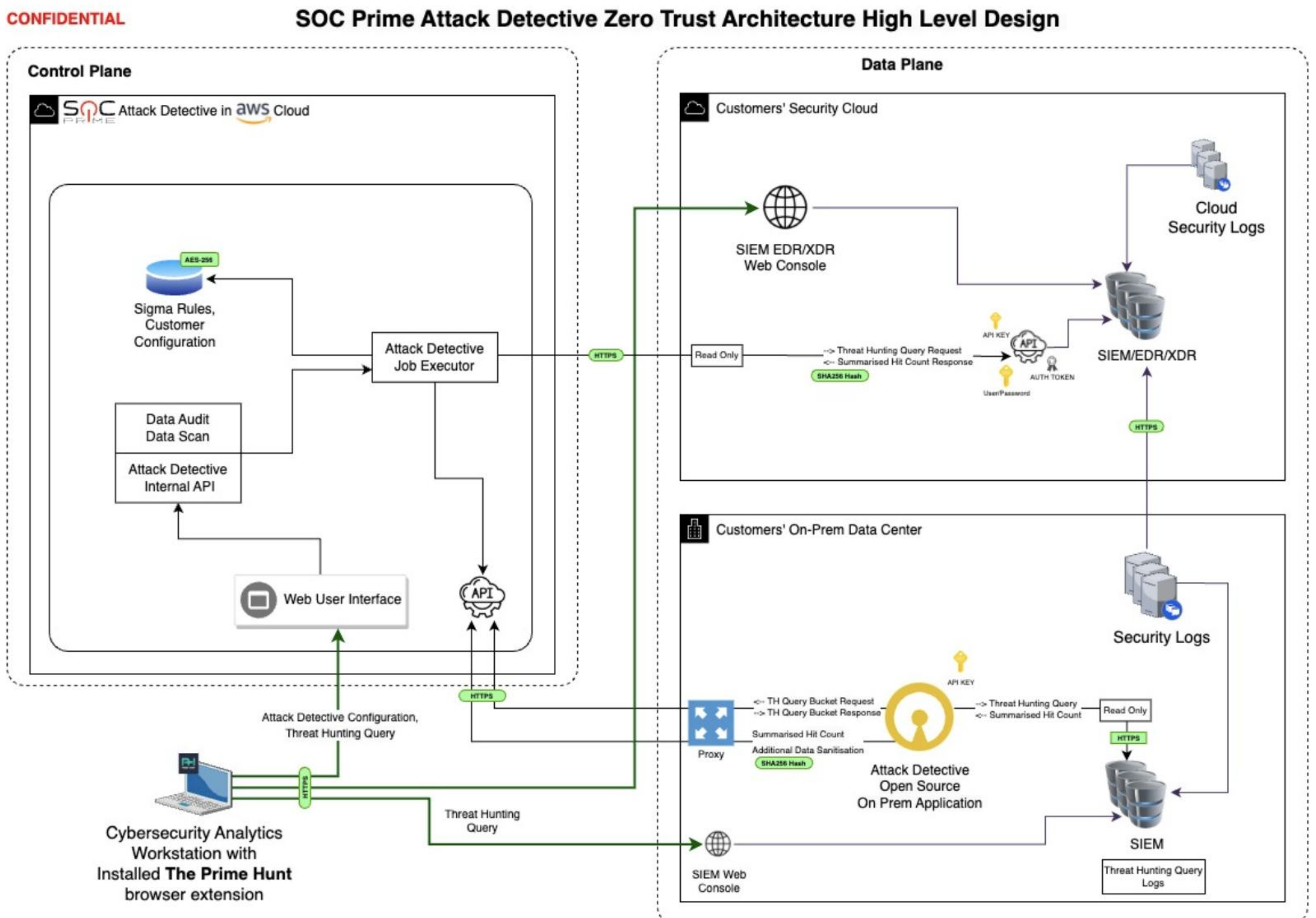
Attack Detective
Data Privacy

Key points

- Read-only access to your data
- Keeping the data where it lives
- Microservice-based architecture
- Amazon AWS hosting
- Retrieved data are encrypted at rest in AES-256
- Data is transferred via a secure HTTPS channel, TLS v1.2

High-Level Diagram


CONFIDENTIAL
SOC Prime Attack Detective Zero Trust Architecture High Level Design
Control Plane
SOC PRIME Attack Detective in aws Cloud
AES-256
Sigma Rules, Customer Configuration
Attack Detective Job Executor
Data Audit Data Scan
Attack Detective Internal API
Web User Interface
API
HTTPS
Attack Detective Configuration, Threat Hunting Query
HTTPS
Cybersecurity Analytics Workstation with Installed The Prime Hunt browser extension
Threat Hunting Query
Data Plane
Customers' Security Cloud
SIEM EDR/XDR Web Console
Cloud Security Logs
Read Only
--> Threat Hunting Query Request
<-- Summarised Hit Count Response
SHA256 Hash
API KEY
API
AUTH TOKEN
User/Password
SIEM/EDR/XDR
HTTPS
Customers' On-Prem Data Center
Security Logs
API KEY
<-- TH Query Bucket Request
--> TH Query Bucket Response
Summarised Hit Count
Additional Data Sanitisation
SHA256 Hash
Proxy
Attack Detective Open Source On Prem Application
--> Threat Hunting Query
<-- Summarised Hit Count
Read Only
HTTPS
SIEM
Threat Hunting Query Logs
SIEM Web Console

Ensuring Data Privacy Protection

Attack Detective stores only aggregated information about the triggered rules, like hit count grouped by 1-hour bins and unique hashes (SHA256) for usernames and hostnames to show asset and account counts in the triggered queries. No SIEM log source data is collected or stored in Attack Detective.

For the Data Audit, table (index) names and unique values of event IDs are stored to calculate the ATT&CK coverage and gaps (blind spots)[1].

SOC Prime infrastructure is SOC 2 Type II certified, which ensures that appropriate access restrictions to the databases for the infrastructure administrators are in place.

Attack Detective is built on the Zero Trust Architecture (ZTA) milestones enabling organizations to risk-optimize their cybersecurity posture. It provides complete visibility based on the organization-specific logs to query data in its native location. This enables avoiding data duplication or distribution and possible permission inconsistency for the same data across different locations, which ensures compliance with zero-trust basic tenets and is aligned with the least privilege principles according to the operative definition of ZTA as per NIST SP 800-207 recommendations.

[1] *We plan to release a Content Audit module in August 2024. To enable customers to take advantage of this functionality, Attack Detective should have access to the customer's SIEM content (rules and queries) to provide a content audit.*

Data Flow Details

An investigation includes two stages: Data Audit and scanning:

- During the Data Audit, we automatically analyze log data collected in your Data Planes to determine your MITRE ATT&CK® coverage and potential gaps in log sources. Attack Detective sends audit queries to your Data Plane (SIEM, EDR, or Data Lake) via API (with read-only privileges). The queries are to:
  - Collect the aggregated statistics on the number of accounts and assets (without their values). Required to build insights on affected accounts and assets during further investigation.
  - Identify available log sources. It is required to map log sources to MITRE ATT&CK, find blind spots according to our dictionaries, and choose detection content relevant to your log sources.
- Scanning involves querying your logs for the selected period with a set of queries from the SOC Prime Platform that are relevant to your Data Planes. Attack Detective sends hunting queries to your Data Plane (SIEM, EDR, or Data Lake) via API (with read-only privileges). The queries are to retrieve aggregated data on hits. **We do not collect affected accounts or assets; we only summarize stats.**

Once the investigation results are ready, the user can review them in Attack Detective. To verify and further investigate any hit, the user launches hunting queries with the Hunt option:

- The queries are passed to the URL to be directly run in the SIEM, EDR, or Data Lake
- To find affected assets, account names, and IPs, the user does further review and investigation directly in the SIEM, EDR, or Data Lake

Optionally, for more efficient threat hunting while further investigation is performed in the SIEM, EDR, or Data Lake, the user can use The Prime Hunt browser add-on. This would allow for getting the query in a more convenient way. The add-on does not send out any data. The entire process is carried out in the user's browser.

Querying the Data Plane

During the Data Audit, Attack Detective gathers the information needed to determine log sources and events collected, check if the needed fields are not empty[2], and get a count of unique assets and accounts in the system.

For sensitive data like assets and accounts, a one-way hashing function (SHA256) is applied to mask the actual value while still maintaining uniqueness in Attack Detective. If the platform does not support such a hashing function in the query itself, all values are hashed in Attack Detective and stored in the database as hashes.

Amazon Athena

A `GROUP BY` function is added to all queries to group the results by the 1-hour split and unique hashes of the users and hosts:

```
SELECT count(eventHour) AS count_, eventHour as eventHour,
to_hex(SHA256(CAST(identity.user.name AS VARBINARY))) as AccountNameHash
FROM {table} WHERE {Query Condition} AND eventHour BETWEEN {interval_start}
AND {interval_end} GROUP BY eventHour, identity.user.name
```

Chronicle Security

Due to the Chronicle API limitations, aggregation functions are not supported. All query results are processed and grouped in Attack Detective. Query results are grouped into 1-hour spans, and only the query results count for each hour is stored in the database. User and host values are hashed with the SHA256 function in Attack Detective and stored in the database as hashes.

[2] Not available for some of the supported platforms due to certain technical limitations of the corresponding SIEMs/EDRs/XDRs/ Data Lakes.

Elasticsearch

An aggregation (aggs) function is added to all queries to group the results by the 1-hour split and unique values of the users and hosts. User and host values are hashed with the SHA256 function in Attack Detective and stored in the database as hashes.

```
{ "query": { "bool": { "must": [ {"query_string": {"query": {query},
"analyze_wildcard": True}}, { "range": { self._fields.timestamp: { "gte":
{period_start}, "lt": {period_end}, "format": "yyyy-MM-dd HH:mm:ss", } } },
] } }, "aggs": { "main": { "composite": { "sources": [ { "timestamp": {
"date_histogram": { "field": {timestamp}, "format": "date_time",
"calendar_interval": "1h", } } }, {"hosts": {"terms": {"field":
{host_name}, "order": "asc", "missing_bucket": True}}}, {"users": {"terms":
{"field": {user_name}, "order": "asc", "missing_bucket": True}}}, ] } } },
}
```

Falcon LogScale

A groupby function is added to all queries to group the results by the 1-hour split and unique hashes of the users and hosts:

```
{query} | hashRewrite(windows.Computer, salt={salt}) |
hashRewrite(windows.EventData.User, salt=socprime) | formattime(\"%d.%m.%Y
%H:00\", as=timehour, field=@timestamp) |
groupby([timehour,windows.Computer,windows.EventData.User])"
```

IBM QRadar

A GROUP BY function is added to all queries to group the results by the 1-hour split and unique values of the users and hosts. User and host values are hashed with the SHA256 function in Attack Detective and stored in the database as hashes.

```
SELECT sourceip AS 'asset', count(sourceip) AS 'count', starttime as
timestamp FROM events WHERE {Query Condition} GROUP BY starttime/36000000,
sourceip START {date_start} STOP {date_finish}
```

Microsoft Defender for Endpoint

A `summarize` function is added to all queries to group the results by the 1-hour split and unique hashes of the users and hosts:

```
{query}
| where {fields.timestamp} between ({interval_start} .. {interval_end})
| summarize count() by bin({fields.timestamp}, 1h),
DeviceNameHash=hash_sha256({fields.host_name}),
AccountNameHash=hash_sha256({fields.user_name})
```

Microsoft Sentinel

A `summarize` function is added to all queries to group the results by the 1-hour split and unique hashes of the users and hosts:

```
{query}
| where {fields.timestamp} between ({interval_start} .. {interval_end})
| summarize count() by bin({fields.timestamp}, 1h),
DeviceNameHash=hash_sha256({fields.host_name}),
AccountNameHash=hash_sha256({fields.user_name})
```

OpenSearch

An aggregation (`aggs`) function is added to all queries to group the results by the 1-hour split and unique values of the users and hosts. User and host values are hashed with the SHA256 function in Attack Detective and stored in the database as hashes.

```
{"query": { "bool": { "must": [ {"query_string": {"query": {query},
"analyze_wildcard": True}}, { "range": { self._fields.timestamp: { "gte":
{period_start}, "lt": {period_end}, "format": "yyyy-MM-dd HH:mm:ss", } } },
] } }, "aggs": { "main": { "composite": { "sources": [ { "timestamp": {
"date_histogram": { "field": {timestamp}, "format": "date_time",
"calendar_interval": "1h", } } }, {"hosts": {"terms": {"field":
{host_name}, "order": "asc", "missing_bucket": True}}}, {"users": {"terms":
{"field": {user_name}, "order": "asc", "missing_bucket": True}}}, ] } }},}
```

Splunk

A `stats` function is added to all queries to group the results by the 1-hour split and unique hashes of the users and hosts:

```
{query} | bin {_time} span=1h | stats count by {_time}, {User}, {host} |
eval sha256_host=sha256({host}) | eval sha256_user=sha256({User}) | table
{_time}, count, sha256_host, sha256_user
```

Sumo Logic

A `count by` function is added to all queries to group the results by the 1-hour split and unique hashes of the users and hosts:

```
{query} | parse "\"WorkstationName\":\"*\"," as Workstation nodrop |
hash("Workstation","sha2_256") as WorkstationHash | parse
"\"TargetUserName\":\"*\"" as AccountName nodrop |
hash("AccountName","sha2_256") as AccountHash | timeslice by 1h | count by
_timeslice , WorkstationHash , AccountHash
```