ExtraHop

WHITE PAPER

How **Reveal**(x) Detects Threats

Applying Machine Learning for Network Behavioral Detection

This white paper explains how ExtraHop Reveal(x) more accurately detects attacks by applying machine learning to a superior data set. Unlike many network traffic analysis (NTA) products that rely primarily on flows and packet header information, Reveal(x) draws on the industry's richest set of network behavioral machine learning features, building sophisticated models and powering proprietary algorithms. Detections in Reveal(x) cover all stages of the attack life cycle, including command and control, reconnaissance, exploit, lateral movement, and actions on objectives.

TABLE OF CONTENTS

Advancing the State of the Art in Network Threat Detection ... 3
How ExtraHop Reveal(x) Detects Threats ... 4
- Examples of Application-Layer Machine Learning Features ... 5
- Machine Learning Models ... 7
  - Predictive Models ... 7
  - Importance and Privilege Inference Models ... 8
  - Peer Group Models ... 8
- Additional Threat Visibility ... 8
- Customizing Detections and Integration ... 9

Examples of Detections in Reveal(x) ... 10
- Command & Control ... 10
  - DNS tunneling ... 10
  - ICMP tunneling ... 10
  - Reverse shell to external IPs ... 10
  - Suspect IPs, domains, and URIs ... 11
  - Unusual inbound connections ... 11
- Reconnaissance ... 11
  - Active Directory reconnaissance ... 11
  - Address and port scans ... 11
  - DNS reverse lookups ... 11
  - Web scans ... 12
  - VOIP scans ... 12
- Exploit ... 12
  - Brute force attacks ... 12
  - DNS rebinding attacks ... 12
  - IDS evasion ... 12
  - Kerberos attacks ... 12
  - LLMNR and NetBIOS poisoning ... 13
  - SQL injection attacks ... 13
  - WPAD attacks ... 13
- Lateral Movement ... 13
  - Abnormal internal data transfer ... 13
  - Abnormal network activity (peer group) ... 13
  - Abuse of machine accounts ... 14
  - Network privilege escalation ... 14
  - PsExec remote command tool ... 14
  - Windows RPC misuse ... 14
- Actions on Objectives ... 14
  - BitTorrent activity ... 14
  - Cryptomining activity ... 14
  - Data exfiltration to external IPs ... 14
  - Data smuggling ... 15
  - Ransomware activity ... 15

Advancing the State of the Art in Network Threat Detection

The promise of detecting threats in network traffic has tantalized the cybersecurity industry for decades and has recently been rejuvenated with machine learning. Because attackers must use the network in nearly every attack, the network is an ideal point of instrumentation for detecting malicious behavior.

Wire data on the network records the activity of all devices regardless of whether they are managed or unmanaged. This means that wire data can capture rogue device activity that IT and security teams would not otherwise be aware of because they do not have logging or endpoint agents configured for those devices. It's worth noting that unlike log and endpoint data—which can be tampered with or deleted and is not available in real time—wire data on the network is ground truth and cannot be turned off.[1] It's like a closed-circuit security camera for your data center.

> "One of our worst nightmares is that out-of-band network tap that really is capturing all the data, understanding anomalous behavior that's going on, and someone's paying attention to it."
>
> Rob Joyce, Head of Tailored Access Operations, U.S. National Security Agency. Joyce is speaking from the point of view of one of the world's most sophisticated hacking organizations.[2]

Network traffic analysis augments intrusion detection systems (IDS) and firewalls, which rely on signatures to detect threats, by applying machine learning to identify attack behaviors in traffic coming in and out of the environment—often dubbed north-south traffic.

However, the state of the art in network traffic analysis lies in detecting attack activity contained in east-west traffic inside the environment. This is partly because rules and signatures are less effective when detecting subtle post-compromise or insider activity such as malicious use of Windows remote procedure calls, abnormal behavior from low-privileged devices or users, abuse of machine and privileged accounts, and abnormal internal data transfer. In these cases, it is critical to understand the normal patterns of behavior and identify anomalies that could indicate malicious activity.

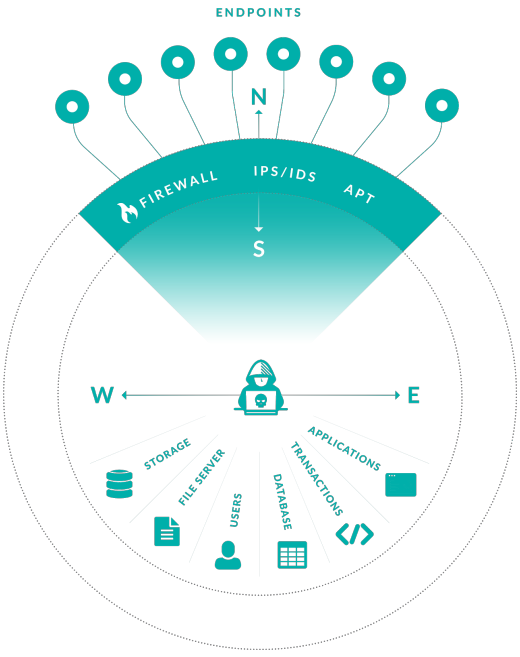


Figure 1. *The east-west network traffic inside the perimeter is typically darkspace that traditional detection techniques do not adequately address.*

[1] In more than half of the incidents observed in the second and third quarters of 2018, attackers had deleted antivirus and security logs according to the Carbon Black 2018 Incident Response Survey: https://www.carbonblack.com/2018/10/30/carbon-black-report-destructive-cyberattacks-increase-ahead-of-2018-midterm-elections/

[2] NSA Hacker Chief Explains How to Keep Him Out of Your System, January 2016, Wired.com: https://www.wired.com/2016/01/nsa-hacker-chief-explains-how-to-keep-him-out-of-your-system/

How ExtraHop Reveal(x) Detects Threats

ExtraHop Reveal(x) employs sophisticated machine learning to detect subtle attack behaviors on the network that other solutions miss. Reveal(x) is able to access many machine learning features that other network traffic analysis products cannot, and the quantity and relevance of features available to Reveal(x) is an important part of what sets its threat detection capabilities apart in the market.

Feature engineering is one of the most critical tasks in creating a machine learning solution. Features are a measurable property of the behavior observed. In speech recognition algorithms, the length and relative power of sounds are features. In spam detection algorithms, the frequency of specific phrases and grammatical errors are features. For behavioral threat detection on the network, features can be simple network details such as the IP addresses or packet sizes, or can include richer and more meaningful application-layer details such as an error message or file name.

> "At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used."
>
> Pedro Domingos, professor of computer science and engineering at the University of Washington and the author of The Master Algorithm.[3]

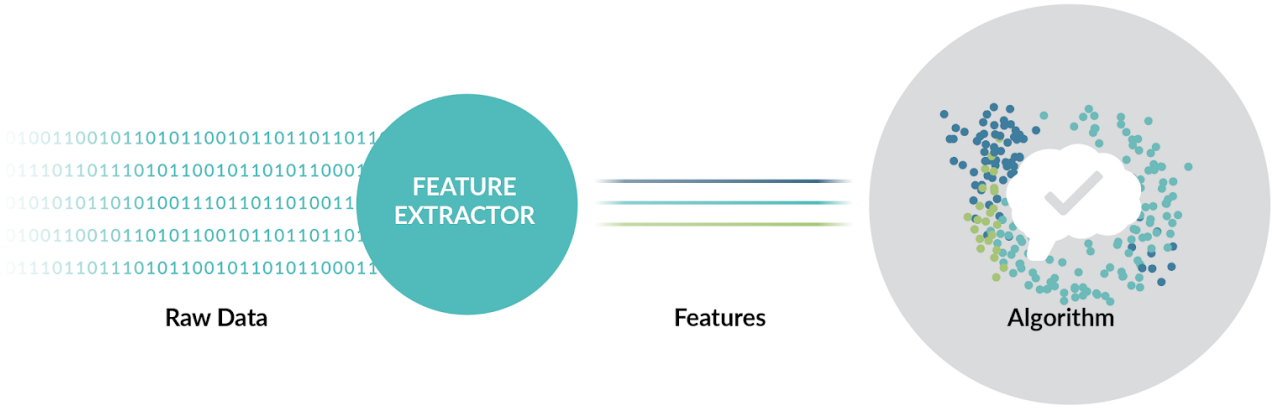


Figure 2. *Features are crucial to the success of machine learning algorithms and must be extracted from raw data.*

Reveal(x) extracts more than 4,600 features from Layers 2 through 7 of the OSI stack. This wealth of in-depth data is crucial for ensuring the accuracy of Reveal(x) detections. But it's not just the number of features that matter. Reveal(x) is also able to access highly relevant application-layer features from network traffic because of its proprietary TCP stream reassembly capabilities. This involves recreating the TCP state machines for all clients and servers on the network and using those to reassemble packets into their full streams. This is the prerequisite to analyzing the full content of the streams and extracting the application-layer details, such as methods, errors, operations, and commands.

[3] A Few Useful Things to Know about Machine Learning: https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf

For encrypted network traffic, Reveal(x) offers real-time, out-of-band decryption at speeds of up to 100 Gbps, including options for decrypting traffic protected by perfect forward secrecy (PFS) cipher suites. Decryption is needed to detect L7 attacks such as SQL injection, LDAP attacks, and cross-site scripting where the important features are not seen if hidden inside an encrypted transport.

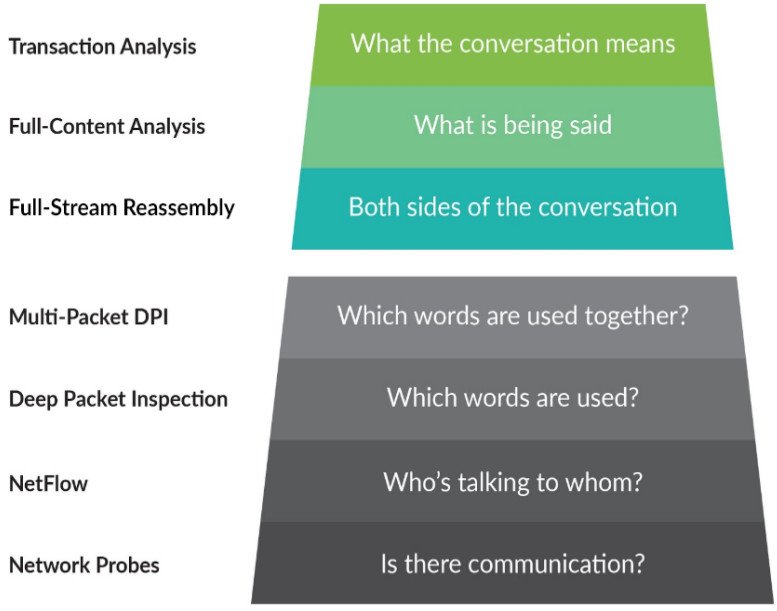


Figure 3. *To understand the full meaning of conversations between machines in network traffic, Reveal(x) performs full-stream reassembly, where the packets are reassembled into the full transactions, flows, and streams. This enables full analysis of transaction contents and a deeper understanding of the conversations between devices.*

Examples of Application-Layer Machine Learning Features

Most NTA products offer limited application-layer (Layer 7) analysis, which means that they do not have access to highly relevant machine learning features. Below are some examples of the application-layer features that are available to Reveal(x) along with examples of how they can be used to detect malicious behavior.

Note: Reveal(x) parses many more application-layer protocols in addition to those listed below, which are provided for the purpose of illustrating the types of machine learning features that are only available after decryption and full-stream reassembly.

Database Application-Layer (Layer 7) Details: Methods, Errors, Requests, Responses, SQL Statements

Example of why it matters: Database error messages are important for accurate detections of brute-force attacks against the database server.

Time: 2018-11-28 22:02:35.226, **Record Type:** DB, **Flow:** edabbb185bff80fb, **Client:** bullseye, **Client IPv4 Address:** 172.23.1.101, **Client Port:** 54553, **Server:** mysql1.lonprod.example.com, **Server IPv4 Address:** 172.23.2.33, **Server Port:** 3306, **Client Zero Windows:** 0, **Server Zero Windows:** 0, **Method:** OTHER,
Error: #28000Access denied for user 'root'@'bullseye.londmz.example.com' (using password: YES), **Request Aborted:** False, **Response Aborted:** False, **Request Bytes:** 84, **Request L2 Bytes:** 356, **Request Packets:** 4, **Request RTO:** 0, **Request Size:** 0, **Round Trip Time:** 0.422, **Response Bytes:** 182, **Response L2 Bytes:** 454, **Response Packets:** 4, **Response RTO:** 0, **Response Size:** 0, **ex:** {"isSuspicious:b":false}

Kerberos Application-Layer (Layer 7) Details: User Principal Names, Service Principal Names, Request and Response Message Types, Error Types

Example of why it matters: Kerberos duplicate ticket errors are important for accurately detecting when an attacker has stolen a ticket and is replaying it. Kerberos requests asking for a service principal name (SPN) that does not exist are important for detecting username scans.

Time: 2018-11-29 03:24:18.701, **Record Type:** Kerberos Response, **Flow:** 0303e1d85bffcc62, **Client:** VMware 192.168.6.183, **Client IPv4 Address:** 192.168.6.183, **Client Port:** 63322, **Server:** 10.10.10.3, **Server IPv4 Address:** 192.168.6.179, **Server Port:** 88, **Client Zero Windows:** 0, **Server Zero Windows:** 0, **Message Type:** KRB_ERROR, **Error:** KDC_ERR_S_PRINCIPAL_UNKNOWN, **Server Name Type:** NT-SRV-INST, **Server Principal Names:** MSSQLSvc, yuzu.fruit.i.extrahop.com:1433, **Processing Time:** 6.234, **Realm Of Server:** FRUIT.I.EXTRAHOP.COM, **Response Bytes:** 128, **Response L2 Bytes:** 242, **Response Packets:** 2, **Response RTO:** 0, **reqBytes:** 0, **reqL2Bytes:** 0, **reqPkts:** 0, **reqRTO:** 0, **ex:** {"isSuspicious:b":false}

DNS Application-Layer (Layer 7) Details: Record Types, Response Codes, Host Queries

Example of why it matters: DNS is frequently misused by attackers for reconnaissance (e.g. scanning) and command and control communications (e.g. tunneling). In these cases, it is important to have visibility into DNS host queries and DNS record types.

Time: 2018-11-29 03:24:54.701, **Record Type:** DNS Response, **Flow:** 0308f4f05bffcc86, **Client:** web2.seadmz.example.com, **Client IPv4 Address:** 172.21.1.81, **Client Port:** 49516, **Server:** VMware 172.21.1.3, **Server IPv4 Address:** 172.21.1.3, **Server Port:** 53, **Client Zero Windows:** 0, **Server Zero Windows:** 0, **Answers:** {"name:s":"81.1.21.172.in-addr.arpa","ttl:n":1,"type:s":"PTR","data:s":"web2.seadmz.example.com"}, **Opcode:** QUERY, **Authoritative:** True, **Response Truncated:** False, **Query Name:** 81.1.21.172.in-addr.arpa, **Query Type:** PTR, **Processing Time:** 0.099, **Response Bytes:** 79, **Response L2 Bytes:** 121, **Response Packets:** 1, **IP Protocol:** UDP, **Recursion Available:** True, **ex:** {"isSuspicious:b":false}, **isAuthenticData:** false, **answers:** [{"name:s":"81.1.21.172.in-addr.arpa","ttl:n":1,"type:s":"PTR","data:s":"web2.seadmz.example.com"}]

LDAP Application-Layer (Layer 7) Details: Errors, Requests, Responses, Plain Text Messages, SASL Messages, Bind Distinguished Names, Search Query

Example of why it matters: LDAP errors and account names are important features when detecting malicious authentication activity such as brute-force attempts.

Time: 2018-11-28 22:47:02.921, **Record Type:** LDAP Response, **Flow:** f6fafe705bff8b66, **Client:** VMware 54D6F7, **Client IPv4 Address:** 172.29.1.101, **Client Port:** 39760, **Server:** ldap1-nyc, **Server IPv4 Address:** 172.22.2.38, **Server Port:** 389, **Client Zero Windows:** 0, **Server Zero Windows:** 0, **Method:** BindResponse, **Error:** invalidCredentials, **Error Detail:** invalidCredentials, **Message Size:** 14, **Processing Time:** 0.242, **Response Bytes:** 14, **Response L2 Bytes:** 146, **Response Packets:** 2, **Response RTO:** 0, **reqBytes:** 0, **reqL2Bytes:** 84, **reqPkts:** 1, **reqRTO:** 0, **ex:** {"isSuspicious:b":false}

SMB/CIFS Application-Layer (Layer 7) Details: Errors, File Paths, Username, Requests, Responses, Reads, Writes, Warnings

Example of why it matters: CIFS write activity and file names are extremely relevant features when detecting ransomware, and logon error messages are important when detecting brute-force attacks.

Time: 2018-11-29 00:18:16.002, **Record Type:** CIFS, **Flow:** fda323185bffa0c7, **Client:** AccountingLaptop, **Client IPv4 Address:** 192.168.35.22, **Client Port:** 40084, **Server:** 10.10.10.3, **Server IPv4 Address:** 192.168.6.179, **Server Port:** 445, **Client Zero Windows:** 0, **Server Zero Windows:** 0, **Method:** SMB2_SESSION_SETUP, **FileInfo Command:** False, **Lock Command:** False, **Read Command:** False, **Write Command:** False, **Create Command:** False, **Delete Command:** False, **Rename Command:** False, **Error:** STATUS_LOGON_FAILURE, **Status Code:** 3221225581, **User:** \\Pre-Login, **Request Bytes:** 572, **Request L2 Bytes:** 638, **Request Packets:** 1, **Request RTO:** 0, **Request Size:** 568, **Response Bytes:** 77, **Response L2 Bytes:** 143, **Response Packets:** 1, **Response RTO:** 0, **Response Size:** 73, **Processing Time:** 0.766, **Request Transfer Time:** 0, **Response Transfer Time:** 0, **ex:** {"isSuspicious:b":false}

SSH Session-Layer (Layer 5) Details: Record Type, Version, Cipher Algorithm, MAC Algorithm, Compression Algorithm, KEX Algorithm, Client Implementation, Server Implementation

Example of why it matters: SSH algorithm details are important for accurate detections of suspicious tunnelling activity.

Time: 2018-11-28 23:49:04.051, **Record Type:** SSH Open, **Flow:** fc4065705bff99ef, **Client:** Device 192.168.0.101,
Client IPv4 Address: 192.168.0.101, **Client Port:** 43702, **Server:** Device 192.168.0.103, **Server IPv4 Address:** 192.168.0.103,
Server Port: 22, **Client Version:** 2.0, **Client Cipher Algorithm:** chacha20-poly1305@openssh.com,
Client Mac Algorithm: umac-64-etm@openssh.com, **Client Compression Algorithm:** none, **Server Version:** 2.0,
Server Cipher Algorithm: chacha20-poly1305@openssh.com, **Server Mac Algorithm:** umac-64-etm@openssh.com,
Server Compression Algorithm: none, **Request Bytes:** 1,425, **Request L2 Bytes:** 2,076, **Request Packets:** 9, **Response Bytes:** 1,017,
Response L2 Bytes: 1,495, **Response Packets:** 7, **Request RTO:** 0, **Response RTO:** 0, **Round Trip Time:** 0.315,
KEX Algorithm: curve25519-sha256@libssh.org, **Client Zero Windows:** 0, **Server Zero Windows:** 0,
Client Implementation: OpenSSH_7.2p2, **Server Implementation:** OpenSSH_7.2p2, **clientBytes:** 1425, **clientL2Bytes:** 2076, **clientPkts:** 9,
serverBytes: 1017, **serverL2Bytes:** 1495, **serverPkts:** 7, **clientRTO:** 0, **serverRTO:** 0, **ex:** {"isSuspicious:b":false}

SSL/TLS Session-Layer (Layer 5) Details: Versions, Alerts, Content Type, JA3 Hash, Certificate Subject, Certificate Expiration Dates, Domains (SNI), Cipher Suites, Record Sizes

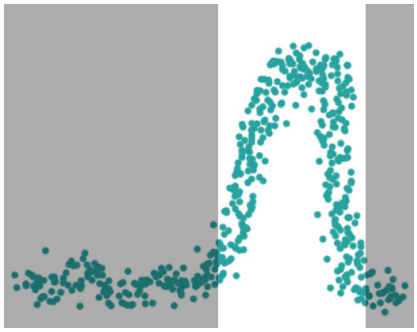
Example of why it matters: SSL/TLS certificate issuer details are extremely relevant details for accurately detecting malicious encrypted traffic.

Time: 2018-11-29 03:24:59.775, **Record Type:** SSL Open, **Flow:** 030981d85bffcc8a, **Client:** VMware 4B139C,
Client IPv4 Address: 179.180.128.35, **Client Port:** 55581, **Server:** VMware B84943, **Server IPv4 Address:** 172.22.1.90, **Server Port:** 443,
Client Zero Windows: 0, **Server Zero Windows:** 0, **Certificate Fingerprint:** D3C4B692662144B66A0C2A95979162D47288230F,
Certificate Key Size: 2048, **Certificate Subject:** orbital.example.com, **Certificate Not After:** 2024-10-21,
Certificate Signature Algorithm: sha1WithRSAEncryption, **Cipher Suite:** TLS_RSA_WITH_RC4_128_MD5, **Handshake Time:** 910.606,
Version: TLSv1.0, **Compressed:** False, **Request Bytes:** 0, **Request RTO:** 0, **Request Packets:** 0, **Request L2 Bytes:** 0, **Response RTO:** 0,
Response Packets: 2, **Response L2 Bytes:** 175, **Response Bytes:** 43, **Certificate Issuer:** orbital.example.com, **Weak Cipher Suite:** True,
Renegotiate: False, **Client Certificate Requested:** False, **Certificate Not Before:** 2014-10-24, **Certificate Self Signed:** True,
JA3 Hash: a1a58caae3b702e2be7609dc2feca70d, **clientBytes:** 0, **clientL2Bytes:** 0, **clientPkts:** 0, **serverBytes:** 43, **serverL2Bytes:** 175,
serverPkts: 2, **clientRTO:** 0, **serverRTO:** 0, **ex:** {"isSuspicious:b":false}

Machine Learning Models

Reveal(x) machine learning is unsupervised, self-adapting, and accurate. Since 2014, our data scientists and cybersecurity experts have been refining our machine learning capabilities to handle enterprise demands and eliminate false positives. To build and train machine learning models,the data science team uses real-world activity from hundreds of ExtraHop enterprise deployments encompassing millions of devices. With this unmatched dataset, the data science team can continuously enhance the accuracy of Reveal(x) detections based on what actually happens in production environments across the entire ExtraHop customer base in addition to users' direct feedback. Some examples of the types of machine learning models used by Reveal(x) are listed below.

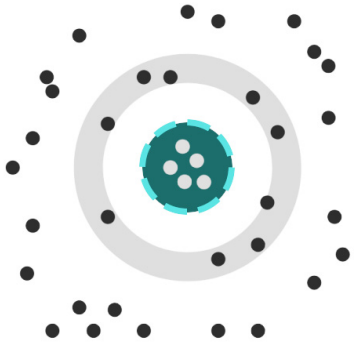
Predictive Models



Reveal(x) uses sophisticated algorithms to understand how each individual device, subnet, and application is expected to behave. In effect, Reveal(x) is using an organization's own environment, combined with Extrahop's domain expertise in cybersecurity, to build tens of thousands of predictive models where significant deviations from the prediction are detected as anomalies or potential threats. This method of predictive modeling results in fewer false positives than legacy baselining, dynamic thresholds, and rules-based approaches which are statistical in nature and may not accurately reflect what should be alerted on in real-world environments. In

addition, the predictive models self-adapt, meaning that organizations do not need to periodically tune the detectors as they may need to do with dynamic thresholds and rules-based systems.

Importance and Privilege Inference Models



Reveal(x) uses novel graph-based algorithms to analyze all the interactions that it observes between different users and devices on the network. It builds multiple inference models to automatically determine which devices are important and which users are high-privileged. These models are unique to each environment and train themselves by continuously analyzing authentication and data access behavior across devices and users. They enable Reveal(x) to detect subtle but suspicious changes in user and device behaviors. For example, these models power the detection of network privilege escalation, a key step in an attack campaign, and helps to detect insider threat activity where the attacker is already inside the network but is interacting with high-importance data servers from a low-privileged account.

Peer Group Models

Reveal(x) is capable of identifying sudden behavioral changes in a device on the network by comparing its behaviors with other similar devices. It first clusters devices into peer groups, by monitoring how each device behaves on the network and grouping the ones that are behave very similarly using proprietary clustering algorithms. Examples of peer groups include: web or database servers behind a load balancer, developer workstations in a team, VoIP phones, and printers. Reveal(X) then builds behavior models for every peer group and applies unsupervised anomaly detection algorithms to accurately identify when an attacker has compromised a device and causes it to start acting differently than its peers.

Additional Threat Visibility

Unlike other NTA products that extract just a few hundred metadata elements from network traffic, Reveal(x) records 4,600+ different metrics across 50+ protocols. This is a data-first approach in which all network behavior is recorded regardless of whether it is flagged as malicious. With comprehensive visibility, users can detect activity that represents risk to their particular organization, such as the use of insecure protocols or non-compliant communications to a file server or database containing sensitive information.

In addition, Reveal(X) enables users to retrospectively investigate and perform forensics analysis post-facto for attacks and behaviors that were not initially detected as an attack. For example, because Reveal(x) continuously records metadata about all network transactions, organizations can use Reveal(x) to see if they had been impacted by a vulnerability before it was publicly announced. For example, many ExtraHop customers examined SSL heartbeat messages sent to their servers for the months prior to the announcement of the Heartbleed vulnerability in 2014.

Threat Intelligence Integration

Reveal(x) is able to ingest third-party threat intelligence data from a wide variety of sources by supporting the industry-standard Structured Threat Information Expressions (STIX) format. While users can curate their own threat feeds, ExtraHop maintains a list of recommended free feeds which can be added to each new Reveal(x) deployment, including Anomali Limo, AlienVault OTX, and the SANS Top 100 Suspicious IP Addresses feed. Reveal(x) integrates threat intelligence throughout detection and

investigation workflows. Threat data, such as suspect URIs, hosts, or IP addresses, is presented alongside detected anomalies within the UI, providing rich context for triage and hunting.

Customizing Detections and Integration

Rapid programmability, easy extensibility, open integration, and community: These are the characteristics of a true platform with which security teams can build a detection solution that fits their organization. With Reveal(x), SecOps teams can quickly meet new requirements, customize the solution, and stream events to other platforms.

Application Inspection Triggers

Reveal(x) offers a programmable interface for the real-time stream processor, enabling you to define new types of metadata that you want to extract from network communications. With rapid programmability, your organization does not need to wait for a vendor to add new network analysis capabilities. Instead, your organization and the community can quickly respond to new threats. For example, within 24 hours of the EternalBlue vulnerability announcement in 2017, the ExtraHop community developed a trigger to track exploit attempts over SMB.

Automation and Orchestration

Reveal(x) offers a programmable interface for the real-time stream processor, enabling users to define new types of metadata to extract from network communications. With rapid programmability, organizations do not need to wait for a vendor to add new network analysis capabilities. For example, within 24 hours of the EternalBlue vulnerability announcement in 2017, the ExtraHop community developed a trigger to track exploit attempts over SMB.

NTA products should not be technology islands, but work with existing security infrastructure. Detection alone is not enough—network traffic analysis solutions should enable investigation and response. To that end, Reveal(x) offers industry-leading investigative workflow and network forensics capabilities along with Open Data Stream functionality and a REST API that enables security teams to build powerful automation and orchestration, kicking off workflows and processes based on activity observed in real time. For example, users can automatically:

- Generate a ticket in a SIEM or other system
- Quarantine infected host machines using network access control (NAC)
- Block malicious IPs using a firewall
- Update CMDBs with newly discovered devices
- Send regular reports to individuals and teams
- Pull Reveal(x) data into the web UIs of other management tools

To provide a concrete example, the following lines of JavaScript code in the programmable interface for Reveal(x) will immediately instruct a NAC platform to quarantine a device detected with ransomware:

```
var my_path = "/utilities/?quarantine" + "&clientmacaddress=" + Flow.client.device.hwaddr +
"&source="ExtraHop" + "&reason="RANSOMWARE";

Remote.HTTP("my_NAC").post( {path: my_path} );
```

In the code snippet above, the user defines a URL path in which to make an outbound REST call. The variable `my_NAC` refers to a predefined Open Data Stream endpoint, which you would configured through the Reveal(x) Admin GUI. Once the NAC platform receives the REST call, it puts the infected workstation into a quarantined state (such that it can no longer encrypt network files).

Examples of Detections in Reveal(x)

Attackers can modify the tools they use, but it's hard to avoid specific techniques required to achieve their goals. Reveal(x) uses machine learning extensively to detect malicious behavior throughout the attack lifecycle, unlike other network traffic analysis products that rely heavily on rules and dynamic thresholds. Reveal(x) places these behavioral detections into categories: Command & Control, Reconnaissance, Exploitation, Lateral Movement, and Actions on Objectives.

Most organizations have a number of security tools to defend the perimeter of the network, but the reality is that these vital perimeter defenses are regularly breached. Reveal(x) provides a vital additional layer of defense in the case of an intrusion by detecting activity throughout the lifecycle of an attack, including subtle post-compromise activity such as misuse of Windows remote procedure calls and abnormal behavior from low-privileged devices or users. The examples below demonstrate the breadth of coverage provided by Reveal(x). **Please note:** These are example categories of detections and not a comprehensive list.

Command & Control

Unless the attacker has physical access to the devices on the network, they must control the systems remotely over the network. This command and control activity can take place over control protocols such as RDP, SSH, or telnet; over a custom protocol; or it may be disguised within the misuse of another protocol, such as DNS.

In addition to matching traffic with malicious IP addresses and domains contained in threat intelligence feeds, Reveal(x) uses behavioral machine learning models to recognize traffic patterns associated with common command and control activities.

DNS tunneling

A command and control server is set up as an authoritative name server for an attacker-controlled domain. A client begins exchanging suspicious data with a server over the DNS protocol, specifically encoding command and control messages or data payloads in DNS queries and responses. Because it can be hard to detect and is often unrestricted, DNS tunnelling has become very popular among attackers both as a method of command and control as well as for data exfiltration.

Reveal(x) analyzes the host queried—an application-layer detail contained in the transaction payload—to detect DNS tunnelling. Many organizations with well-segmented networks keep port 53 open to DNS so that internal DNS servers can proxy DNS queries of external domains, making the ability to detect anomalous behavior more important. Additionally, analyzing the logs of the local DNS servers is not sufficient if clients can connect to public DNS servers.

ICMP tunneling

An attacker can encapsulate data inside ICMP requests and replies, obfuscating their communications from firewalls and network monitoring tools. In addition to command and control communications, attackers use ICMP tunneling for data exfiltration. This group of detections can be applied to both categories of the attack lifecycle.

Reverse shell to external IPs

An external device uses reverse shell to remotely control an internal device. This technique is often used to evade firewalls for remote access. For example, a malware-infected system opens an SSH connection to an external device, which is allowed by the

firewall because it's outbound. Once the connection is open, the attacker opens a reverse SSH shell back into the infected device, tunneling through the original SSH connection.

Suspect IPs, domains, and URIs

Threat intelligence feed data matches observed IPs, domains, and URIs, indicating suspicious communications.

Unusual inbound connections

An endpoint accepts a new connection from a external device using a control protocol such as RDP, SSH, telnet, or VNC. Because Reveal(x) parses the protocol natively and does not rely on port matching, it can detect unusual inbound connections for control protocols even if they are not used on a standard port. Note: Reveal(x) will flag this behavior the first time it occurs even if it is a legitimate inbound connection, but will subsequently learn that this behavior is normal.

Reconnaissance

Before they ever send a phishing email or attempt a web exploit, sophisticated attackers will gather information about their target. But this reconnaissance activity does not stop once they gain access to a system. Once inside a network, attackers continue to reconnoiter the environment to orient themselves and determine what steps they need to take next to achieve their overall objective. Attackers often perform scans of the network to discover devices, services, file servers, and directories. This activity can look similar to legitimate vulnerability scanning activity.

Reveal(x) detects a wide variety of scanning activity, but applies heuristics to identify known vulnerability scanners so that analysts can more easily determine if the scan is approved or not.

Active Directory reconnaissance

Most companies use Active Directory to manage users and Windows devices on their network. Attackers often query Active Directory for detailed information about users and devices. This is a commonly used technique to stealthily map out the network and assets without triggering a lot of traffic. To detect this behavior, Reveal(X) performs in-depth application-layer analysis of various Active Directory protocols, including MSRPC, Kerberos, LDAP, and CLDAP.

Address and port scans

A device sends various probes in an attempt to determine what services are available or other information about a host. An attacker might be looking for services, such as DNS, SNMP, or DHCP, that are listening on open ports. This group of detections identifies activity over ICMP, TCP, and UDP, including several variants of TCP stealth scans.

DNS reverse lookups

A device attempts an excessive number of reverse DNS lookups indicating that the client may be compromised. DNS reverse lookups are legitimate activity to associate a domain name with an IP address, but attackers can also use them to reconnoiter the environment. To detect this activity, Reveal(x) analyzes the DNS request messages, an application-layer detail. Importantly, Reveal(x) employs machine learning to provide superior detection accuracy, learning normal behavior and only flagging anomalous DNS reverse lookup enumerations.

Web scans

A device probes a web directory or otherwise attempts to scan and enumerate web site content. This requires analysis of the HTTP messages and errors contained in the transaction payload, both application-layer details.

VOIP scans

A device scans a phone system, indicating an attacker may be attempting to enumerate their services and find an attack vector. Phone systems can be attractive targets as they are often running outdated firmware and may not be generating log information. To identify VOIP scans, Reveal(x) analyzes VOIP protocols including SIP and RTP, both of which are application-layer protocols.

Exploit

Once they have identified their targets inside the network, attackers use tools to gain access to sensitive systems by exploiting vulnerabilities. Detection at this stage of the attack lifecycle is crucial because at this point, attackers still have not accessed sensitive systems. If defenders can detect and respond to attack activity at this stage, they can avoid financial and brand damage.

Brute force attacks

A client sends an excessive number of requests and checks the resulting return status, indicating that an attacker is attempting to find valid login credentials. This group of detections analyzes protocols used in authentication, such as telnet, database, HTTP, FTP, Kerberos, LDAP, RDP, RFB, SSH, VNC, WordPress (over HTTP). This requires analysis of the error messages contained in the transaction payload, an application-layer detail. For example: HTTP 401 returns, Kerberos unknown SPN errors, and LDAP bind requests.

DNS rebinding attacks

A malicious website modifies its own DNS IP addresses to attempt to hijack a client's browser to act as a relay, allowing them to bypass the victim's firewall and communicate directly with devices on the internal network. DNS rebinding can be used to attack Internet-connected devices including networking access points and routers from Cisco, Netgear, Extreme, Aruba, and Avaya; IP cameras from Avaya, Cisco, Dell, NEC, and Polycom; and printers from Hewlett Packard, Epson, Konica, Lexmark, and Xerox. This group of detections requires analysis of the DNS responses, an application-layer detail.

IDS evasion

An attacker can hide their payload from detection by an intrusion detection system (IDS) by specially crafting packets in such a way that the host correctly interprets the payload but the IDS does not. For example, a device can send or receive an unusual number of overlapping IP fragments, which can create challenges for IP fragment reassembly. This group of detections can also be applied at other stages of the attack lifecycle.

Kerberos attacks

A device sends Kerberos requests asking for a service principal name (SPN) that does not exist, indicating that an attacker may be scanning for usernames. Or, a device generates a Kerberos duplicate ticket error, indicating that an attacker has stolen a ticket and is replaying it. To detect this group of attacks, Reveal(x) analyzes the Kerberos error messages and tracks uses and issuances of Kerberos tickets, which are application-layer details.

LLMNR and NetBIOS poisoning

A device sends Link-Local Multicast Resolution (LLMNR) or NetBIOS Name Service (NBT-NS) responses to one or more clients in an attempt to impersonate another device. The device may attempt to collect credentials or other information from clients that experienced failed DNS queries and use LLMNR or NBT-NS for name resolution instead.

SQL injection attacks

A device sends a HTTP request with malformed content to a web application to modify the database request. This indicates that an attacker is attempting to obtain extra information from the database. In the case of in-band SQL injection, the attacker relies on trial-and-error to modify the SQL request sent by the application to the database until the attack is successful. This group of detections requires analysis of the HTTP transaction payload, an application-layer detail. This may also be detected by inspecting the database queries themselves and looking for anomalous requests.

WPAD attacks

A device responds to clients' Windows Proxy Auto-Discovery (WPAD) requests to establish a man-in-the-middle session, where they can either harvest user credentials or inject code into a browser. WPAD should be disallowed or monitored closely to detect abuse. This group of detections flags unapproved WPAD behavior over HTTP, DNS, and DHCP. To detect this behavior, Reveal(x) analyzes the HTTP, DNS, and DHCP requests and responses, all of which are application-layer details.

Lateral Movement

Attackers are opportunistic and enter the network however they can. From there, the attacker needs to move laterally to gain remote access to their target systems. The Lateral Movement category of detections focuses on activities and behaviors that allow the attack to expand their presence from one system to another, escalate privileges, and collect data. Insider threats may also begin their mission at this stage of the attack lifecycle because they already know their targets but lack the privileges needed to modify or steal data.

Reveal(x) applies machine learning to detect exploits to gain access to remote systems within the network, privilege escalation, and unusual data movement as the attacker collects data prior to exfiltration.

Abnormal internal data transfer

Detects when a device accesses abnormally large volume of data over file transfer protocols (e.g. FTP, NFS, CIFS, database) or remote control protocols (e.g. SSH, RDP, VNC), indicating that an attacker is collecting data and staging it for exfiltration. Because it does not rely on port matching but parses the protocol natively, Reveal(x) can detect abnormal internal data transfer even if protocols are running over non-standard ports.

Abnormal network activity (peer group)

Anomalous network activity relative to a particular device's behavioral peer group. For example, one of your VOIP phones has started using control protocols to interact with other devices on the network. Or a printer suddenly communicates with an external service that no other printer in the network has talked to before.This group of detections is not exclusive to the Lateral Movement category.

Abuse of machine accounts

A device attempts to log in with machine accounts in Active Directory, such as `MACHINENAME$`, using expired or invalid passwords, or receives a policy error. Both of these activities may indicate an attacker is attempting to elevate their privileges. To detect this group of behaviors, Reveal(x) analyzes LDAP and Kerberos login and error messages, all of which are application-layer details contained within the transaction payload.

Network privilege escalation

A low-privileged device accesses data on high-importance servers, when that device historically only has access to less important servers. For example, a malicious insider attempts to access sensitive assets that they normally would not. Reveal(x) detects privilege escalation by inferring the importance and privilege for all devices on the network using graph-based machine learning. These models are continuously updated based on observed authentication and data access behavior, and can detect insider threats or other malicious devices accessing high-importance assets that are historically above their privilege.

PsExec remote command tool

A device runs the `psexesvc` service over SMB on a remote system. PsExec is a Windows utility that allows administrators to run remote commands through RPC tunneled inside SMB, but can also be used by attackers to run commands and launch processes to compromise a Windows system. Reveal(x) detects these PsExec attacks as well as stealthy variants such as SMBexec.

Windows RPC misuse

A device begins using Windows remote procedure calls in an abnormal manner, indicating that an attacker is misusing legitimate Windows services to compromise a target machine. The most sophisticated attackers employ these techniques because they do not require scanning or brute-force methods, which are more noisy and relatively easier to identify. In addition, these techniques do not require running new executables, so they are very stealthy and can often evade endpoint monitoring solutions. This group of detections analyzes behavior to recognize the suspicious use of Windows/Active Directory APIs and tools, including CIFS, MSRPC, WinRM, and WMI on the network.

Actions on Objectives

Attackers may have various objectives, including sabotage, espionage, fraud, theft of intellectual property, theft of personal information, holding files for ransom, or simply making money by installing malware such as cryptomining or botnets. Reveal(x) uses a number of methods to detect actions on objectives, including ransomware.

BitTorrent activity

A device transfers data through the BitTorrent protocol, which has no built-in security controls, exposing the organization to illegal content or malware within files that were downloaded by BitTorrent devices on the network.

Cryptomining activity

A device sends data over protocols associated with cryptomining.

Data exfiltration to external IPs

A device sends an unusually large amount of data to external IP addresses, indicating potential data exfiltration.

Data smuggling

A device downloads an unusually large amount of data from highly valued internal servers, indicating potential data exfiltration.

Ransomware activity

Detects suspicious file system activity as well as artifacts associated with known ransomware activity, including the EternalBlue exploit which was prominent in the WannaCry ransomware outbreak but also leveraged by other forms of malware. This group of detections requires analysis of the file names and methods used, application-layer details that are contained in the transaction payload.

ExtraHop Networks, Inc.
520 Pike Street, Suite 1600
Seattle, WA 98101 USA

http://www.extrahop.com/
info@extrahop.com
T 877-333-9872
F 206-274-6393