NXLog

NXLog produces log collection software including the **NXLog Community Edition**.

Gitlab: https://gitlab.com/nxlog-public/nxlog-ce

Download at https://nxlog.co/products/nxlog-community-edition/download

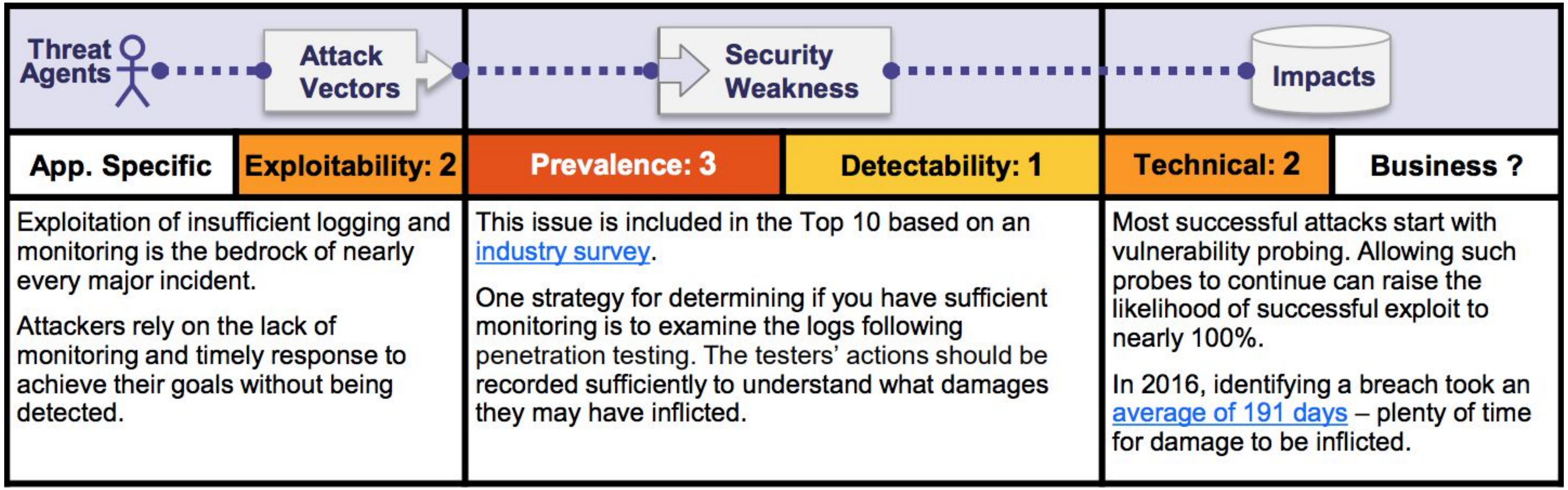
NXLog Community Edition on Docker - https://hub.docker.com/r/nxlog/nxlog-ce/

Documentation - https://nxlog.co/documentation

Why log and why centralized logs

- Event data accessible even if originating server is offline, compromised, decommissioned.
- Data can be analyzed and correlated across more than one system.
- More difficult for malicious actors to remove evidence from logs that have already been forwarded.
- Incident investigation and auditing is easier, as all event data is collected in one location.
- Scalable, high-availability, and redundancy solutions are easier to implement and maintain because they can be implemented at the point of the collection server.
- Compliance with internal and external standards for log data retention can be managed at a single point.

A10 :2017 Insufficient Logging & Monitoring



Threat Agents	Attack Vectors	Security Weakness		Impacts	
App. Specific	Exploitability: 2	Prevalence: 3	Detectability: 1	Technical: 2	Business ?
Exploitation of insufficient logging and monitoring is the bedrock of nearly every major incident. Attackers rely on the lack of monitoring and timely response to achieve their goals without being detected.		This issue is included in the Top 10 based on an industry survey. One strategy for determining if you have sufficient monitoring is to examine the logs following penetration testing. The testers' actions should be recorded sufficiently to understand what damages they may have inflicted.		Most successful attacks start with vulnerability probing. Allowing such probes to continue can raise the likelihood of successful exploit to nearly 100%. In 2016, identifying a breach took an average of 191 days – plenty of time for damage to be inflicted.	

➔ Log messages are not clear (and to add, logs messages unstructured)

➔ Alerts are not generated in a way that initiates a response from the security team

Unstructured example

```
<38>Jan 22 10:30:12 myhost sshd[8459]: Failed password for invalid
user linda from 192.168.1.60 port 38176 ssh2
```

The log can still contain a pre-defined structure which is the metadata before the colon (:).

The rest of the field (`Failed password for invalid user linda from 192.168.1.60 port 38176 ssh2`) is unstructured.

Structured example of a few key value pairs from the metadata

```
$AuthMethod=password
$AccountName=linda
$SourceIPAddress=192.168.1.60
```

Unstructured DNS server log 1 example

```
12-Jan-2019 06:38:30.142 info: client 192.168.100.105#58985
(_http._tcp.security.ubuntu.com): query: _http._tcp.security.ubuntu.com IN SRV +
(192.168.100.105)
```

Structured example in JSON of DNS Server log 2

```
{
  "EventReceivedTime": "2019-01-12 07:55:34",
  "SourceModuleName": "dns_queries",
  "SourceModuleType": "im_file",
  "Date": "12-Jan-2019",
  "QName": "example.com",
  "QType": "A",
  "RFlags": "+E",
  "RemoteIP": "127.0.0.1",
  "Severity": "info",
  "Time": "07:17:09.816",
  "EventTime": "2019-01-12 07:17:09"
}
```

Configuration is modular, text-based, no GUI.

- **[1]** Accept data from different sources

  ie MongoDB, Snort, Nginx logs, Bro Network Security Monitor, Windows sources etc.

- **[2]** Convert the data internally, leverage at the source

  i.e. Convert Windows EventLog to Snare agent format, BSD Syslog, Syslog encapsulated JSON

- **[3]** Output to destinations - local file-system, or external

[1] Snort Rule

```
alert tcp any any -> any any
(msg:"Exploit detected"; sid:1000001;
content:"exploit";)
```

[3a] NXLog CE config excerpt

```
<Extension snort>
    Module      xm_multiline
    HeaderLine  /^\[\*\*\] \[\S+] (.*) \[\*\*\]/
    Exec        if $raw_event =~ /^\s+$/ drop();
</Extension>

<Input in>
    Module      im_file
    File        "/var/log/snort/alert"
    InputType   snort
  <Exec>
        <<add rules here for the key value pairs>>
  </Exec>
</Input>
```

[2] Example Log from Rule

```
[**] [1:1000001:0] Exploit
detected [**]
[Priority: 0]
04/30-07:54:38.312536
172.25.212.204:80 ->
192.168.255.110:46127
TCP TTL:64 TOS:0x0 ID:19844
IpLen:20 DgmLen:505 DF
***AP*** Seq: 0xF936BE12  Ack:
0x2C9A47D8  Win: 0x7B  TcpLen:
20
```

[3b] NXLog CE config excerpt (continue)

```
Exec rules:
        if $raw_event =~ /(?x)^\[\*\*\]\ \[\S+\]\ (.*)\ \[\*\*\]\s+
                          (?:\[Classification:\ ([^\]]+)\]\ )?
                          \[Priority:\ (\d+)\]\s+
                          (\d\d).(\d\d)\-(\d\d:\d\d:\d\d\.\d+)
                          \ (\d+.\d+.\d+.\d+):?(\d+)?\ ->
                          \ (\d+.\d+.\d+.\d+):?(\d+)?\s+\ /
        {
            $EventName = $1;
            $Classification = $2;
            $Priority = $3;
            $EventTime = parsedate(year(now()) + "-" + $4 + "-" + $5 + " " + $6);
            $SourceIPAddress = $7;
            $SourcePort = $8;
            $DestinationIPAddress = $9;
            $DestinationPort = $10;
        }
```

[4] Structured data example - JSON excerpt only

```
{

"EventName": "Advanced exploit detected",
"Classification": "Executable Code was Detected",
"Priority": "100",
"SourceIPAddress": "192.168.255.110",
"DestinationIPAddress": "172.25.212.204"

<<and more key value pairs….>>

}
```

Let’s do Powershell logging...

```
<Extension _json>
    Module  xm_json
</Extension>

<Extension _syslog>
    Module  xm_syslog
</Extension>

<Input module_logging>
    Module  im_msvistalog
    <QueryXML>
        <QueryList>
            <Query Id="0"
Path="Microsoft-Windows-PowerShell/Operational">
                <Select
Path="Microsoft-Windows-PowerShell/Operational">
                    *[System[EventID=4103]]</Select>
            </Query>
        </QueryList>
    </QueryXML>
    Exec    $Message = to_json(); to_syslog_bsd();
</Input>
```

```
<Output out>
    Module      om_http
    URL         http://localhost:9200
    ContentType application/json
    <Exec>

set_http_request_path(strftime($Event
Time, "/nxlog-%Y%m%d/" +

$SourceModuleName));
        rename_field("timestamp",
"@timestamp");
        to_json();
    </Exec>
</Output>
```

About NXLog Community Edition

- Log collection software
- Can be integrated with SIEM (Security Information and Event Management) suites as a log collector including Rapid7, IBM QRadar, RSA NetWitness
- Can send logs directly to Elasticsearch or to Logstash, and Kibana
- Use as a collector for Graylog (open source log management tool with a GUI that uses Elasticsearch as a backend)
- And so on.. (see the User Guide)

Docker Hub https://hub.docker.com/r/nxlog/nxlog-ce/
Gitlab https://gitlab.com/nxlog-public/nxlog-ce
Twitter @nxlog_team

NXLog Community Edition reference manual
https://nxlog.co/docs/nxlog-ce/nxlog-reference-manual.html

Main User Guide
https://nxlog.co/documentation/nxlog-user-guide/nxlog-user-guide

NXLog Community Edition download
https://nxlog.co/products/nxlog-community-edition/download

See main User Guide for samples
https://nxlog.co/documentation/nxlog-user-guide/nxlog-user-guide