Reflecting on
Twenty Years of Bro

Vern Paxson

International Computer Science Institute
Lawrence Berkeley National Laboratory
EECS Department, University of California, Berkeley
Broala, LLC

vern@berkeley.edu

August 4, 2015
Part I: Origin & Technical Evolution
The Bro Network Security Monitor

Why Choose Bro? Bro is a powerful network analysis framework that is much different from the typical IDS you may know.

Adaptable
Bro's domain-specific scripting language enables site-specific monitoring policies.

Efficient
Bro targets high-performance networks and is used operationally at a variety of large sites.

Flexible
Bro is not restricted to any particular detection approach and does not rely on traditional signatures.

In-depth Analysis
Bro comes with analyzers for many protocols, enabling high-level semantic analysis at the application layer.

Highly Stateful
Bro keeps extensive application-layer state about the network it monitors.

Open Interfaces
Bro interfaces with other applications for real-time exchange of information.
Bro: A System for Detecting Network Intruders in Real-Time

Vern Paxson
Network Research Group
Lawrence Berkeley National Laboratory*
Berkeley, CA 94720
vern@ee.lbl.gov

USENIX Technical Program - 7th USENIX Security Symposium, 1998
Prior to developing Bro, we had significant operational experience with a simpler system based on off-line analysis of tcpdump [JLM89] trace files. Out of this experience we formulated a number of design goals and requirements:
Growth Trends in Wide-Area TCP Connections

Vern Paxson
Lawrence Berkeley Laboratory and
EECS Division, University of California, Berkeley
1 Cyclotron Road
Berkeley, CA 94720
vern@ee.lbl.gov

Revised May 11, 1994
Most popular protocols grew 50-70%/year; Some of this fueled by abuse
HTTP grew 300x / year!
TCPDUMP(1)

NAME
tcpdump - dump traffic on a network

SYNOPSIS
tcpdump [ -AbdDefhHIJKlLmNOpPqRstuuvxX ] [ -B buffer_size ] [ -c count ]
[ -C file_size ] [ -G rotate_seconds ] [ -F file ]
[ -i interface ] [ -j timestamp_type ] [ -k (metadata_arg) ]
[ -m module ] [ -M secret ]

PCAP(3PCAP)

NAME
cap - Packet Capture library

SYNOPSIS
#include <pcap pcap.h>

DESCRIPTION
The Packet Capture library provides a high level interface to packet capture systems. All packets on the network, even those destined for
Prior to developing Bro, we had significant operational experience with a simpler system based on off-line analysis of tcpdump [JLM89] trace files. Out of this experience we formulated a number of design goals and requirements:

High-speed, large volume monitoring

No packet filter drops

Real-time notification

Mechanism separate from policy

Extensible

Avoid simple mistakes

The monitor will be attacked
Original Architecture

- Taps network link passively, sends up a copy of all network traffic.
Original Architecture

- Kernel filters down high-volume stream via standard *libpcap* packet capture library.
“Event engine” decodes protocols, distills filtered stream into high-level, *policy-neutral* events reflecting underlying network activity

- E.g., connection_attempt, http_reply, teredo_authentication
- These span a range of semantic levels
- Currently 400+ different types
Original Architecture

- Script written in Domain Specific Language processes event stream, incorporates:
  - Context/state from past events
  - Additional input sources
  - Site’s particular policies

... and *takes action*:
- Records to disk - *extensive* logs
- Generates real-time alerts
- *Executes programs* as a form of response
Architecture As It Has Evolved

Scalable high performance via Bro Cluster
Architecture As It Has Evolved

No filtering by default; analyze off-port traffic using Dynamic Protocol Detection
Architecture As It Has Evolved

Analysis of events from other sources; parsing of non-network formats (items/files)
Architecture As It Has Evolved

Extensive library functionality, input/logging/output & analysis frameworks

Packet Stream

Network

libpcap

Policy Script Interpreter

Real-time Action Log Archive

Event Engine

Event Control

Tcpdump Filter

Filtered Packet Stream

Packet Stream

Real-time Action Log Archive

Policy Script Interpreter
Prior to developing Bro, we had significant operational experience with a simpler system based on off-line analysis of tcpdump [JLM89] trace files. Out of this experience we formulated a number of design goals and requirements:

**High-speed, large volume monitoring**

**High-speed, large volume monitoring** For our environment, we view the greatest source of threats as external hosts connecting to our hosts over the Internet. Since the network we want to protect has a single link connecting it to the remainder of the Internet (a “DMZ”), we can economically monitor our greatest potential source of attacks by passively watching the DMZ link.

Key enabler: donation of DEC Alphas (kudos Jeff Mogul)
Prior to developing Bro, we had significant operational experience with a simpler system based on off-line analysis of tcpdump [JLM89] trace files. Out of this experience we formulated a number of design goals and requirements:

**Institution:**
Lawrence Berkeley Laboratory  
University of California  
Berkeley, CA 94720

**Research Title:**  
Real-time detection of network intruders

**Date:** 08 February 1995

**List Price of Digital Products**  
ca. $24,000

we can economically monitor our greatest potential source of attacks by passively watching the DMZ link.

**Key enabler:** donation of DEC Alphas (kudos Jeff Mogul)
Prior to developing Bro, we had significant operational experience with a simpler system based on off-line analysis of tcpdump [JLM89] trace files. Out of this experience we formulated a number of design goals and requirements:

**High-speed, large volume monitoring**

**No packet filter drops**

*No packet filter drops* If an application using a packet filter cannot consume packets as quickly as they arrive on the monitored link, then the filter will buffer the packets for later consumption. However, eventually the filter will run out of buffer, at which point it *drops* any further packets that arrive. From a security monitoring perspective, *drops can completely defeat the monitoring*, since the missing packets might contain exactly the interesting traffic that identifies a network intruder.
Prior to developing Bro, we had significant operational experience with a simpler system based on off-line analysis of tcpdump [JLM89] trace files. Out of this experience we formulated a number of design goals and requirements:

High-speed, large volume monitoring

No packet filter drops

**No packet filter drops** If an application using a packet filter cannot consume packets as quickly as they arrive on the monitored link, then the filter will buffer the packets for later consumption. However, eventually the filter will run out of buffer, at which point it drops any further packets that arrive. From a security monitoring perspective, drops can completely defeat the monitoring, since the missing packets might contain exactly the interesting traffic that identifies a network intruder.
Prior to developing Bro, we had significant operational experience with a simpler system based on off-line analysis of tcpdump [JLM89] trace files. Out of this experience we formulated a number of design goals and requirements:

**High-speed, large volume monitoring**

**No packet filter drops**

**Real-time notification**

**Mechanism separate from policy**

**Extensible**

**Avoid simple mistakes**

**The monitor will be attacked**
Operational Experiences with High-Volume Network Intrusion Detection

Figure 2: Memory required by scan detector on mwn-week-hdr using inactivity timeouts for connections.

(a) Default configuration
(b) With user-level timeouts
Figure 4: Load-levels

![Diagram showing load levels and CPU load over time.]

- **Load-Level**
- **CPU load [%]**

- **Time:**
  - Tue 16:00
  - Tue 17:00
  - Tue 18:00

Legend:
- **CPU load**
- **Threshold**
- **Filter change**
Rethinking Hardware Support for Network Analysis and Intrusion Prevention

1-10 Gbps
Stream Demux
Packet Streams
~10^4 Instances
TCP Stream Reassembly
Assembled Packet Streams
~10^5 Instances
Protocol Analyzers
Event Streams
~10^4 Instances
Per Connection Policies
~10^3 Instances
Filtered Event Streams
Aggregate Policies
Aggregated Event Streams
~10-100 Instances
Global Policies

2006
The NIDS Cluster: Scalable, Stateful Network Intrusion Detection on Commodity Hardware

Fig. 3. Probability densities of backends’ CPU load (left), and probability densities for varying numbers of backends (right).
Shunting: A Hardware/Software Architecture for Flexible, High-Performance Network Intrusion Prevention

The Shunt: An FPGA-Based Accelerator for Network Intrusion Prevention

Figure 1: Shunting Main Architecture. The shunt examines the headers of received packets to determine the associated action: forward, drop, or shunt to the Analysis Engine. The Analysis Engine directly updates the Shunt’s caches to control future processing, and either drops analyzed packets for immediate intrusion prevention or reinjects them once vetted for safety.
An Architecture for Exploiting Multi-Core Processors to Parallelize Network Intrusion Prevention
HILTI: An Abstract Execution Environment for Deep, Stateful Network Traffic Analysis

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Mnemonic</th>
<th>Functionality</th>
<th>Mnemonic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitsets</td>
<td>bitset</td>
<td>Packet i/o</td>
<td>iosrc</td>
</tr>
<tr>
<td>Booleans</td>
<td>bool</td>
<td>Packet classification</td>
<td>classifier</td>
</tr>
<tr>
<td>CIDR masks</td>
<td>network</td>
<td>Packet dissection</td>
<td>overlay</td>
</tr>
<tr>
<td>Callbacks</td>
<td>hook</td>
<td>Ports</td>
<td>port</td>
</tr>
<tr>
<td>Closures</td>
<td>callable</td>
<td>Profiling</td>
<td>profiler</td>
</tr>
<tr>
<td>Channels</td>
<td>channel</td>
<td>Raw data</td>
<td>bytes</td>
</tr>
<tr>
<td>Debug support</td>
<td>debug</td>
<td>References</td>
<td>ref</td>
</tr>
<tr>
<td>Doubles</td>
<td>double</td>
<td>Regular expressions</td>
<td>regexp</td>
</tr>
<tr>
<td>Enumerations</td>
<td>enum</td>
<td>Strings</td>
<td>string</td>
</tr>
<tr>
<td>Exceptions</td>
<td>exception</td>
<td>Structs</td>
<td>struct</td>
</tr>
<tr>
<td>File i/o</td>
<td>file</td>
<td>Time intervals</td>
<td>interval</td>
</tr>
<tr>
<td>Flow control</td>
<td>(No joint prefix)</td>
<td>Timer management</td>
<td>timer_mgr</td>
</tr>
<tr>
<td>Hashmaps</td>
<td>map</td>
<td>Timers</td>
<td>timer</td>
</tr>
<tr>
<td>Hashsets</td>
<td>set</td>
<td>Times</td>
<td>time</td>
</tr>
<tr>
<td>IP addresses</td>
<td>addr</td>
<td>Tuples</td>
<td>tuple</td>
</tr>
<tr>
<td>Integers</td>
<td>int</td>
<td>Vectors/arrays</td>
<td>vector</td>
</tr>
<tr>
<td>Lists</td>
<td>list</td>
<td>Virtual threads</td>
<td>thread</td>
</tr>
</tbody>
</table>

Table 1: HILTI’s main instruction groups.
HILTI: An Abstract Execution Environment for Deep, Stateful Network Traffic Analysis

Traffic Analysis Building Blocks
- Domain-specific data types
- State management
- Concurrent analysis
- Real-time performance
- Robust execution
- High-level components

HILTI Environment
- Built-in first-class networking types
- Containers with state management support
- Domain-specific concurrency model
- Compilation to native code
- Contained execution environment
- Platform for building reusable functionality
- Asynchronous and timer-driven execution
- Incremental processing
- Extensive optimization potential
- Static type system

2014 (2009)
Beyond Pattern Matching: A Concurrency Model for Stateful Deep Packet Inspection

**Network traffic** → **Low-level traffic parsing (per-connection)** → **Event stream**

**IDS LOGIC**

**SINGLE-THREADED IDS**

```java
void run_IDS() {
    while (p = read_packet()) {
        if (p.SYN) {
            count_connections(p);
        }
    }
}
```

**CONCURRENT IDS (LOCK-BASED)**

```java
void run_IDS() {
    i = 0;
    while (p = read_packet()) {
        if (p.SYN) {
            event c = new connectionEvent(p);
            send_event(threads[i], c);
            i = (i+1) % N;
        }
    }
}
```

**CONCURRENT IDS (SCOPE-BASED)**

```java
void run_IDS() {
    while (p = read_packet()) {
        if (p.SYN) {
            event c = new connectionEvent(p);
            send_event(threads[c.src % N], c);
        }
    }
}
```

**DETECTOR**

**SINGLE-THREADED IDS**

```java
void count_connections(packet p) {
    if (++counts[p.src] > THRESH) report_host(p.src);
}
```

**CONCURRENT IDS (LOCK-BASED)**

```java
handler count_connections(connectionEvent c) {
    lock_element(counts[c.src])
    v = ++counts[c.src];
    unlock_element(counts[c.src])
    if (v > THRESH) report_host(c.src);
}
```

**CONCURRENT IDS (SCOPE-BASED)**

```java
handler count_connections(connectionEvent c) {
    if (++counts[c.src] > THRESH) report_host(c.src);
}
```

*Figure 1: Simple portscan detector*
Count Me In: Viable Distributed Summary Statistics for Securing High-Speed Networks

**Fig. 2.** Distributed Architecture.
100G Monitoring

Aashish Sharma
Vincent Stoffer

Bro4Pros
February 19th, 2015
OpenDNS, SF
Real-time notification One of our main dissatisfactions with our initial off-line system was the lengthy delay incurred before detecting an attack. If an attack, or an attempted attack, is detected quickly, then it can be much easier to trace back the attacker (for example, by telephoning the site from which they are coming), minimize damage, prevent further break-ins, and initiate full recording of all of the attacker's network activity.

Real-time notification

Mechanism separate from policy

Extensible

Avoid simple mistakes

The monitor will be attacked
Prior to developing Bro, we had significant operational ex-

Institution:
Lawrence Berkeley Laboratory
University of California
Berkeley, CA  94720

Research Title:
Real-time detection of network intruders

Date: 08 February 1995

List Price of Digital Products  ca. $24,000

...  

$ 2,950 DEFTA-DA  Dual-attach FDDI card. We could instead get the single-attach card, $700 less, DEFTA-AA. The use I see for dual-attach is a possible outgrowth of the project, which is using the machine as an intelligent "reactive" firewall (one which stops forwarding packets belonging to misbehaving sessions).
The Worm Era Begins

1999: LBL enables Bro to automatically block scanners

The peak of "autorooters"

Cybercrime starts to take off
Services Scanned Over Time

<table>
<thead>
<tr>
<th>Service Scanned</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
</tr>
<tr>
<td>SMB</td>
</tr>
<tr>
<td>NetBIOS</td>
</tr>
<tr>
<td>DCE/RPC</td>
</tr>
<tr>
<td>SQL</td>
</tr>
<tr>
<td>FTP</td>
</tr>
<tr>
<td>SMTP</td>
</tr>
<tr>
<td>SSH</td>
</tr>
<tr>
<td>SUN/RPC</td>
</tr>
<tr>
<td>Telnet</td>
</tr>
<tr>
<td>9898/tcp (Sasser backdoor)</td>
</tr>
<tr>
<td>LPR</td>
</tr>
<tr>
<td>DNS</td>
</tr>
<tr>
<td>Other</td>
</tr>
</tbody>
</table>

Year:


A Brief History of Scanning

2007
Real-time notification One of our main dissatisfactions with our initial off-line system was the lengthy delay incurred before detecting an attack. If an attack, or an attempted attack, is detected quickly, then it can be much easier to trace back the attacker (for example, by telephoning the site from which they are coming), minimize damage, prevent further break-ins, and initiate full recording of all of the attacker's network activity.

This is not to discount the enormous utility of keeping extensive, permanent logs of network activity for later analysis. Invariably, when we have suffered a break-in, we turn to these logs for retrospective damage assessment, sometimes searching back a number of months.
Building a Time Machine for Efficient Recording and Retrieval of High-Volume Network Traffic

Enriching Network Security Analysis with Time Travel

Figure 1: Log-log CCDF of connection sizes
Mechanism separate from policy  Sound software design often stresses constructing a clear separation between mechanism and policy; done properly, this buys both simplicity and flexibility. The problems faced by our system demand such complexity.

Extensible  Because there are an enormous number of different network attacks, with who knows how many waiting to be discovered, the system clearly must be designed in order to make it easy to add to it knowledge of new types of attacks. In addition, while our system

Mechanism separate from policy

Extensible

Avoid simple mistakes

The monitor will be attacked
A Lone Wolf No More: Supporting Network Intrusion Detection with Real-Time Intelligence
## Through the Eye of the PLC: Semantic Security Monitoring for Industrial Processes

**Table 1: Summary of plausible attacks against PLC implementations: Modbus example**

<table>
<thead>
<tr>
<th>Level</th>
<th>Impact</th>
<th>Attack description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Data integrity</td>
<td>Corrupt integrity by adding data to the packet.</td>
<td>Craft a packet that has a different length than defined in parameters or in spec [2].</td>
</tr>
<tr>
<td>2</td>
<td>IT System</td>
<td>Analyse functionality a PLC implements.</td>
<td>Probe FC, listen for responses and exceptions [2].</td>
</tr>
<tr>
<td></td>
<td>Reconnaissance</td>
<td>Analyse functionality a PLC implements.</td>
<td>Probe FC, listen for responses and exceptions [2].</td>
</tr>
<tr>
<td></td>
<td>Integrity</td>
<td>Exploit lack of specification compliance.</td>
<td>Manipulate application parameters within spec (e.g., offset) or outside of spec (e.g., illegal FC) [2, 9, 37].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Perform unauthorized use of an administrative command.</td>
<td>Use FC 8-0A to clear counters and diagnostics audit [2].</td>
</tr>
<tr>
<td></td>
<td>Denial of service</td>
<td>Perform MITM to enforce system delay.</td>
<td>Send exception codes 05, 06 or FC 8-04 to enforce Listen mode [2].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Perform unauthorized use of administrative command.</td>
<td>Use FC 8-01 to restart TCP communication [2, 9].</td>
</tr>
<tr>
<td>3</td>
<td>Process</td>
<td>Analyse structure of memory map.</td>
<td>Probe readable/writable points. Exceptions tell process implementation details [2].</td>
</tr>
<tr>
<td></td>
<td>Reconnaissance</td>
<td>Analyse structure of memory map.</td>
<td>Probe readable/writable points. Exceptions tell process implementation details [2].</td>
</tr>
<tr>
<td></td>
<td>Direct control</td>
<td>Perform change on process variable.</td>
<td>Write inverted or min/max values [10]. Modify key setpoint variables [14, 26].</td>
</tr>
<tr>
<td></td>
<td>Indirect control</td>
<td>Tamper with process values.</td>
<td>Replay values [14].</td>
</tr>
</tbody>
</table>

*FC: Function code defining the type of functionality in Modbus.*

*MITM: Man-in-the-middle attack.*
Rapid and Scalable ISP Service Delivery through a Programmable MiddleBox

MAdFraud: Investigating Ad Fraud in Android Applications

Pitfalls in HTTP Traffic Measurements and Analysis

Investigating IPv6 Traffic
What happened at the World IPv6 Day?

Exploring EDNS-Client-Subnet Adopters in your Free Time*

On Modern DNS Behavior and Properties

Enabling Content-aware Traffic Engineering

Pushing CDN-ISP Collaboration to the Limit

100+ more at https://www.bro.org/research/index.html
Prior to developing Bro, we had significant operational experience with a simpler system based on off-line analysis of

Avoid simple mistakes  Of course, we always want to avoid mistakes. However, here we mean that we particularly desire that the way that a site defines its security policy be both clear and as error-free as possible. (For example, we would not consider expressing the policy in C code as meeting these goals.)

Mechanism separate from policy

Extensible

Avoid simple mistakes

The monitor will be attacked
Glish: A User-Level Software Bus for Loosely-Coupled Distributed Systems
2002: &attributes – state management, persistence, defaults, file rotation, logging

2002: IPv6 (due to ESnet need)

2002/2004: ALERT’s / NOTICE’s

2002: Signature engine

2003: modules

2005: BinPAC – DSL for protocol analyzers

2006: “when” statement

2009: BroControl

2012: “hook” construct
Frameworks

- File Analysis
- GeoLocation
- Input Framework
- Intelligence Framework
- Logging Framework
- Notice Framework
- Signature Framework
- Summary Statistics
- Broker-Enabled Communication Framework
Prior to developing Bro, we had significant operational experience with a simpler system based on off-line analysis of tcpdump [JLM89] trace files. Out of this experience we formulated a number of design goals and requirements:

**The monitor will be attacked** We must assume that attackers will (eventually) have full knowledge of the techniques used by the monitor, and access to its source code, and will use this knowledge in attempts to subvert or overwhelm the monitor so that it fails to detect the attacker’s break-in activity. This assumption significantly complicates the design of the monitor, but failing to address it is to build a house of cards.

Avoid simple mistakes

**The monitor will be attacked**
Part II: Project Evolution
Packet traces I’m gathering for research become of interest for operational security analysis.
Utility of on-going/real-time monitoring at LBL leads to designing & developing Bro; enabled by hardware grant from DEC.
Summer-long tracking of spoofing/NFS “cracker” provides our first large-scale incident since *Cuckoo’s Egg*.
First Bro semi-public release; Paper appears in USENIX Security; "cat ~/.bash_history >documentation.txt"
Interest in Bro

First Bro user manual sort of

Downloads

Year

Interest in Bro

First release of Snort

Downloads

Year


0 5000 10000 15000

Papers
Distros
Interest in Bro

![Graph showing interest in Bro with Snort paper appearing in USENIX Lisa]

Snort – Lightweight Intrusion Detection for Networks

Martin Roesch – Stanford Telecommunications, Inc.

USENIX Technical Program - 13th Systems Administration Conference - LISA '99

Snort paper appears in USENIX Lisa
Interest in Bro

Download counts for papers and distros over years 1994 to 2002.

- Sourcefire founded – commercial support for Snort

Year:
- 1994
- 1996
- 1998
- 2000
- 2002

Downloads:
- 0
- 5000
- 10000
- 15000

Legend:
- Papers (○)
- Distros (△)
Interest in Bro

Downloads

Year


Just 3 years!

Papers
Distros

SOURCEfire®
Interest in Bro

Robin Sommer begins working on Bro as a student
Robin Sommer begins working on Bro as a student; interns at ICSI
Interest in Bro

Downloads

Year

Bro tutorials at CCS & Supercomputing


0

5000

10000

15000

Papers

Distros
3-year grant begins for Bro work via NSF Strategic Technologies for the Internet program.
**Award Abstract #0334088**

**STI: Viable Network Defense for Scientific Research Institutions**

| **NSF Org:** | **ACI**  
<table>
<thead>
<tr>
<th></th>
<th>Div Of Advanced Cyberinfrastructure</th>
</tr>
</thead>
</table>
| **Program Manager:** | Kevin L. Thompson  
|               | ACI Div Of Advanced Cyberinfrastructure  
|               | CSE Direct For Computer & Info Scie & Enginr |
| **Start Date:** | November 1, 2003 |
| **End Date:** | October 31, 2007 (Estimated) |
| **Awarded Amount to Date:** | $900,000.00 |
| **Investigator(s):** | Vern Paxson vern@icsi.berkeley.edu (Principal Investigator) |
3-year grant begins for Bro work via NSF Strategic Technologies for the Internet program; it includes “nucleate a Bro development community”, but as \( \approx 10\% \) of overall effort, insufficiently funded; But does yield a steady stream of papers
Interest in Bro

- **Download Count**
  - Papers: Blue circles
  - Distros: Red triangles

- **Year Range**:
  - 1998 to 2006

- **Key Event**:
  - DOE regime change cancels Bro Lite

- **Graph Key**:
  - Paper download count increases sharply after 2002.
  - Distros download count shows a similar trend but with a delay.

- **Note**:
  - Data points indicate a significant rise in interest from 2002 onwards.
Driven by LBNL operational need, work begins on “Bro Cluster”; Puts Bro ahead in the “scaling game”; Leads to development of “Bro Control” (*operator-oriented*); Hard to sell as research 😞
We pitch a large-scale continuation of the Bro project to NSF.
Award Abstract #0627320

**CT-T: Approaches to Network Defense Proven in Open Scientific Environments**

<table>
<thead>
<tr>
<th>NSF Org:</th>
<th>CNS Division Of Computer and Network Systems</th>
</tr>
</thead>
</table>
| Program Manager: | Carl Landwehr  
CNS Division Of Computer and Network Systems  
CSE Direct For Computer & Info Scie & Enginr |
| Start Date:    | October 1, 2006 |
| $1,999,054?    | End Date: September 30, 2009 (Estimated) |
| Awarded Amount to Date: | $236,066.00 |
| Investigator(s): | Vern Paxson vern@icsi.berkeley.edu (Principal Investigator)  
Mark Allman (Co-Principal Investigator)  
Robin Sommer (Co-Principal Investigator) |
Interest in Bro

DHS goes with Suricata rather than Bro

Downloads

Year

NSF SDCI program comes on our radar; We discover NCSA is thinking similarly for Blue Waters supercomputer facility and decide to partner for 3-year proposal
<table>
<thead>
<tr>
<th>Field</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NSF Org:</strong></td>
<td>ACI</td>
</tr>
<tr>
<td></td>
<td>Div Of Advanced Cyberinfrastructure</td>
</tr>
<tr>
<td><strong>Program Manager:</strong></td>
<td>Anita Nikolich</td>
</tr>
<tr>
<td></td>
<td>ACI Div Of Advanced Cyberinfrastructure</td>
</tr>
<tr>
<td></td>
<td>CSE Direct For Computer &amp; Info Scie &amp; Enginr</td>
</tr>
<tr>
<td><strong>Start Date:</strong></td>
<td>September 1, 2010</td>
</tr>
<tr>
<td><strong>End Date:</strong></td>
<td>August 31, 2014 (Estimated)</td>
</tr>
<tr>
<td><strong>Awarded Amount to Date:</strong></td>
<td>$2,995,905.00</td>
</tr>
<tr>
<td><strong>Investigator(s):</strong></td>
<td>Robin Sommer <a href="mailto:robin@icsi.berkeley.edu">robin@icsi.berkeley.edu</a> (Principal Investigator)</td>
</tr>
<tr>
<td></td>
<td>Vern Paxson (Co-Principal Investigator)</td>
</tr>
<tr>
<td></td>
<td>Adam Slagell (Co-Principal Investigator)</td>
</tr>
</tbody>
</table>
More specifically, this project (1) improves the perspective of Bro's end-users by providing extensive up-to-date documentation and support, and refining many of the rough edges that the system has accumulated over time; (2) unifies and modernizes Bro's current code base that has evolved over 14 years of active development; (3) improves Bro's processing performance to the degree required for operation in current and future large-scale scientific environments; and (4) adds new data analysis functionality in the form of a highly interactive graphical user interface and a transparent database.

Investigator(s): Robin Sommer robin@icsi.berkeley.edu (Principal Investigator)
Vern Paxson (Co-Principal Investigator)
Adam Slagell (Co-Principal Investigator)
# SDCI Sec Improvement: Enhancing Bro for Operational Network Security Monitoring in Scientific Environments

<table>
<thead>
<tr>
<th>NSF Org:</th>
<th>ACI Div Of Advanced Cyberinfrastructure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Manager:</td>
<td>Cyberinfrastructure Center &amp; Info Scie &amp; Engnr</td>
</tr>
<tr>
<td>Start Date:</td>
<td></td>
</tr>
<tr>
<td>End Date:</td>
<td>(terminated)</td>
</tr>
<tr>
<td>Awarded Amount to Date:</td>
<td></td>
</tr>
<tr>
<td>Investigator(s):</td>
<td>Robin Sommer <a href="mailto:robin@icsi.berkeley.edu">robin@icsi.berkeley.edu</a> (Principal Investigator) Vern Paxson (Co-Principal Investigator) Adam Slagell (Co-Principal Investigator)</td>
</tr>
</tbody>
</table>
NSF SDCI program comes on our radar; We discover NCSA is thinking similarly for Blue Waters supercomputer facility; decide to partner for 3-year proposal; **Major Luck #1**: Seth is available to hire! **Major Luck #2**: new collaboration *gels* highly effectively!
NSF works with us to foster continuation of Bro project & community.
### Award Abstract #1348077

**A Bro Center of Expertise for the NSF Community**

<table>
<thead>
<tr>
<th>NSF Org:</th>
<th>ACI Div Of Advanced Cyberinfrastructure</th>
</tr>
</thead>
</table>
| Program Manager: | Kevin L. Thompson  
ACI Div Of Advanced Cyberinfrastructure  
CSE Direct For Computer & Info Scie & Enginr |
| Start Date: | October 1, 2013 |
| End Date: | September 30, 2016 (Estimated) |
| Awarded Amount to Date: | $3,360,092.00 |
| Investigator(s): | Robin Sommer robin@icsi.berkeley.edu (Principal Investigator)  
Vern Paxson (Co-Principal Investigator)  
Adam Slagell (Co-Principal Investigator) |
A Bro Center of Expertise for the NSF Community

This activity promotes Bro as a comprehensive, low-cost security capability for these communities; providing guidance and support on all aspects of a Bro installation. The project devises reference scenarios for deployment and integration; and develops novel technical capabilities that cater to NSF environments. The project supports existing Bro users in optimizing and extending their setups, and makes Bro's capabilities available to new sites and projects that lack the resources to deploy Bro effectively on their own. At a technical level, the project is the focal point of Bro's open-source development, maintaining its code base and documentation. To the research community, the project acts as a facilitator for transitioning networking research results into practice by leveraging Bro as a deployment platform.

Investigator(s): Robin Sommer robin@icsi.berkeley.edu (Principal Investigator)
Vern Paxson (Co-Principal Investigator)
Adam Slagell (Co-Principal Investigator)
@Bro_IDS Twitter Followers

Growth = 500+/year
@Bro_IDS Twitter Followers

Growth = 1,100+/year
Arrival of Open Source Contributors

- **All**
- **>= 10 Contributions**

# Contributors vs Year

- Y-axis: Number of Contributors

- The number of contributors has increased significantly from 1995 to 2015.
- There is a notable increase in the number of contributors who have made at least 10 contributions.
Looking Forward

• **Visibility:** *Deep Bro*
  – Extensive interior site deployment
  – Enterprise protocols; distributed coordination

• **Performance:** *HILTI + Spicy*
  – Compiled multithreaded/multicore Bro

• **Archive:** *VAST (Visibility Across Space and Time)*
  – Very high-performance event/logging archive
  – To support interactive forensic analysis …
  – … and capture of IOCs

• **Longevity & Support:** *Bro Foundation*
  – Via Software Freedom Conservancy