Leader: Defense Against Exploit-Based Denial-of-Service Attacks on Web Servers

STEEL Security Research Lab@ISI
Information Sciences Institute,
University of Southern California

-Dr. Jelena Mirkovic
sunshine@isi.edu

-Dr. Christophe Hauser
hauser@isi.edu

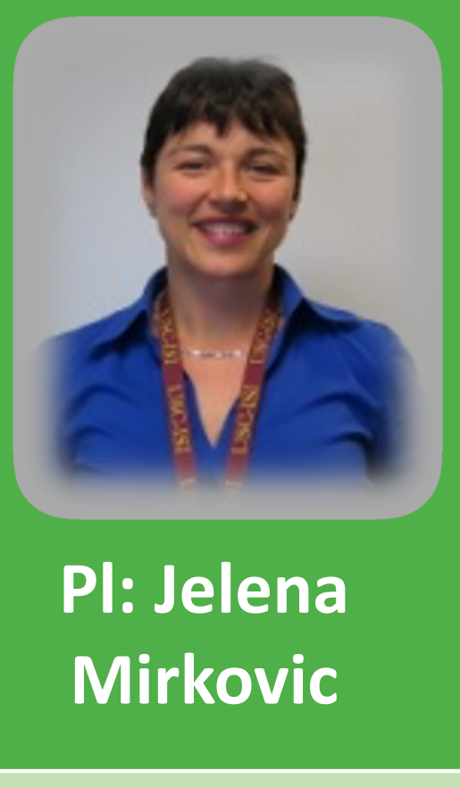
-Rajat Tandon
rajattan@usc.edu

-Haoda Wang
haodawan@usc.edu

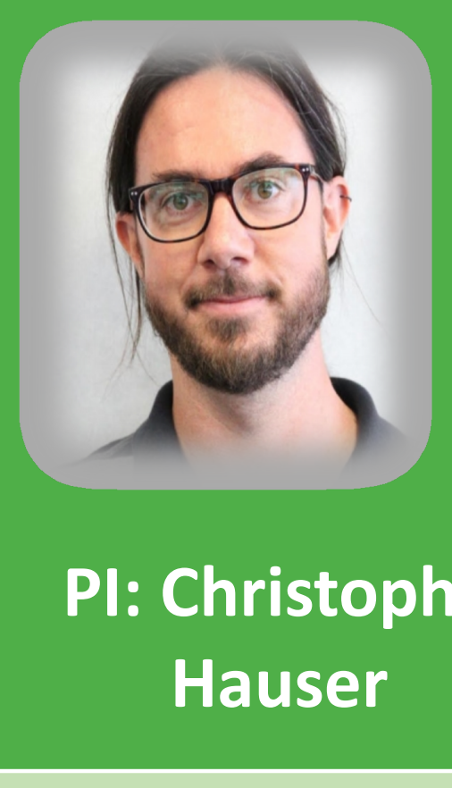
-Nicolaas Weideman
nweidema@usc.edu

- Shushan Arakelyan
shushana@usc.edu

-Dr. Genevieve Bartlett
bartlett@isi.edu



PI: Jelena Mirkovic



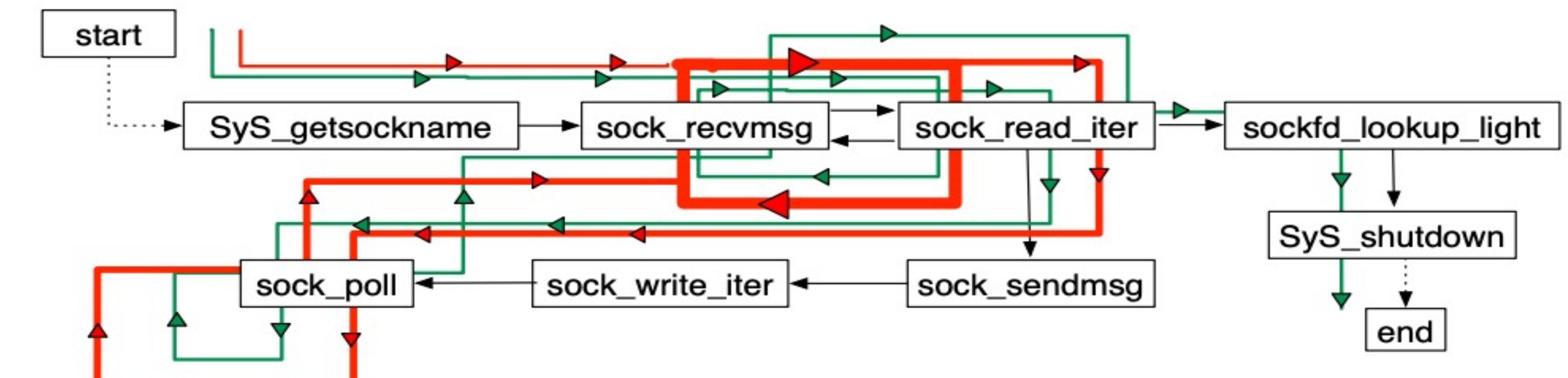
PI: Christophe Hauser

Exploit-based DoS attacks: exploit some vulnerability at the target application and are effective at a low request rate (e.g., 100 requests/second)

Attack detection at the network level is very challenging
- Attacks consume little bandwidth and appear legitimate

Such attacks are usually mitigated piecemeal
- Custom-tailored defenses are proposed for each attack variant
- E.g., SlowLoris defenses, hash collision defenses, infinite recursion defenses

The common problems with custom tailored defenses are:
- Low portability to other applications and attacks
- Low chance of adoption



Sample legitimate and attack connection life stages.

Leader is an application-agnostic and attack-agnostic defense

It monitors all external requests for running services at the end host

Behavior Profiling

During normal operations, Leader learns how each application's requests use host resources and builds ***baseline models per application***

Attack Detection

- Continuously, Leader builds instantaneous profiles of how each request uses host resources
  - Compares them to baseline models
  - Deviations signal attacks, which Leader blocks

Connection Life Stages

- For each connection, Leader builds
  - a fine-grained pattern of resource consumption
  - by each service as it processes each request,
- We use the tuple <thread id, process id> to uniquely identify a given external (incoming) connection to the **application**
- We then link this tuple to the source IP address and source port of the external client
- A connection's life stage corresponds to a function call of:
  - net/socket.c and the resource usage
  - (e.g: CPU cycles, page faults, file descriptors and memory) per call
- Therefore, Leader is aimed to be application and attack agnostic

CPU cycles

times called

memory

call duration

file descriptors

page faults

Connection Life Stages

Each life stage pattern is a snapshot of the function call sequence and resource usages until the given moment in time.

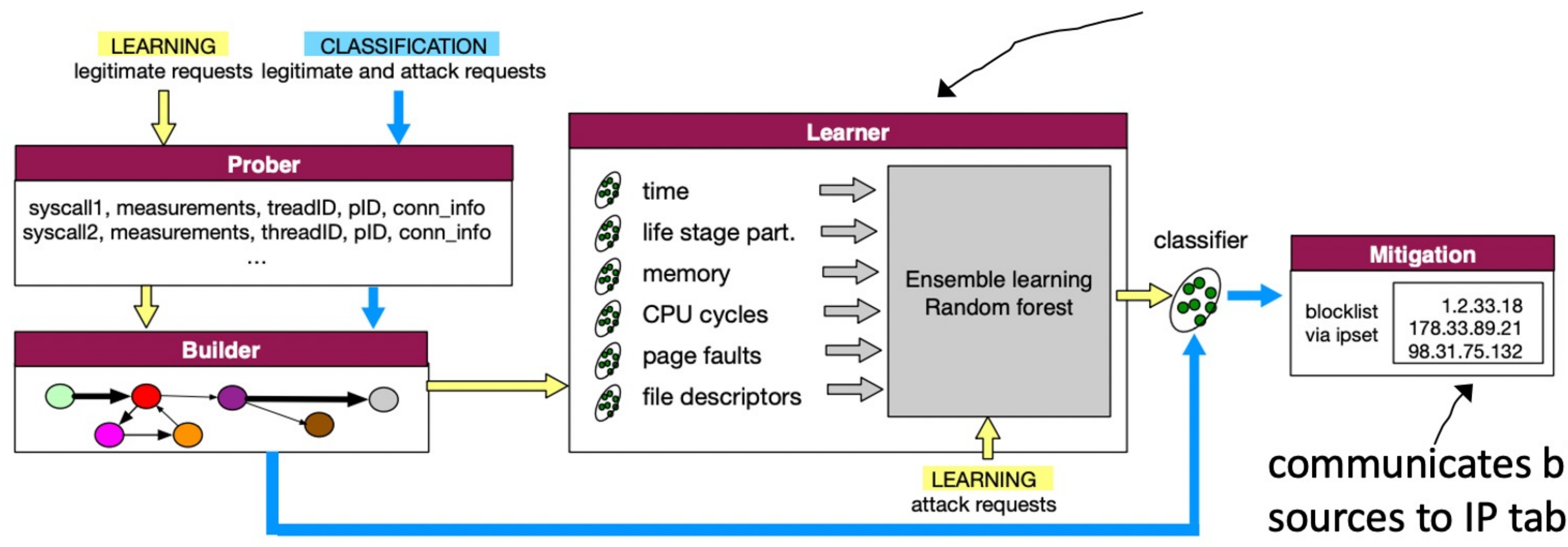
The red lines show the transitions that differ in duration or frequency between an attack and a legitimate connection.

Different attacks may follow different sequences and consume different amount of resources at different stages.

Leader's Operation: Learning and Classification

collects the data needed for learning/classification

generates the baseline model of legitimate client behavior, and classifies connections using Elliptic Envelope



communicates blocked sources to IP tables

continuously builds connection life stage sequences

Leader: Experimental Setup

- Server:
  - Mirrored websites—Imgur (Apache2), Wikipedia (Nginx)
  - Added Web pages with vulnerabilities on these sites
  - Crafted an implementation of a vulnerable Web application in Flask
- Legitimate traffic:
  - MTurk study to gather training data for learning the baseline models
  - Legitimate clients replay requests from logs in a congestion-responsive manner
  - Multiplex multiple source IPs on a single physical machine
  - Emulate 100 clients being active simultaneously

Attack Scenarios

Slowloris (SL): uses partial HTTP requests to open connections between the attacker and the Web server for as long as possible.

Hash Collision (HC): uses Web requests with colliding keys, thus dramatically slowing down the server.

Regular Expression Denial of Service (ReDoS): creates inputs that take inordinately long time to process regular expressions.

preg_replace() PHP Function Exploitation (PHPEx): preg_replace(), PHP function, can lead to a remote code execution if the Web application passes user input to it and if that input includes executable PHP code.

Infinite recursive calls denial of service (IRC): Passing a PHP file as an argument to itself can in some cases lead to infinite recursive call.

Maliciously Crafted URL Attack on a Flask Application (MCU): exploit URL parameters to generate hundreds of times larger return values than those of legitimate requests.

Resource usage by the the different stages of a sample legitimate and a sample attack connection

	sample legitimate connection						sample exDoS attack connection					
call	**dur**	**#calls**	**mem**	**CPU cyc.**	**pf**	**fd.**	**dur**	**#calls**	**mem**	**CPU cyc.**	**pf**	**fd**
SyS_getsockname	6.5μs	1	0KB	0.01M	0	0	16μs	1	0KB	0.01M	0	0
sock_recvmsg	789μs	4	0KB	0.1M	0	1	22,939μs	295	0KB	44M	1	1
sock_read_iter	34μs	4	1KB	0.03M	0	2	8,750μs	295	16KB	15M	0	1
sock_sendmsg	415μs	2	1KB	0.1M	0	1	752μs	2	1KB	0.1M	0	0
sock_write_iter	9.8μs	1	1KB	0.01M	0	1	32μs	1	1KB	0.01M	0	1
sock_poll	2,491μs	3	0KB	3M	0	0	11,073,328μs	97	0KB	55M	0	0
sockfd_lookup_light	53μs	3	0KB	0.01M	0	0	120μs	3	0KB	0.01M	0	0
Sys_shutdown	62μs	1	0KB	0.01M	0	0	101μs	1	0KB	0.01M	0	0

Design scenarios and results

Liberal design
- Assumes that each anomalous connection is attack connection
- Ensures fast decision time but if there are any errors in classification, a legitimate source may become blocked by the module

Conservative design
- Requires that a source receives some fraction of anomalous conn. classifications before being blocked
- Reduces misclassification of legitimate sources, at the expense of longer decision time
- We use ROC curves to calibrate our conservative design

measure/scenario	**liberal**						**conservative**					
	SL	**HC**	**Re-DoS**	**PHPEx**	**IRC**	**MCU**	**SL**	**HC**	**Re-DoS**	**PHPEx**	**IRC**	**MCU**
true positive	99.9%	99.9%	99.4%	99.1%	99.9%	100%	99.9%	99.9%	99.4%	99.1%	99.9%	100%
true negative	99.4%	99.2%	98.1%	97.5%	96.9%	99.95%	100%	100%	100%	100%	99.8%	99.95%
false positive	0.6%	0.8%	1.9%	2.5%	3.1%	0.05%	0%	0%	0%	0%	0.2%	0.05%
false negative	0.1%	0.1%	0.6%	0.9%	0.1%	0%	0.1%	0.1%	0.6%	0.9%	0.1%	0%
att. req. before block	1.65	1.92	1.27	1.25	1.44	1.18	5.17	5.32	5.50	5.02	5.84	5.07

Funded by NSF SaTC award # 1815495

https://steel.isi.edu/Projects/Leader/