Could a purpose built supercomputer play DEF CON Capture the Flag?

Mike Walker
Program Manager

Approved for Public Release, Distribution Unlimited

Cyber Competition Challenges

Turing, Rice, & Undecidable Problems:

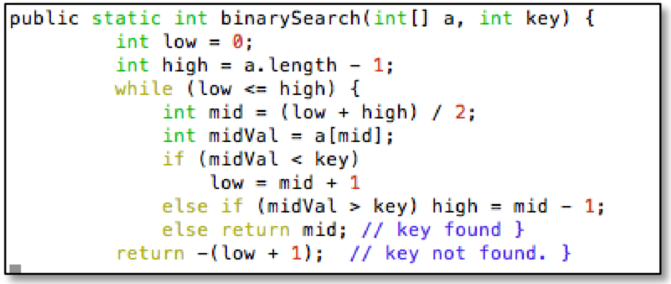
- Is the software correct & secure?
- If not, how incorrect or insecure is it?

Q: Can we *compete* when the answers required to name a victor are undecidable?

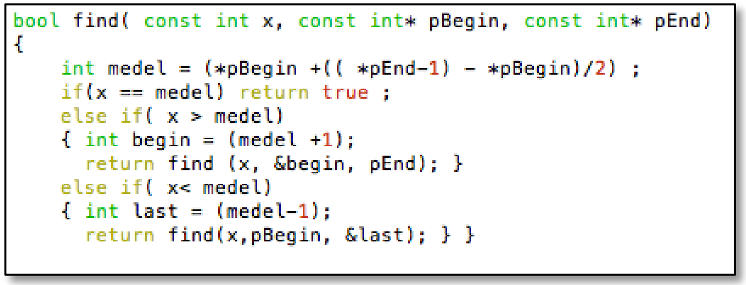
Competitive Programming: TopCoder

1: Construct

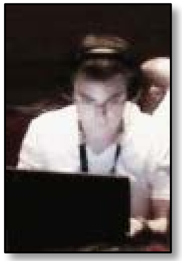
2: Challenge

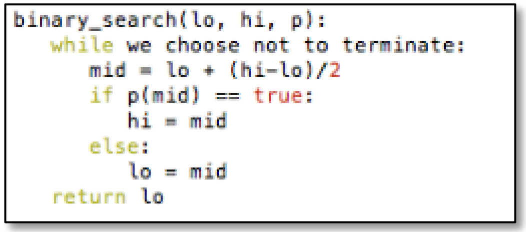


```
public static int binarySearch(int[] a, int key) {
        int low = 0;
        int high = a.length - 1;
        while (low <= high) {
            int mid = (low + high) / 2;
            int midVal = a[mid];
            if (midVal < key)
                low = mid + 1
            else if (midVal > key) high = mid - 1;
            else return mid; // key found }
        return -(low + 1);  // key not found. }
```



```
bool find( const int x, const int* pBegin, const int* pEnd)
{
    int medel = (*pBegin +(( *pEnd-1) - *pBegin)/2) ;
    if(x == medel) return true ;
    else if( x > medel)
    { int begin = (medel +1);
      return find (x, &begin, pEnd); }
    else if( x< medel)
    { int last = (medel-1);
      return find(x,pBegin, &last); } }
```





```
binary_search(lo, hi, p):
   while we choose not to terminate:
      mid = lo + (hi-lo)/2
      if p(mid) == true:
         hi = mid
      else:
         lo = mid
   return lo
```

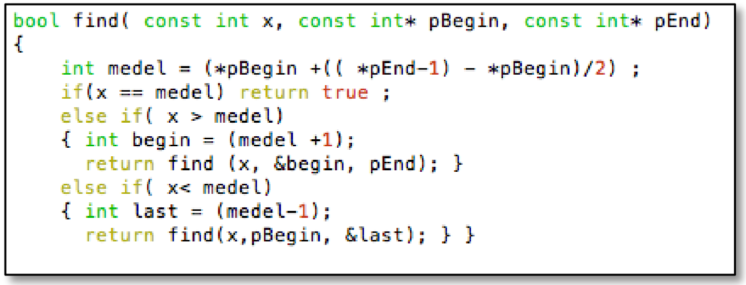
http://technorazzi.com/wp-content/uploads/2010/08/ctf_denmark2.jpg

Approved for Public Release, Distribution Unlimited

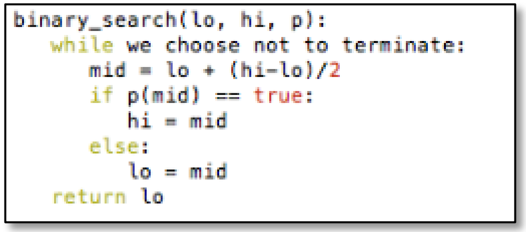
Competitive Programming: TopCoder

1: Construct

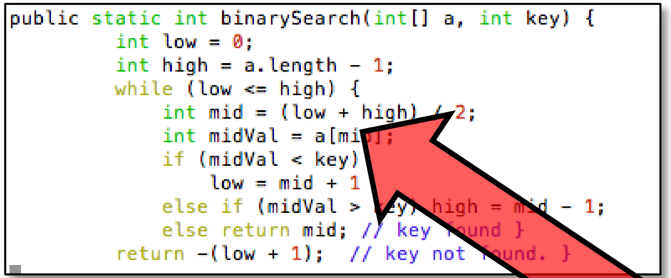
2: Challenge



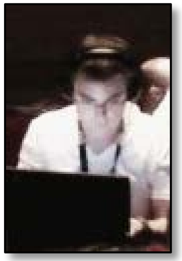
```
bool find( const int x, const int* pBegin, const int* pEnd)
{
    int medel = (*pBegin +(( *pEnd-1) - *pBegin)/2) ;
    if(x == medel) return true ;
    else if( x > medel)
    { int begin = (medel +1);
      return find (x, &begin, pEnd); }
    else if( x< medel)
    { int last = (medel-1);
      return find(x,pBegin, &last); } }
```



```
binary_search(lo, hi, p):
   while we choose not to terminate:
      mid = lo + (hi-lo)/2
      if p(mid) == true:
         hi = mid
      else:
         lo = mid
   return lo
```



```
public static int binarySearch(int[] a, int key) {
        int low = 0;
        int high = a.length - 1;
        while (low <= high) {
            int mid = (low + high) / 2;
            int midVal = a[mid];
            if (midVal < key)
                low = mid + 1
            else if (midVal > key) high = mid - 1;
            else return mid; // key found }
        return -(low + 1);  // key not found. }
```

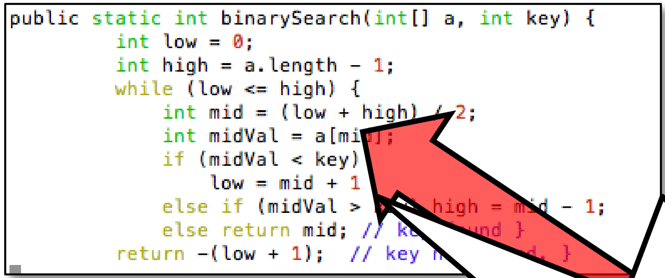


http://technorazzi.com/wp-content/uploads/2010/08/ctf_denmark2.jpg

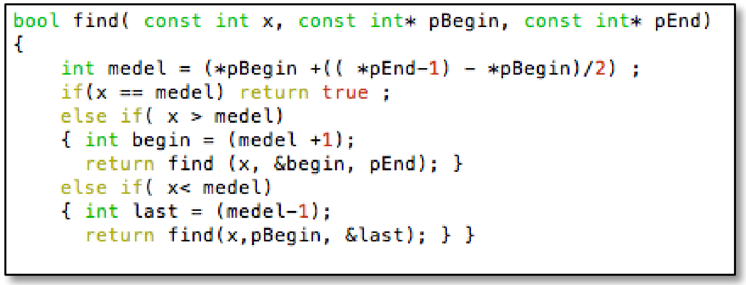
Competitive Programming: TopCoder

1: Construct

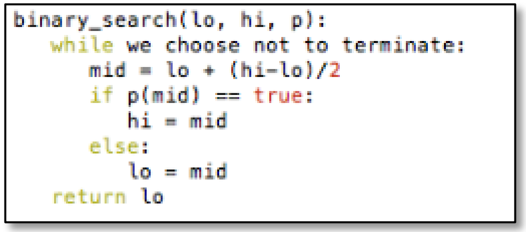
2: Challenge



```
public static int binarySearch(int[] a, int key) {
        int low = 0;
        int high = a.length - 1;
        while (low <= high) {
            int mid = (low + high) / 2;
            int midVal = a[mid];
            if (midVal < key)
                low = mid + 1
            else if (midVal > key) high = mid - 1;
            else return mid; // key found }
        return -(low + 1);  // key not found. }
```



```
bool find( const int x, const int* pBegin, const int* pEnd)
{
    int medel = (*pBegin +(( *pEnd-1) - *pBegin)/2) ;
    if(x == medel) return true ;
    else if( x > medel)
    { int begin = (medel +1);
      return find (x, &begin, pEnd); }
    else if( x< medel)
    { int last = (medel-1);
      return find(x,pBegin, &last); } }
```

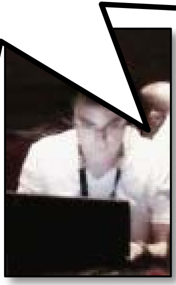


```
binary_search(lo, hi, p):
   while we choose not to terminate:
      mid = lo + (hi-lo)/2
      if p(mid) == true:
         hi = mid
      else:
         lo = mid
   return lo
```

2^{31}

`int mid = (low + high) / 2;`

ArrayIndexOutOfBoundsException *



http://technorazzi.com/wp-content/uploads/2010/08/ctf_denmark2.jpg

*http://googleresearch.blogspot.com/2006/06/extra-extra-read-all-about-it-nearly.html

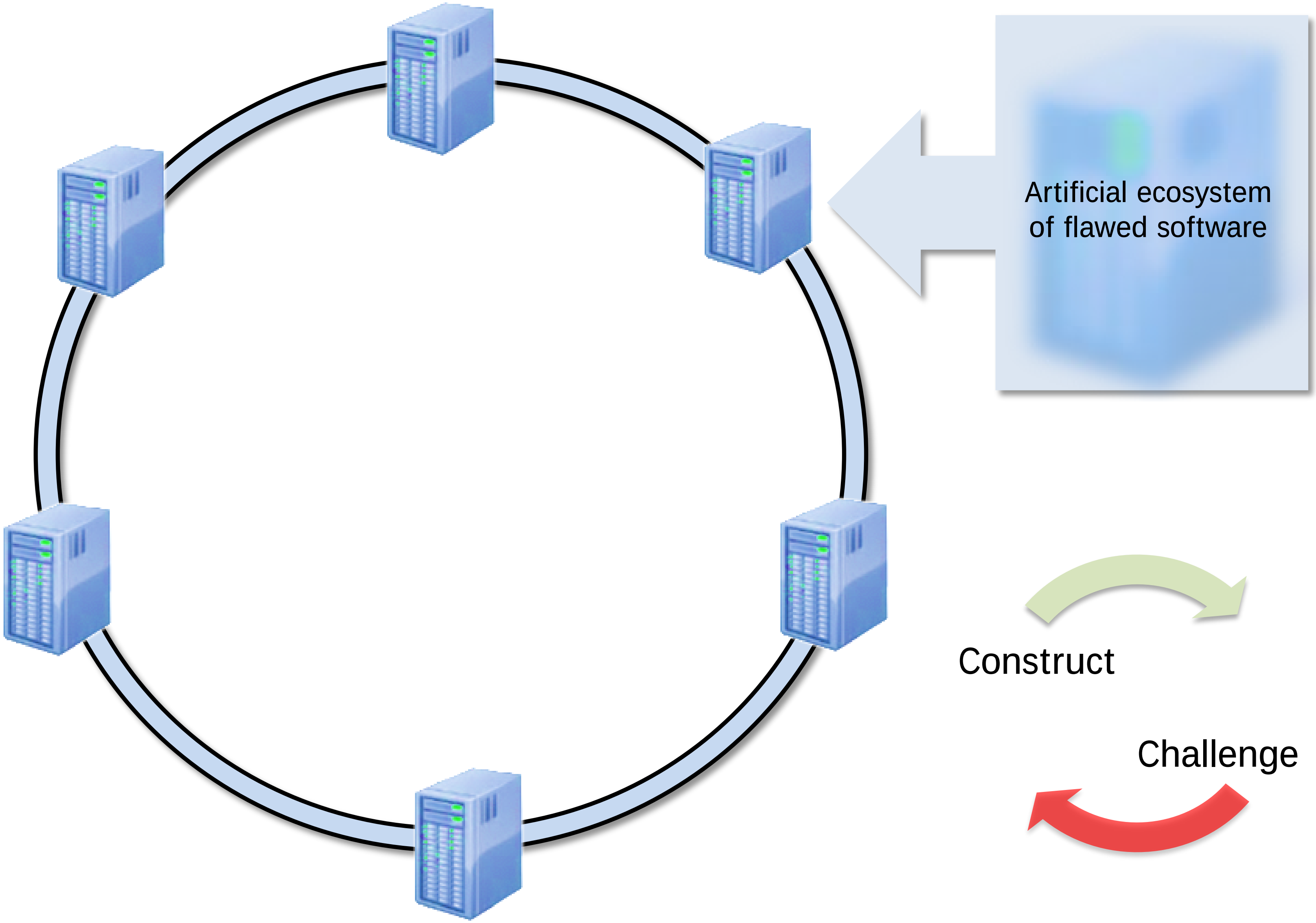
Approved for Public Release, Distribution Unlimited

Competitive Programming

Q: Can we *compete* when the answers required to name a victor are undecidable?

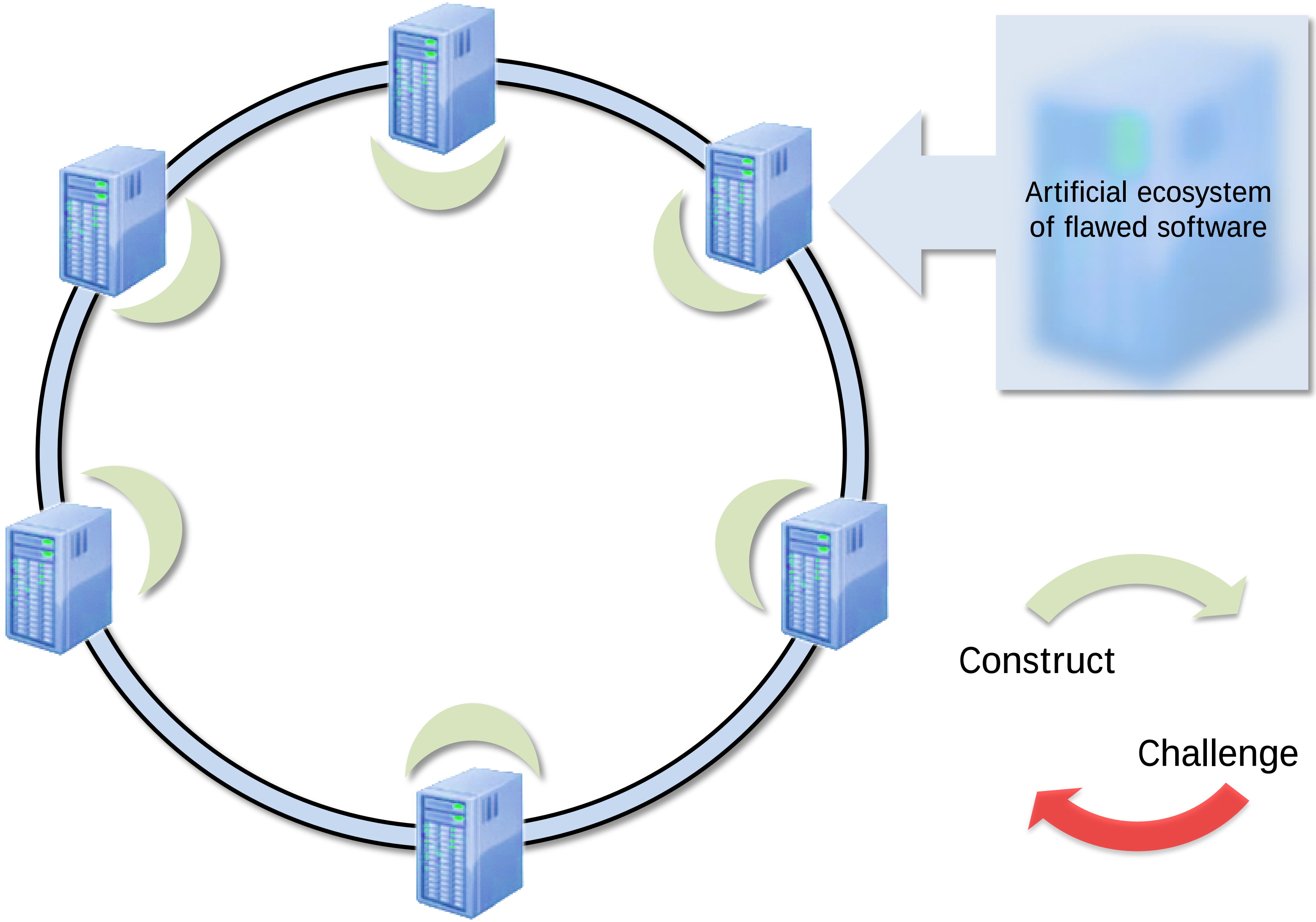
A: *consensus evaluation*

Competitive Computer Security: DEF CON CTF

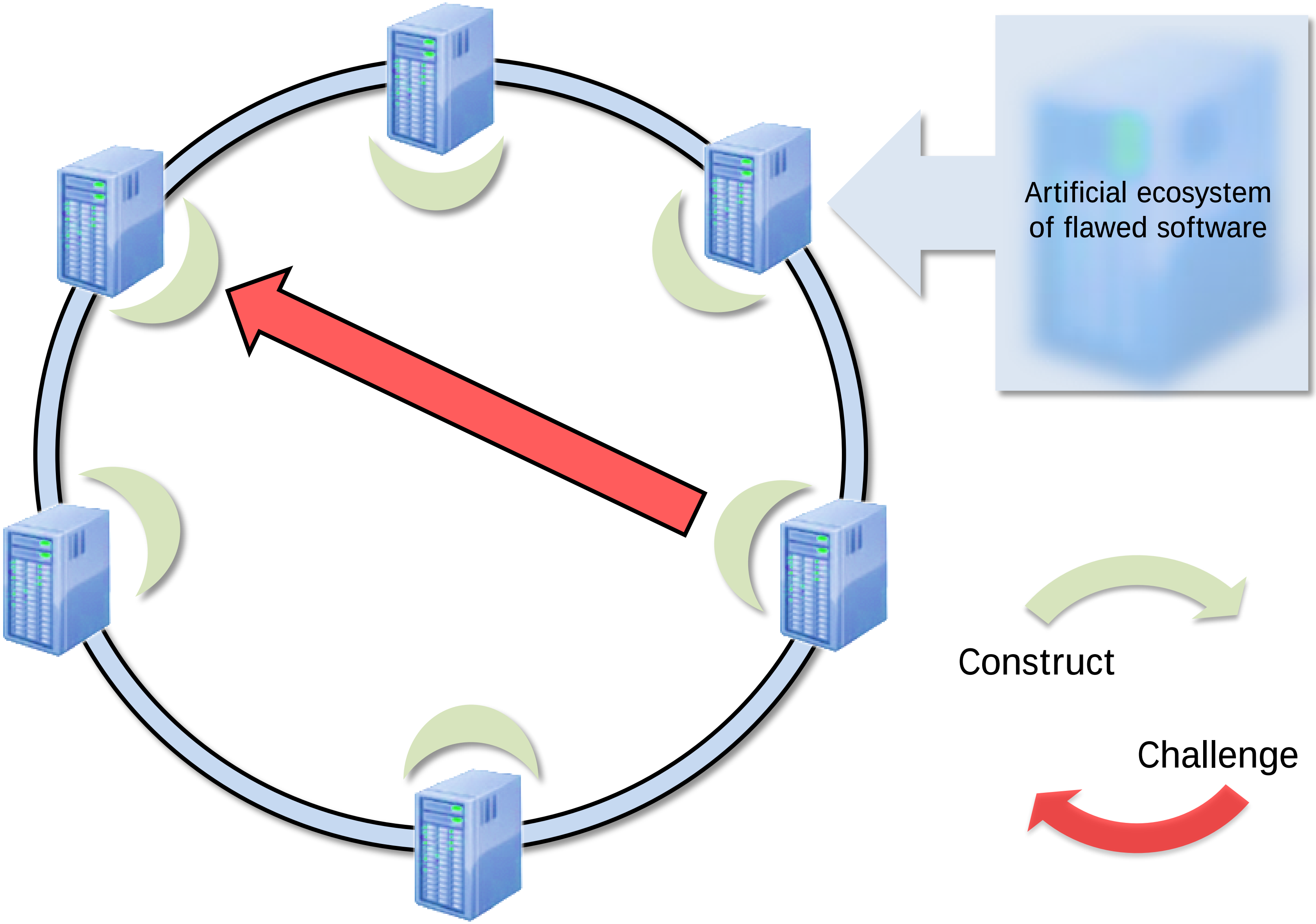


Approved for Public Release, Distribution Unlimited

Competitive Computer Security: DEF CON CTF



Competitive Computer Security: DEF CON CTF



Artificial ecosystem of flawed software

Construct

Challenge

Competition Paradigm

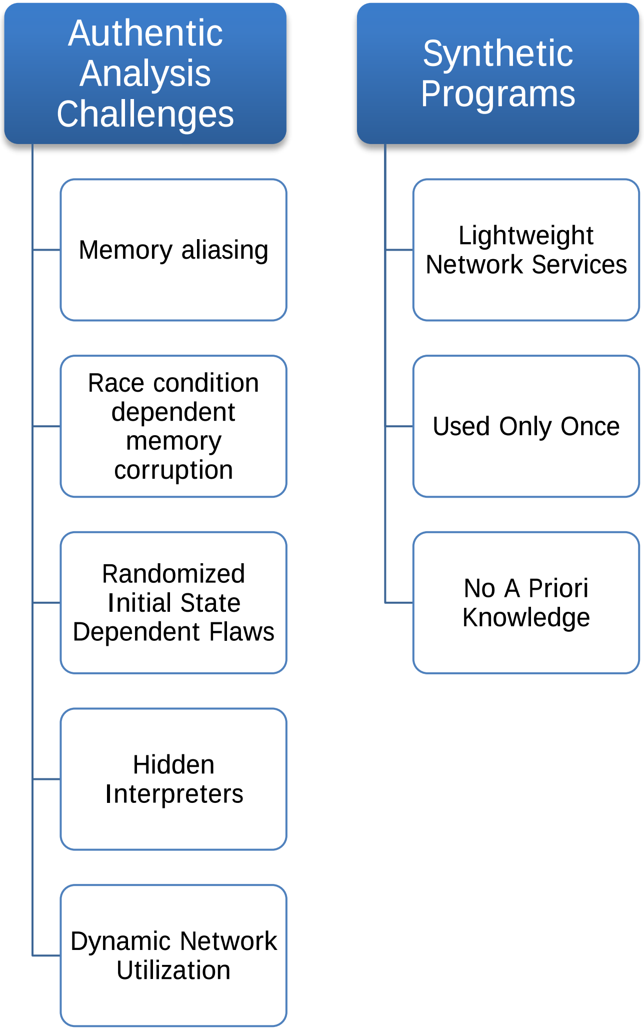
Harness consensus evaluation to identify breakthrough technology.

A tournament for fully automated network defense

DECREE (https://github.com/CyberGrandChallenge)

- DARPA Experimental Cyber Research Evaluation Environment
- Specially Designed Environment
  - 7 System Calls [Garfinkel2003]
    - terminate – end program (exit)
    - transmit – write data to an fd (write)
    - receive – read data from an fd (read)
    - fdwait – wait for fds (select)
    - allocate – allocates memory (mmap)
    - deallocate – releases allocated memory (munmap)
    - random – populate a buffer with random bytes
  - Restricted Inter-Process Communication
    - No shared memory
    - Only socketpairs
      - Clean bidirectional communication
      - Automatically created by system on startup
      - Shared between all processes in an IPC CB

CTF: Alternative Software Ecosystem

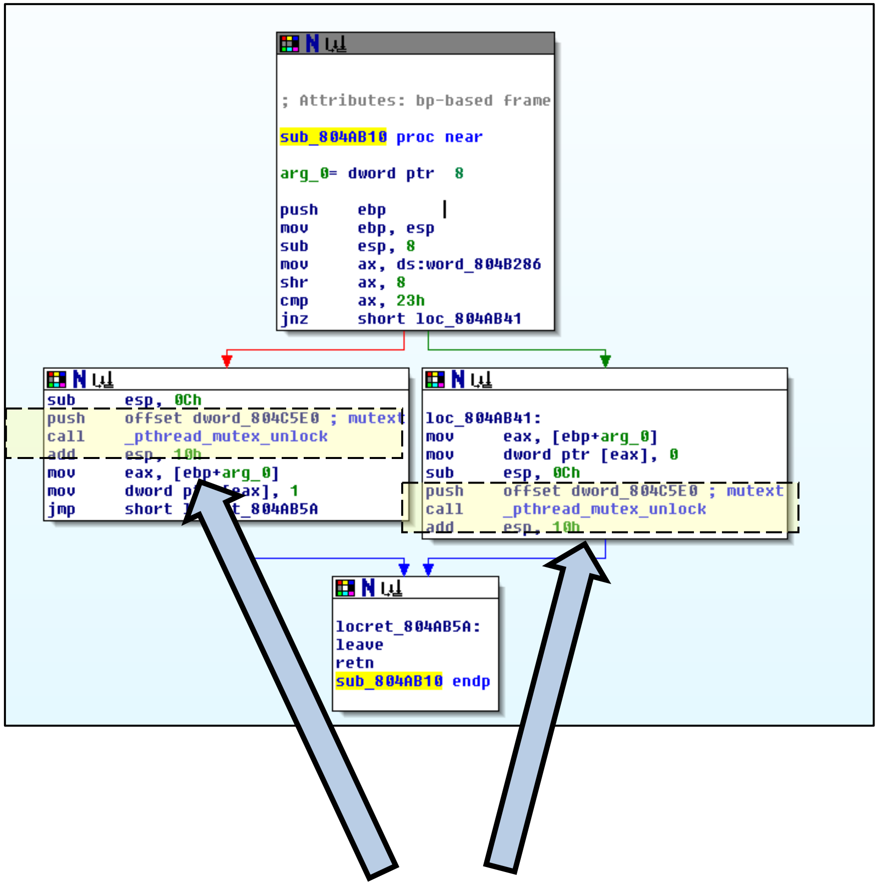


Authentic Analysis Challenges

- Memory aliasing
- Race condition dependent memory corruption
- Randomized Initial State Dependent Flaws
- Hidden Interpreters
- Dynamic Network Utilization

Synthetic Programs

- Lightweight Network Services
- Used Only Once
- No A Priori Knowledge



```
; Attributes: bp-based frame

sub_804AB10 proc near

arg_0= dword ptr  8

push    ebp
mov     ebp, esp
sub     esp, 8
mov     ax, ds:word_804B286
shr     ax, 8
cmp     ax, 23h
jnz     short loc_804AB41
```

```
sub     esp, 0Ch
push    offset dword_804C5E0 ; mutext
call    _pthread_mutex_unlock
add     esp, 10h
mov     eax, [ebp+arg_0]
mov     dword ptr [eax], 1
jmp     short locret_804AB5A
```

```
loc_804AB41:
mov     eax, [ebp+arg_0]
mov     dword ptr [eax], 0
sub     esp, 0Ch
push    offset dword_804C5E0 ; mutext
call    _pthread_mutex_unlock
add     esp, 10h
```

```
locret_804AB5A:
leave
retn
sub_804AB10 endp
```

Defcon CTF Qualifiers 2007
Highest difficulty (500), network application flaw category
Hidden mutex unlock condition triggers timing specific memory corruption*

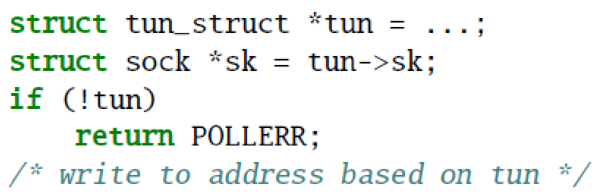
Authentic Skills, Synthetic Software

Approved for Public Release, Distribution Unlimited

*nopsr.us

Challenge Binaries

- No filesystem access, no new network connections, no process creation, no signals, no shared memory
- Userspace only and statically linked [Qu2011]
- Compiled Binaries only (not hand coded)
  - Always available
  - Ground truth



```
struct tun_struct *tun = ...;
struct sock *sk = tun->sk;
if (!tun)
    return POLLERR;
/* write to address based on tun */
```

"A null pointer dereference vulnerability (CVE-2009-1897) in the Linux kernel, where the dereference of pointer tun is before the null pointer check. The code becomes exploitable as **gcc optimizes away** the null pointer check [10]" [Wang2013]

RedHat 7.0	- (default Sendmail 8.11.0)	does not crash
RedHat 7.2	- (default Sendmail 8.11.6)	does not crash
RedHat 7.3 (p)	- (patched Sendmail 8.11.6)	does not crash
RedHat 7.0	- (self compiled Sendmail 8.11.6)	crashes
RedHat 7.2	- (self compiled Sendmail 8.11.6)	crashes
RedHat 7.3	- (self compiled Sendmail 8.11.6)	crashes
Slackware 8.0 (p)	- (patched Sendmail 8.11.6 binary)	crashes
Slackware 8.0	- (self compiled Sendmail 8.12.7)	does not crash
RedHat 7.x	- (self compiled Sendmail 8.12.7)	does not crash

(p) - patched box

"Due to the nature of the overflowed buffer declaration (static), exploitation of this issue is **highly dependent on the way compiler orders the static data** in the data segment" [LSD2003]

  - Wide availability of "lifters" (these are open source x86)
    - BAP (BAP IR) - http://bap.ece.cmu.edu/
    - BitBlaze (VINE IR) - http://bitblaze.cs.berkeley.edu/
    - McSema (LLVM IR) - "It is in the process of being open sourced" [Dinaburg2014]
    - QEMU (TCG IR) – http://www.qemu.org/
    - Valgrind (VEX IR) – http://www.valgrind.org/

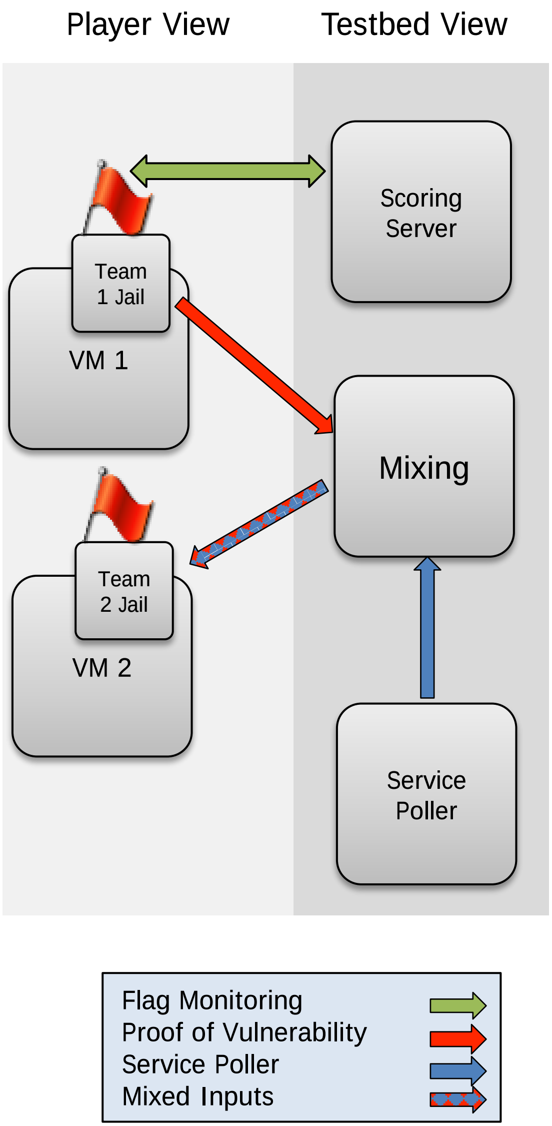
Reproducibility

> “Evaluating a non-trivial idea is beyond the time budget of any single paper as this requires running many benchmarks on **multiple implementations with different hardware and software platforms**. Often a careful comparison to the state of the art means implementing competing solutions. The result of this state of affairs is that papers presenting potentially useful novel ideas regularly appear without a **comparison to the state of the art**, **without appropriate benchmarks**, without any mention of limitations, and **without sufficient detail to reproduce the experiments**. This hampers scientific progress and perpetuates the cycle.” [Vitek2011]

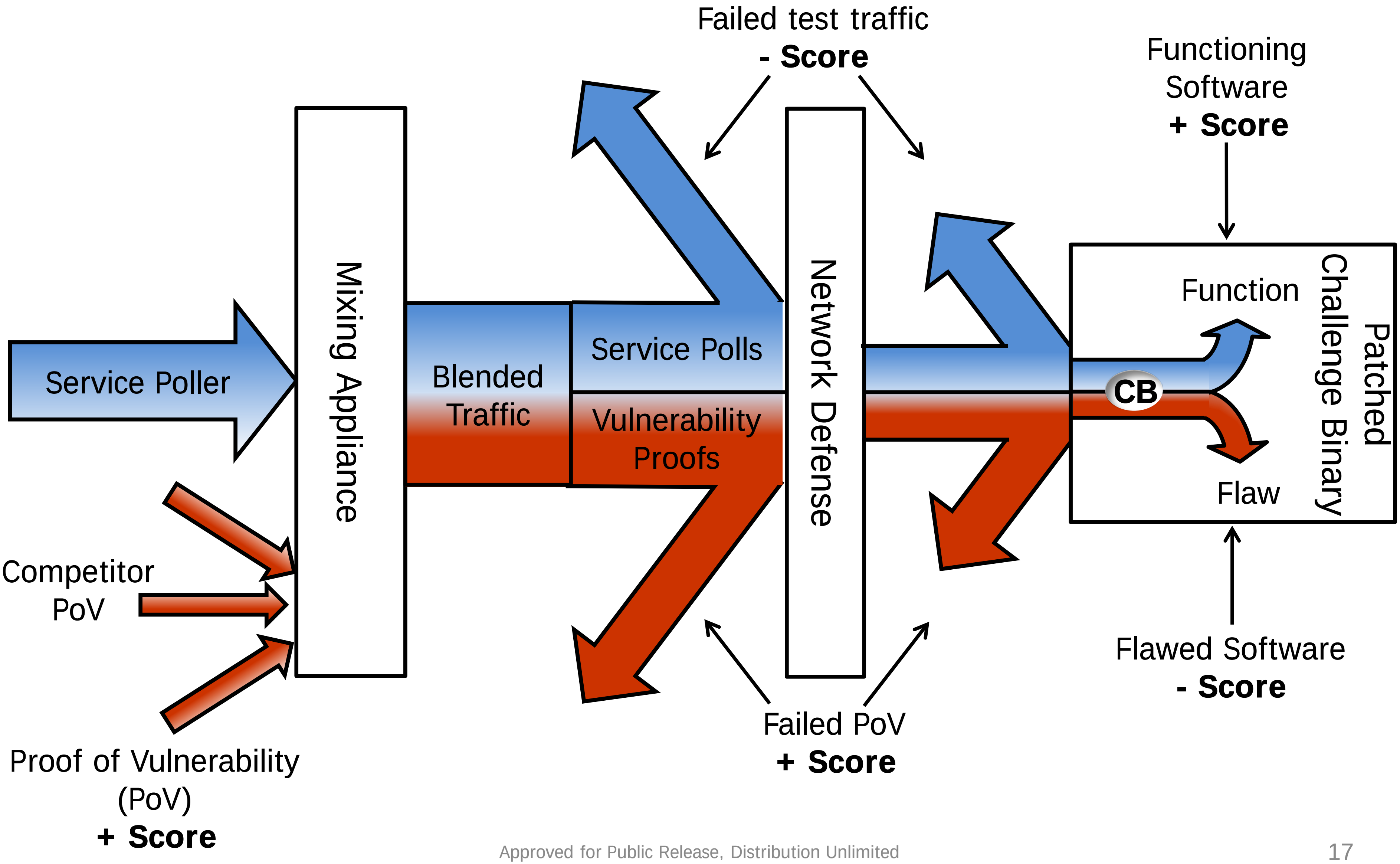
- DARPA’s Intentions
  - One single software platform – DECREE
  - One single hardware architecture – x86
  - One large set of benchmarks (~200) – Challenge Binaries
    - Source code, Vulnerable Binary, Patched Binary, Deterministic Proof(s) Of Vulnerability, Polls
    - Specially designed by the authors to distinguish between techniques
  - Large set of data from the events
    - Network traffic captures
    - Competitor patched binaries, actual POVs used, etc.

CTF: Real World Challenges

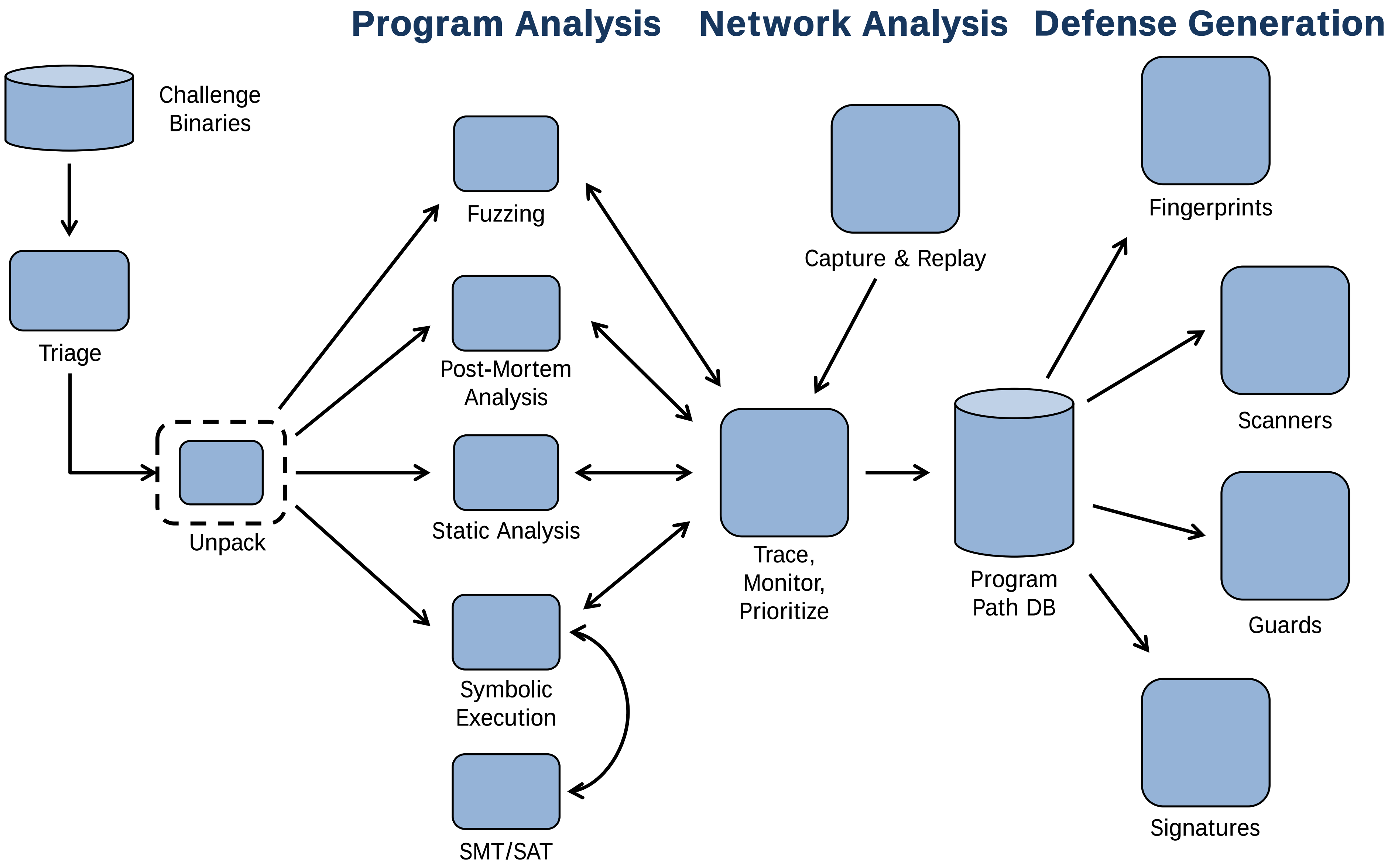
Challenges	CTF
Attribution & Reputation	Network Mixing
Resilience	New Flags Random Intervals
Availability	Service Poller



CTF: Real Time Defense

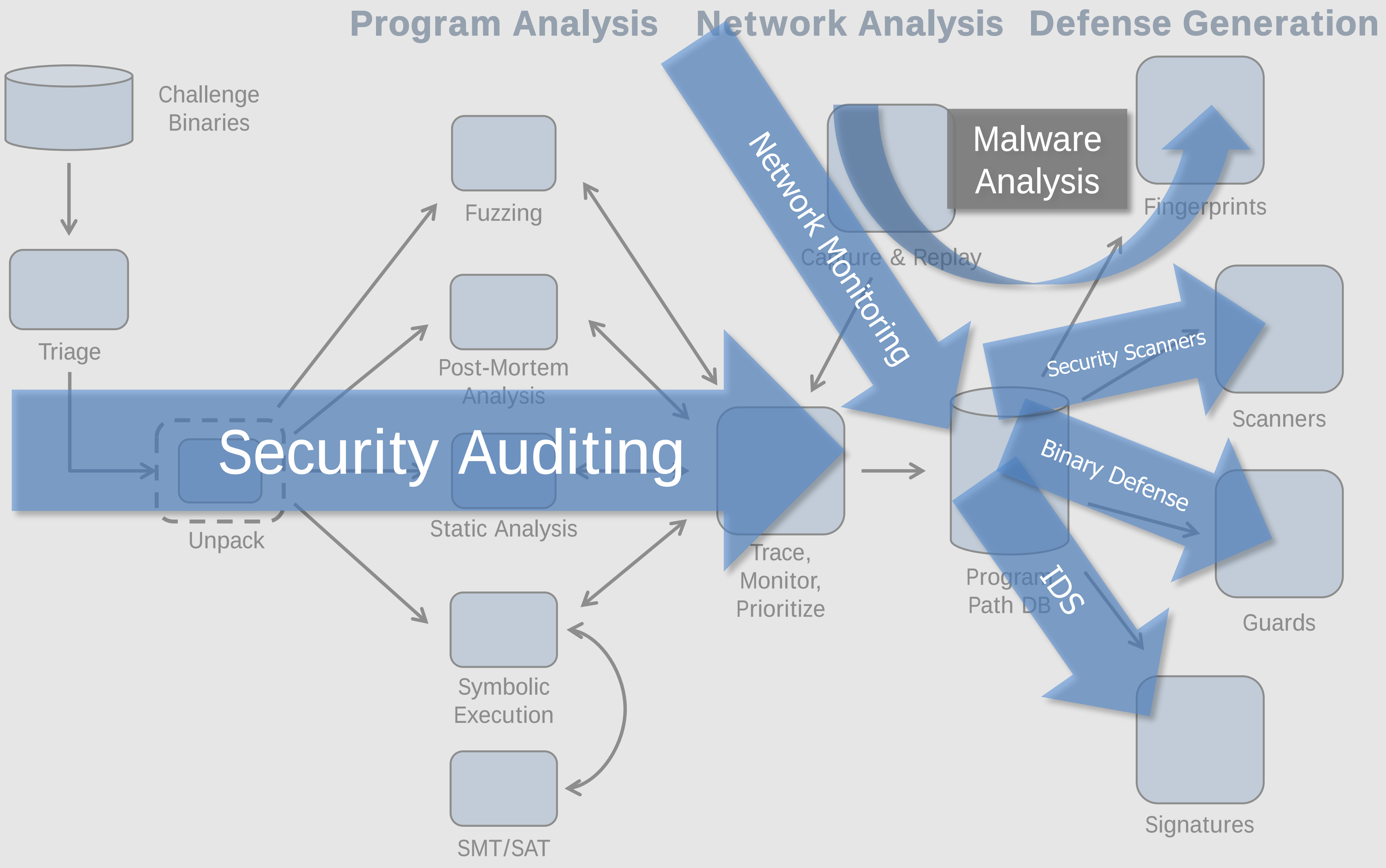


CTF: Human Reasoning Workflow

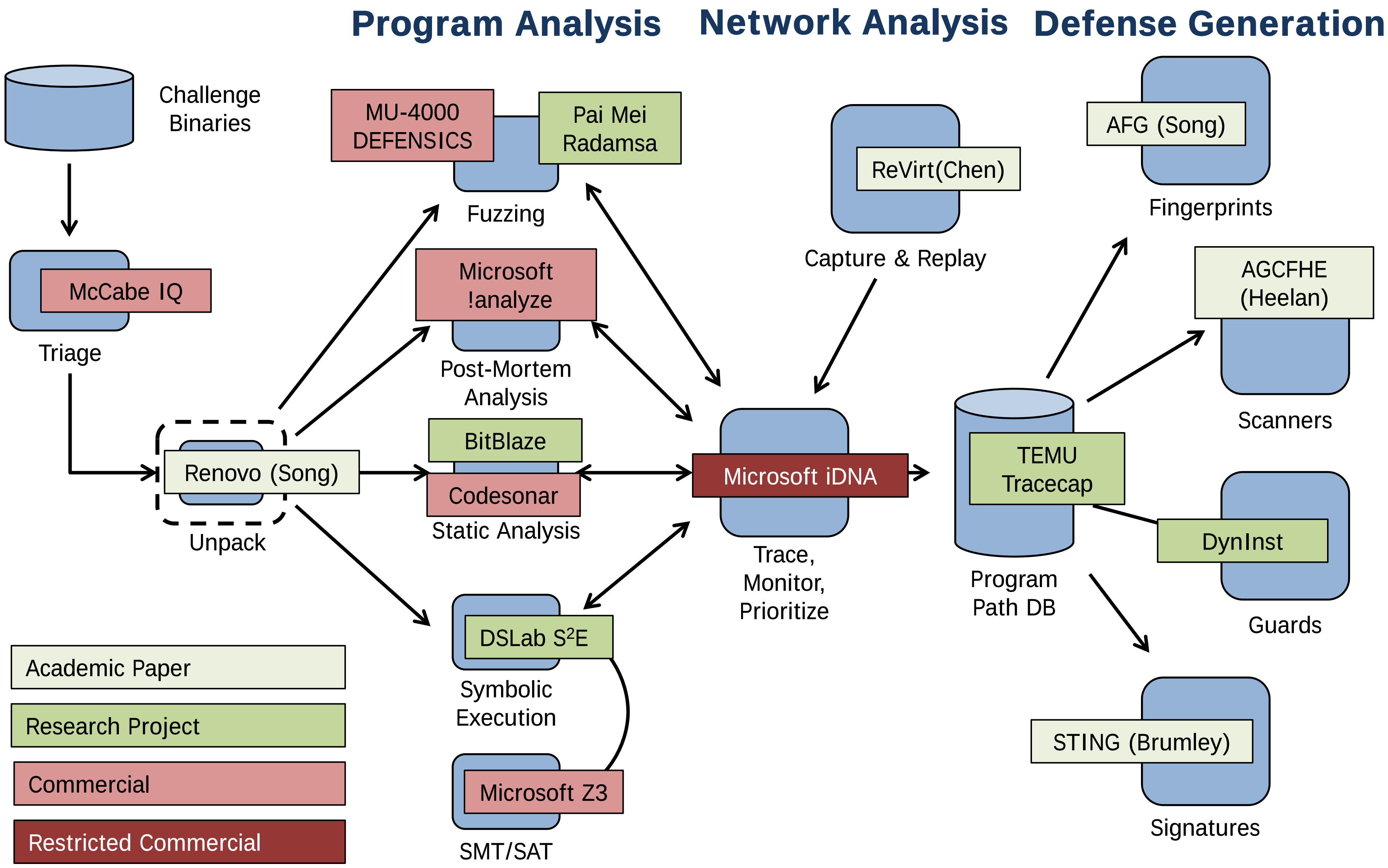


Approved for Public Release, Distribution Unlimited

CTF: Representative Microcosm



CTF in 2014: Seeds of Automation



We've Been Here Before

Chess Grandmasters

Dedicated Systems

World Class CS

© IBM Research

Deep Blue

http://blog.pontiflex.com/2010/05/13/ibm-enters-social-media/

Can We Do It Again?

Cyber Grandmasters

Dedicated Systems

Program Analysis

dailyheadlines.uark.edu

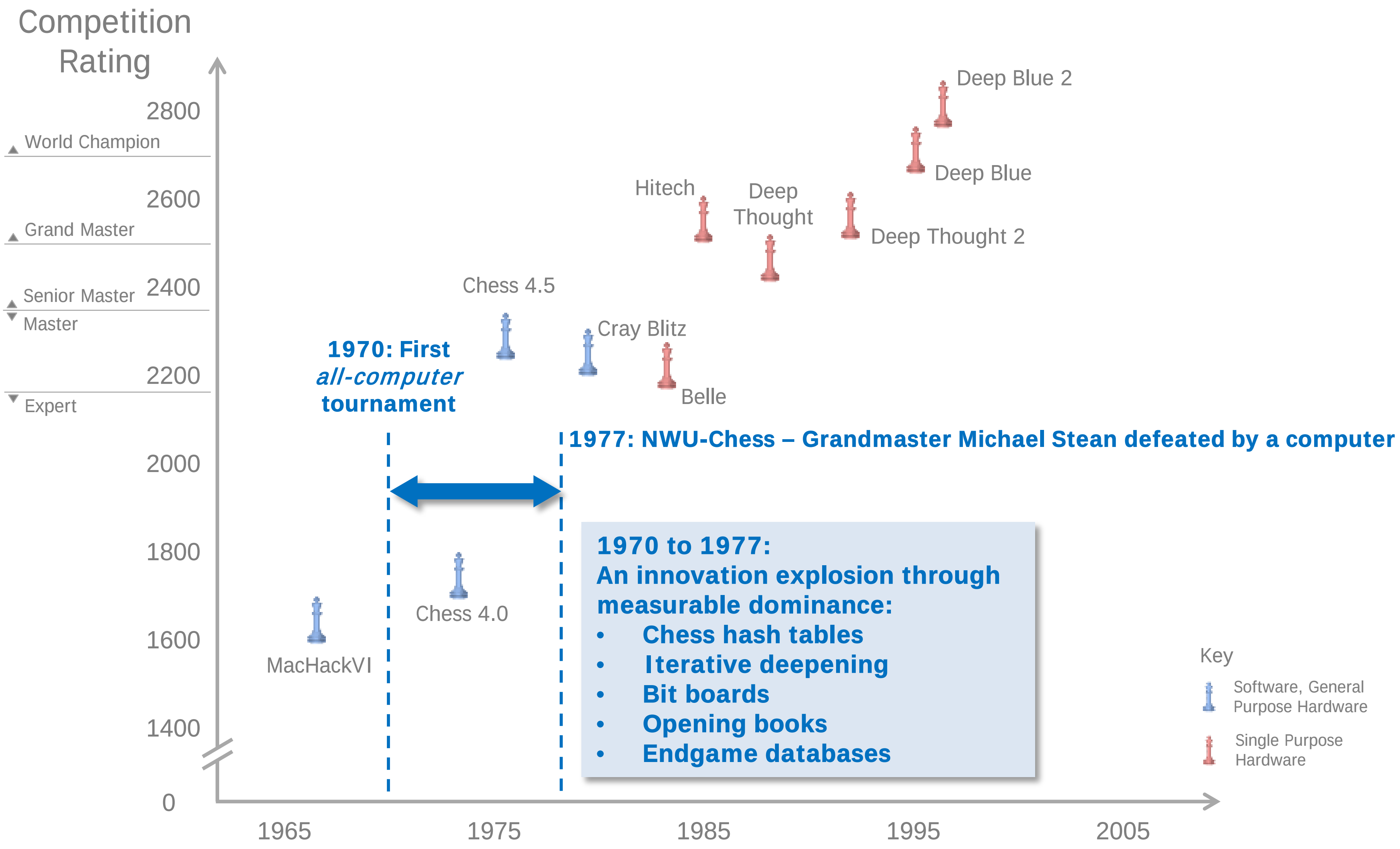
Deep CTF?

http://technorazzi.com/wp-content/uploads/2010/08/ctf_denmark2.jpg

Photo courtesy US Air Force Academy Cyber Competition Club

Approved for Public Release, Distribution Unlimited

A League of Their Own



1970: First *all-computer* tournament

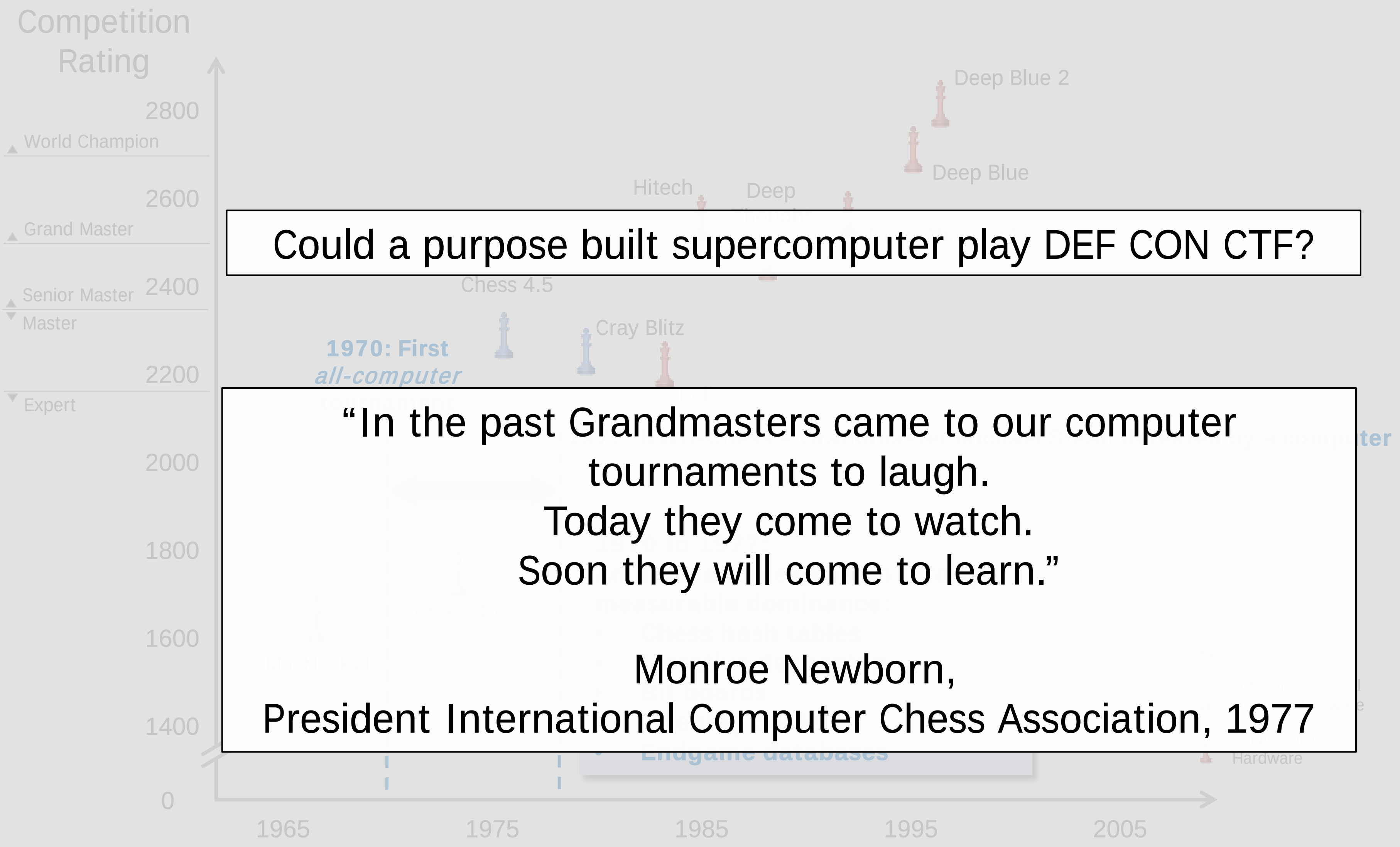
1977: NWU-Chess – Grandmaster Michael Stean defeated by a computer

1970 to 1977:
An innovation explosion through measurable dominance:

- **Chess hash tables**
- **Iterative deepening**
- **Bit boards**
- **Opening books**
- **Endgame databases**

Data Source: Computer History Museum
http://archive.computerhistory.org/resources/still-image/Chess_temporary/still-images/5.1a.%20Chess_Rating_Chart.L062303076.jpg

A League of Their Own



Could a purpose built supercomputer play DEF CON CTF?

“In the past Grandmasters came to our computer tournaments to laugh.
Today they come to watch.
Soon they will come to learn.”

Monroe Newborn,
President International Computer Chess Association, 1977

Data Source: Computer History Museum
http://archive.computerhistory.org/resources/still-image/Chess_temporary/still-images/5.1a.%20Chess_Rating_Chart.L062303076.jpg

A new DARPA Challenge...

Cyber Grand Challenge: Scheduled Participation Opportunities

Open Track

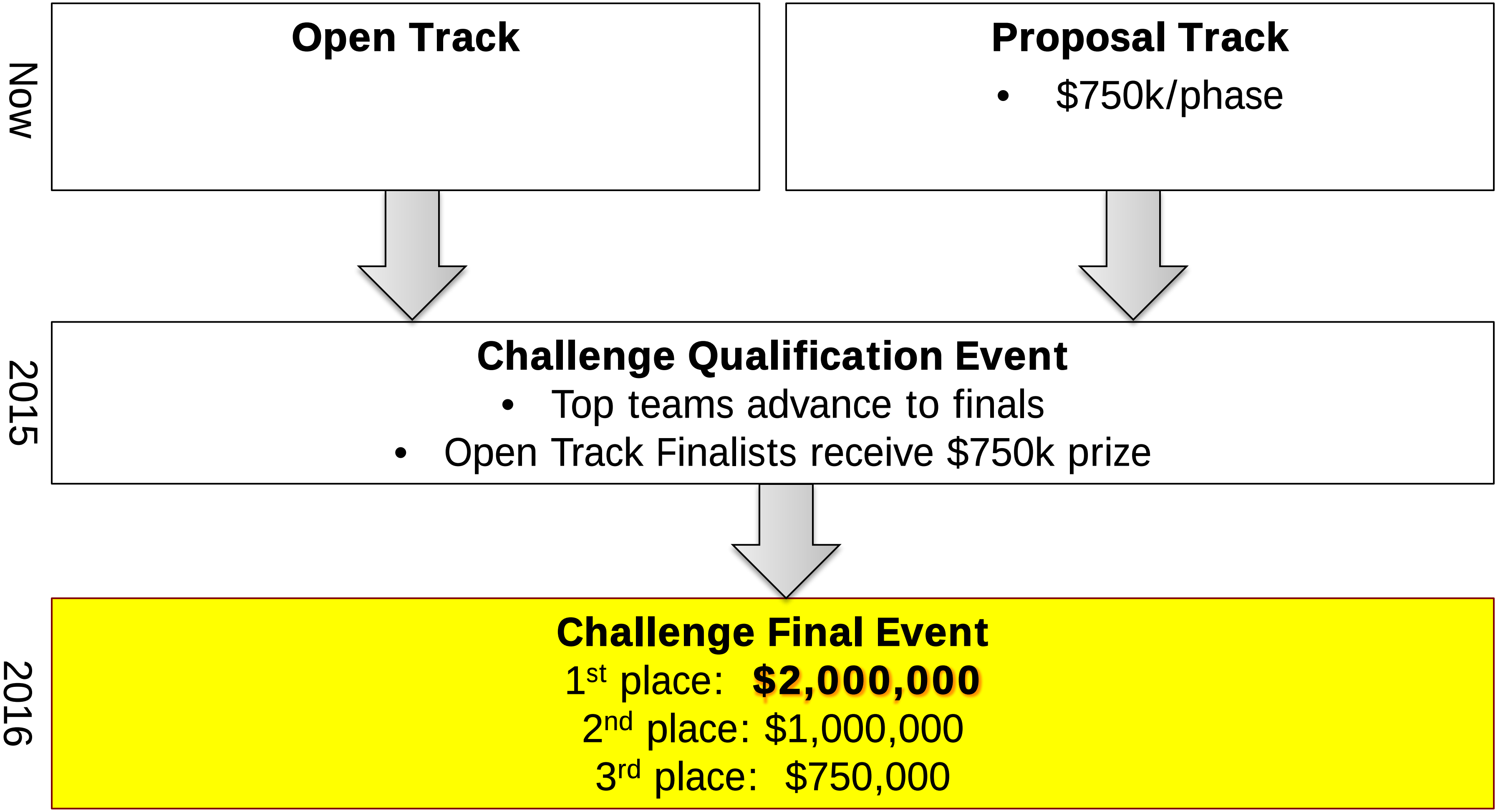
- Open to any eligible team
- No IP restrictions on entrant system

Proposal Track

- DARPA Scientific Review Board
- Funded $750k/phase
- Government Purpose Rights to funded development

See rules at www.darpa.mil/cybergrandchallenge for full details

Cyber Grand Challenge: Scheduled Events



Now

Open Track

Proposal Track

- $750k/phase

2015

Challenge Qualification Event

- Top teams advance to finals
- Open Track Finalists receive $750k prize

2016

Challenge Final Event

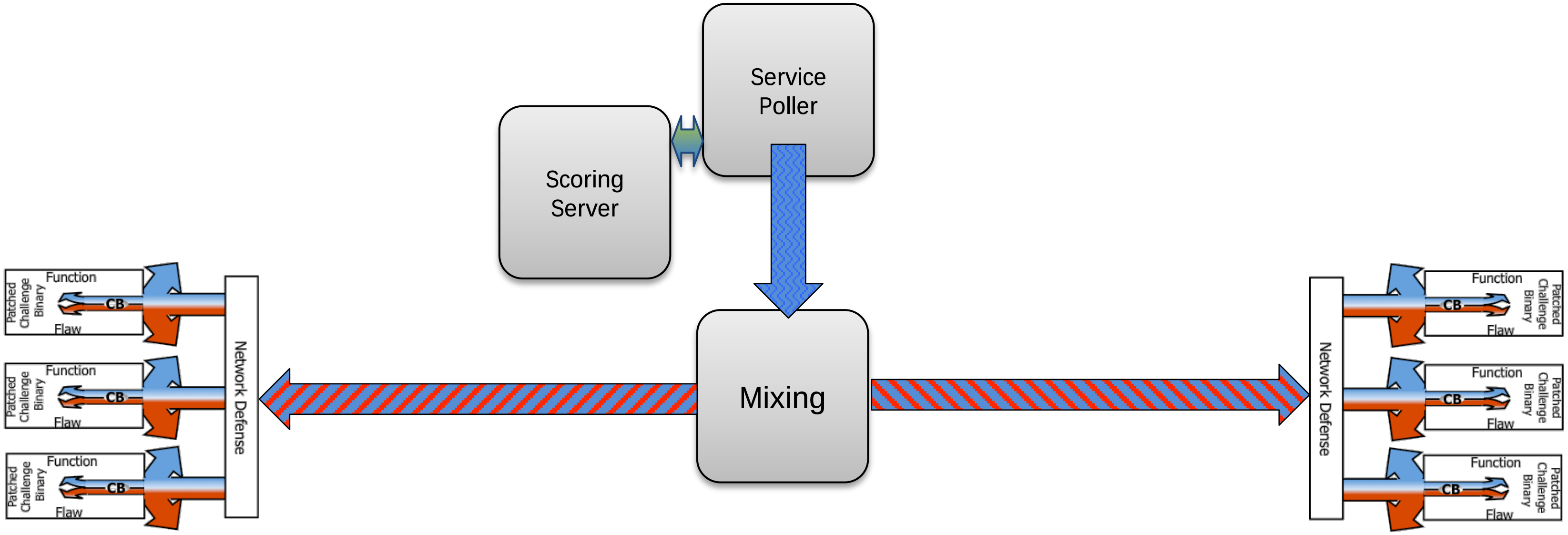
1st place: **$2,000,000**

2nd place: $1,000,000

3rd place: $750,000

For All Secure
GrammaTech
Lekkertech
SIFT
SRI
Trail of Bits
University of California, Berkeley

Consensus Evaluation



Service Poller

Scoring Server

Mixing

Network Defense

Network Defense

Patched Challenge Binary

Function

CB

Flaw

Patched Challenge Binary

Function

CB

Flaw

Patched Challenge Binary

Function

CB

Flaw

Patched Challenge Binary

Function

CB

Flaw

Patched Challenge Binary

Function

CB

Flaw

Patched Challenge Binary

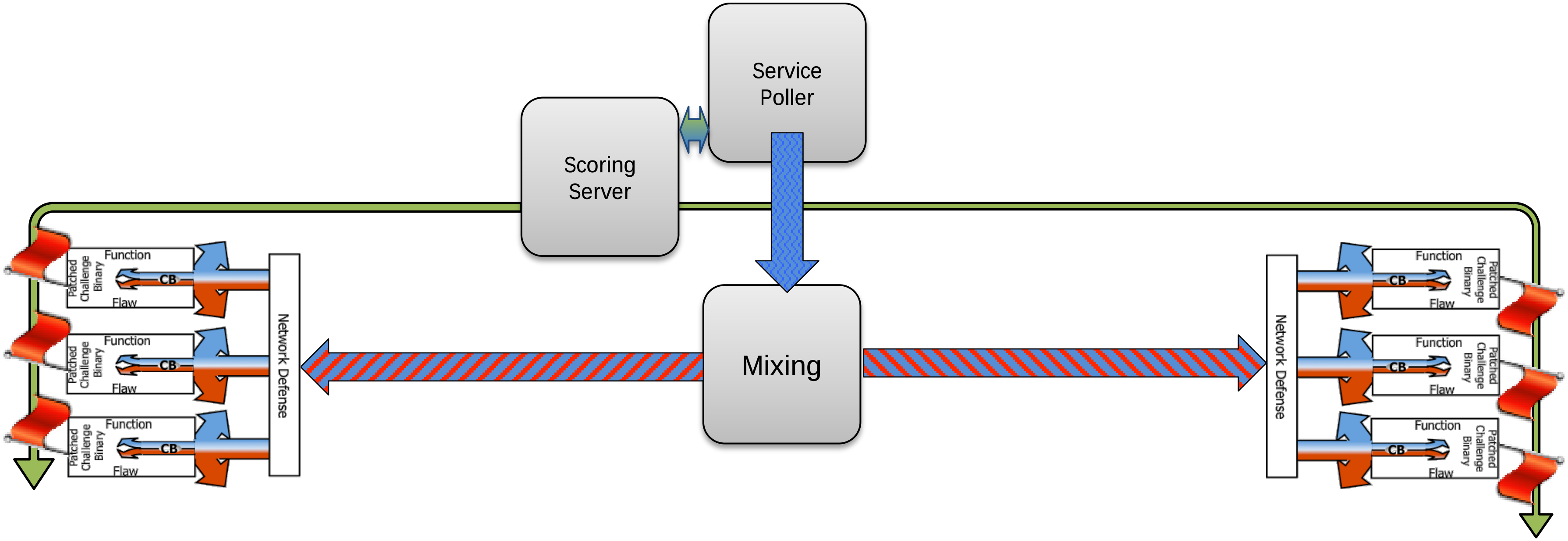
Function

CB

Flaw

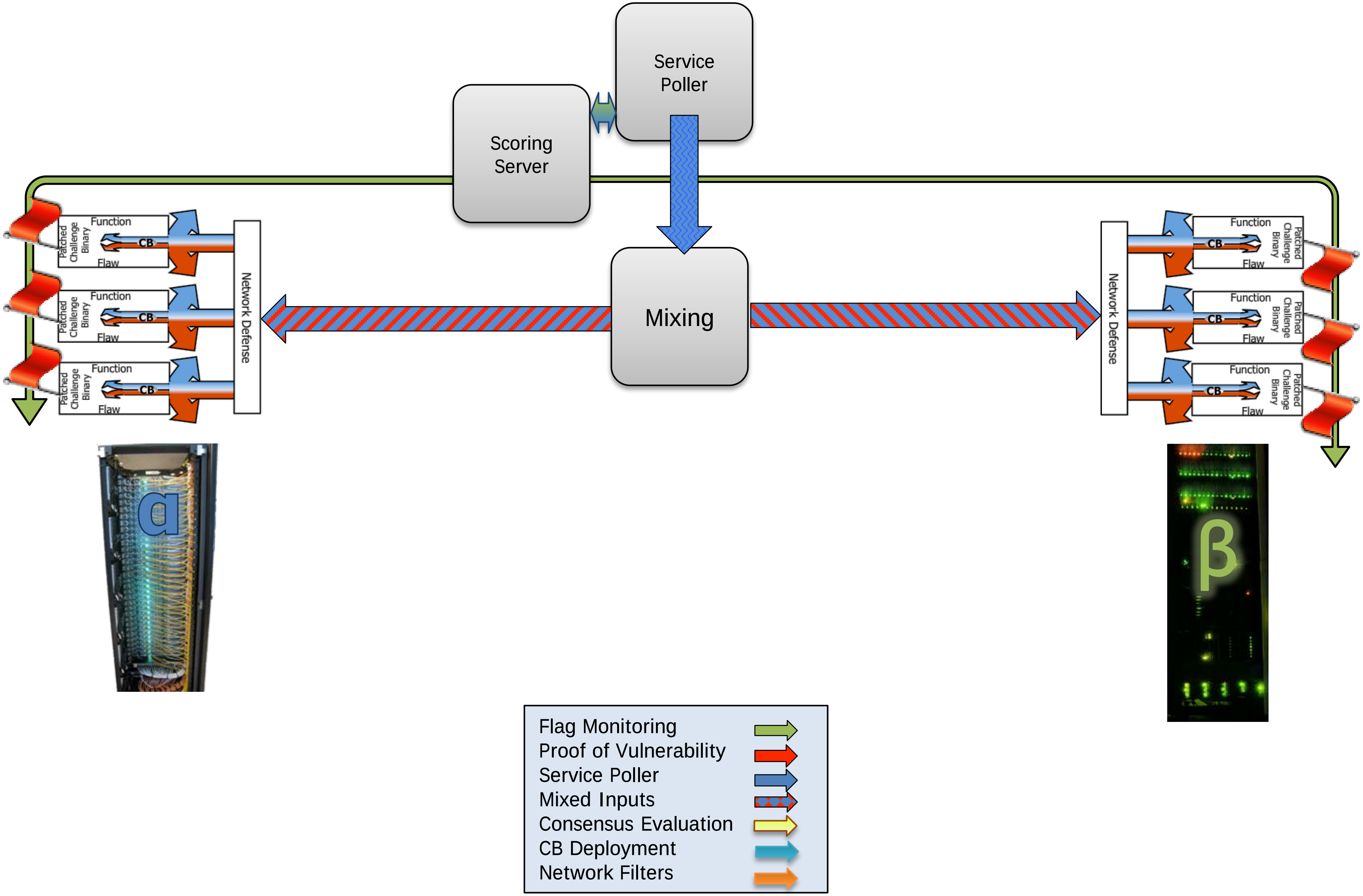
Flag Monitoring ➡
Proof of Vulnerability ➡
Service Poller ➡
Mixed Inputs ➡
Consensus Evaluation ➡
CB Deployment ➡
Network Filters ➡

Consensus Evaluation

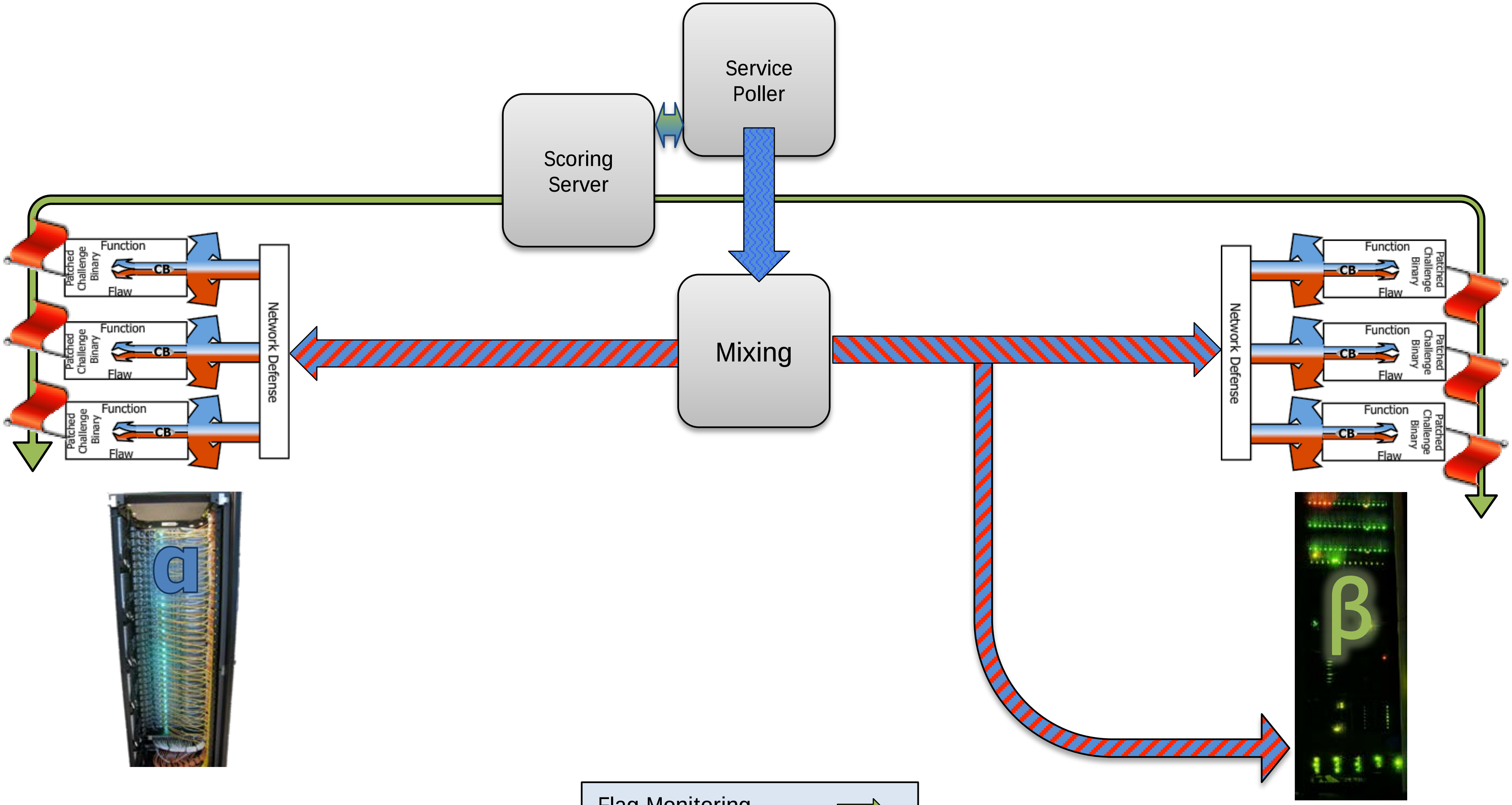


Flag Monitoring 🟩
Proof of Vulnerability 🟥
Service Poller 🟦
Mixed Inputs ▨
Consensus Evaluation 🟨
CB Deployment 🟦
Network Filters 🟧

Consensus Evaluation

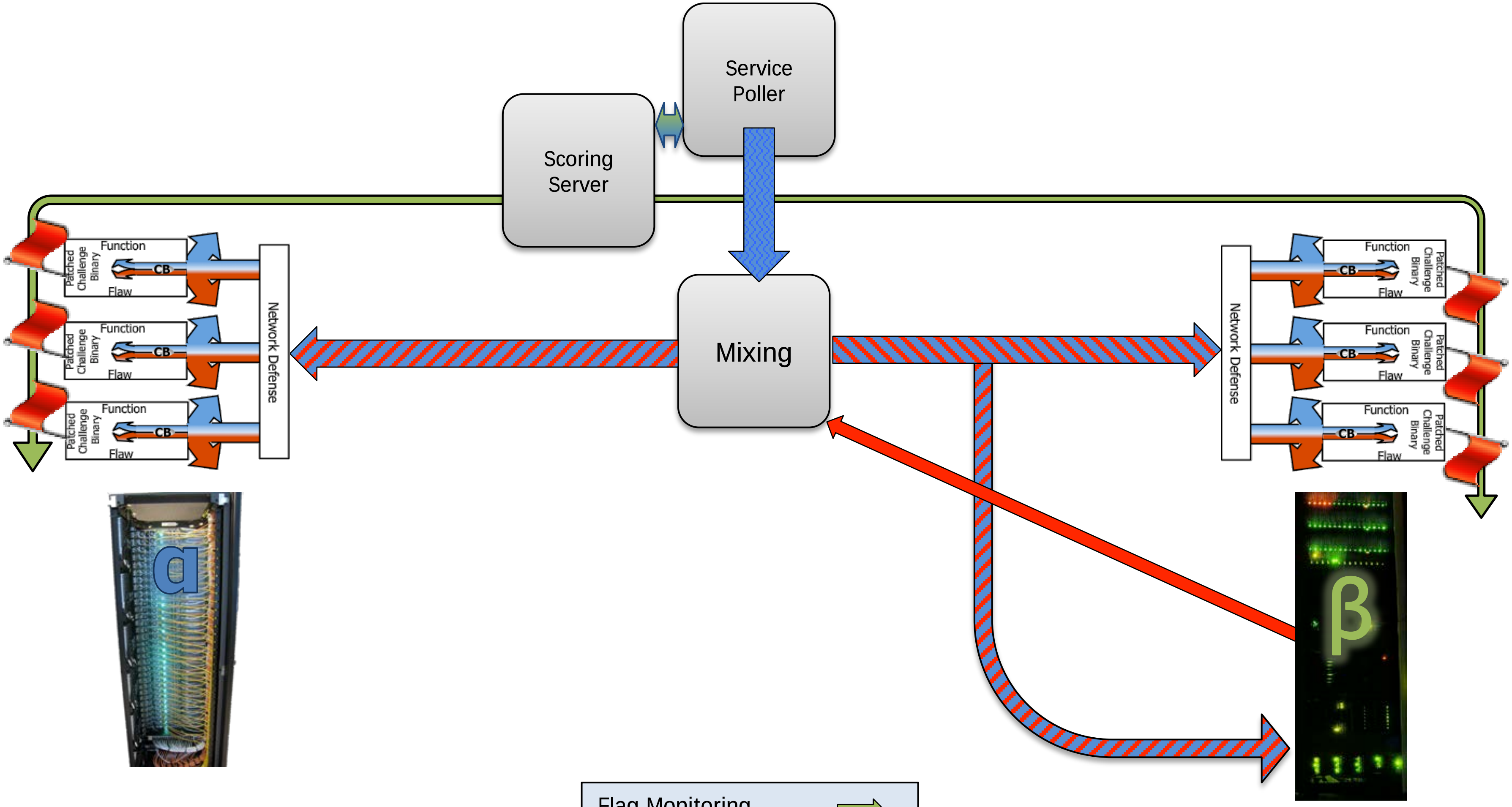


Consensus Evaluation



- Flag Monitoring 🟩
- Proof of Vulnerability 🟥
- Service Poller 🟦
- Mixed Inputs ▨
- Consensus Evaluation 🟨
- CB Deployment 🟦
- Network Filters 🟧

Consensus Evaluation



Service Poller

Scoring Server

Mixing

Network Defense

Network Defense

Function

CB

Flaw

Patched Challenge Binary

Function

CB

Flaw

Patched Challenge Binary

Function

CB

Flaw

Patched Challenge Binary

Function

CB

Flaw

Patched Challenge Binary

Function

CB

Flaw

Patched Challenge Binary

Function

CB

Flaw

Patched Challenge Binary

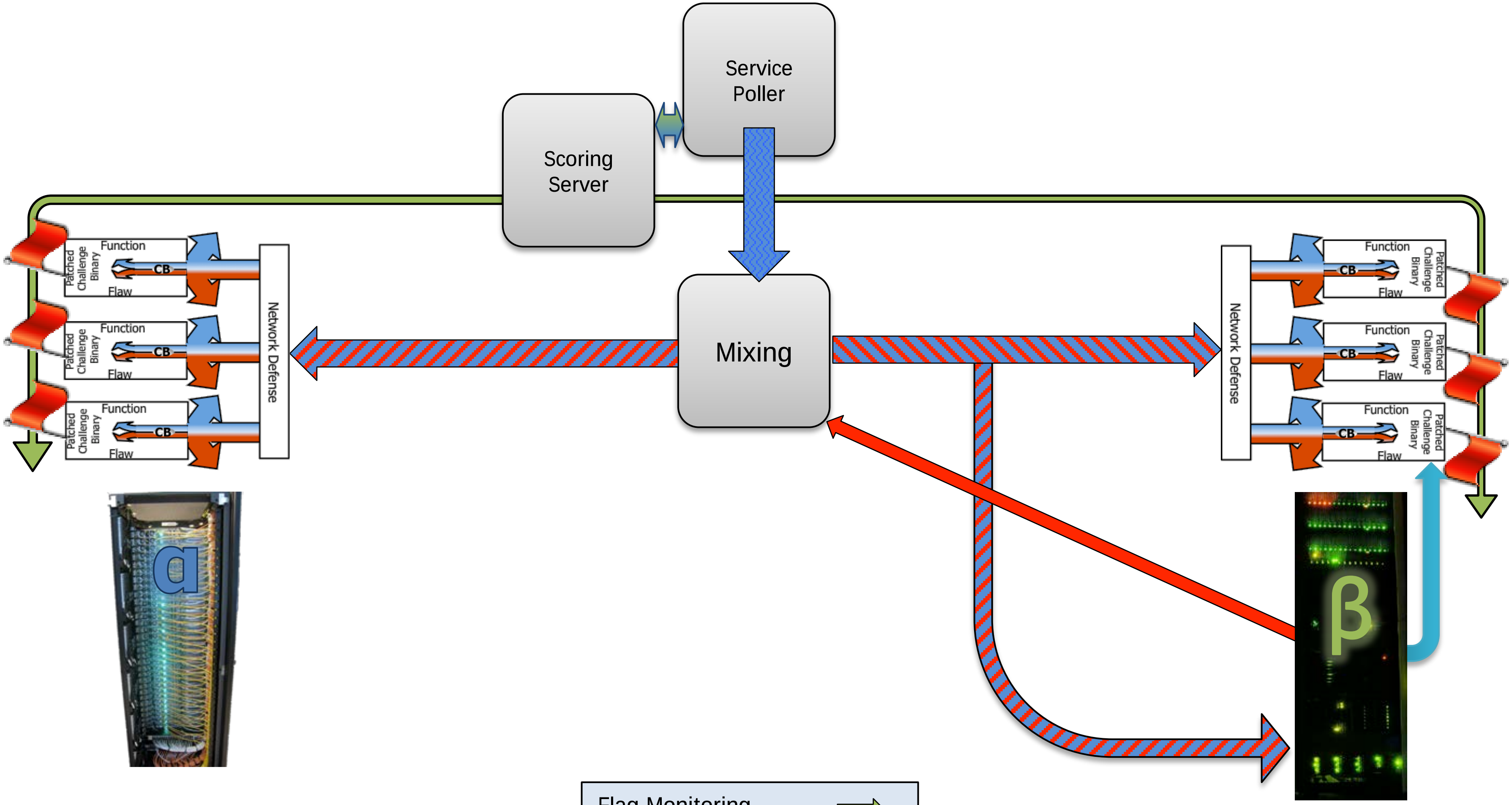
α

β

Flag Monitoring 🟩
Proof of Vulnerability 🟥
Service Poller 🟦
Mixed Inputs ▨
Consensus Evaluation 🟨
CB Deployment 🟦
Network Filters 🟧

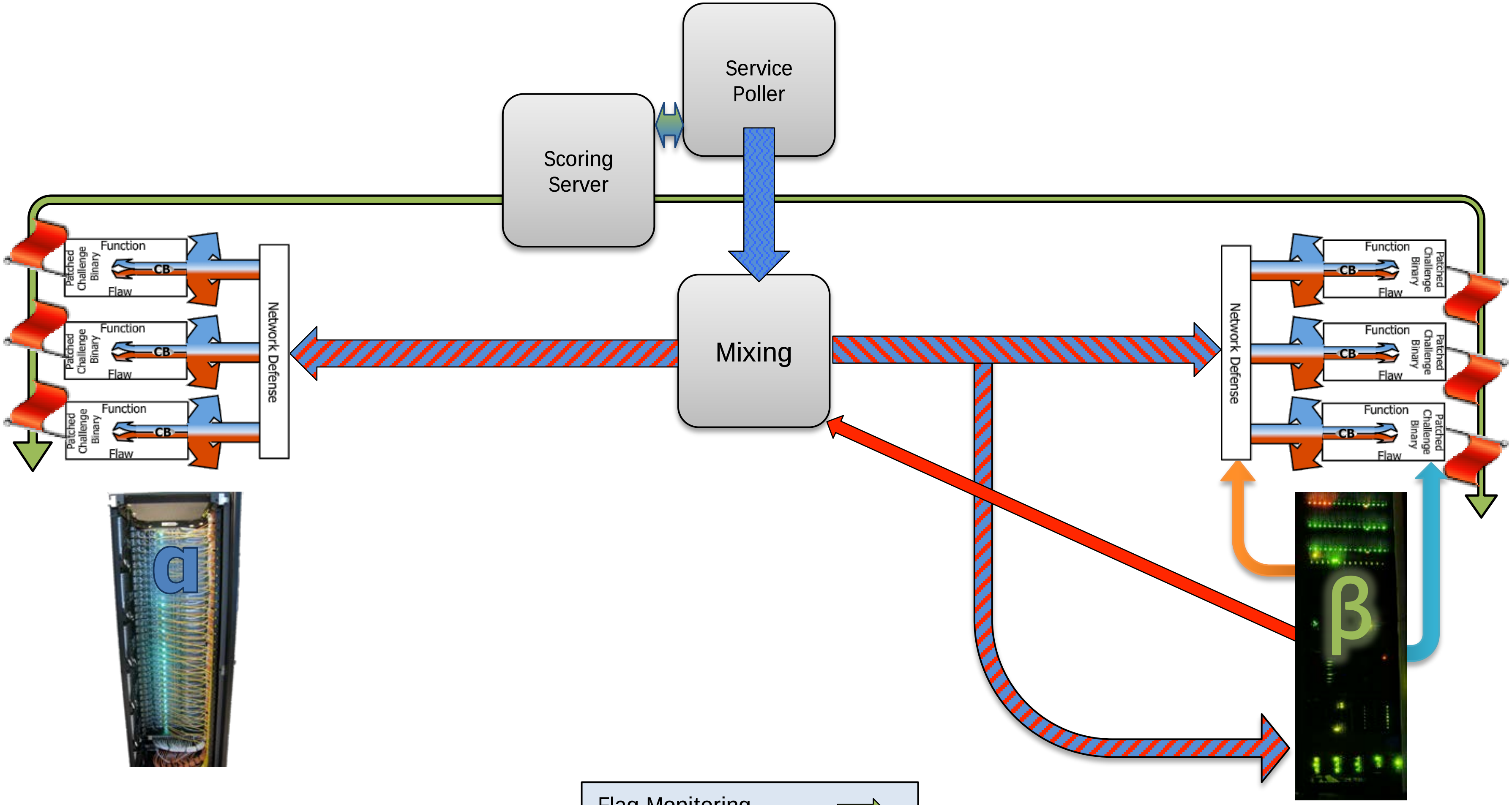
Approved for Public Release, Distribution Unlimited

Consensus Evaluation



Flag Monitoring	🟩
Proof of Vulnerability	🟥
Service Poller	🟦
Mixed Inputs	▨
Consensus Evaluation	🟨
CB Deployment	🟦
Network Filters	🟧

Consensus Evaluation



- Flag Monitoring 🟩
- Proof of Vulnerability 🟥
- Service Poller 🟦
- Mixed Inputs ▨
- Consensus Evaluation 🟨
- CB Deployment 🟦
- Network Filters 🟧

Additional security layers often create vulnerabilities...

Current vulnerability watch list:

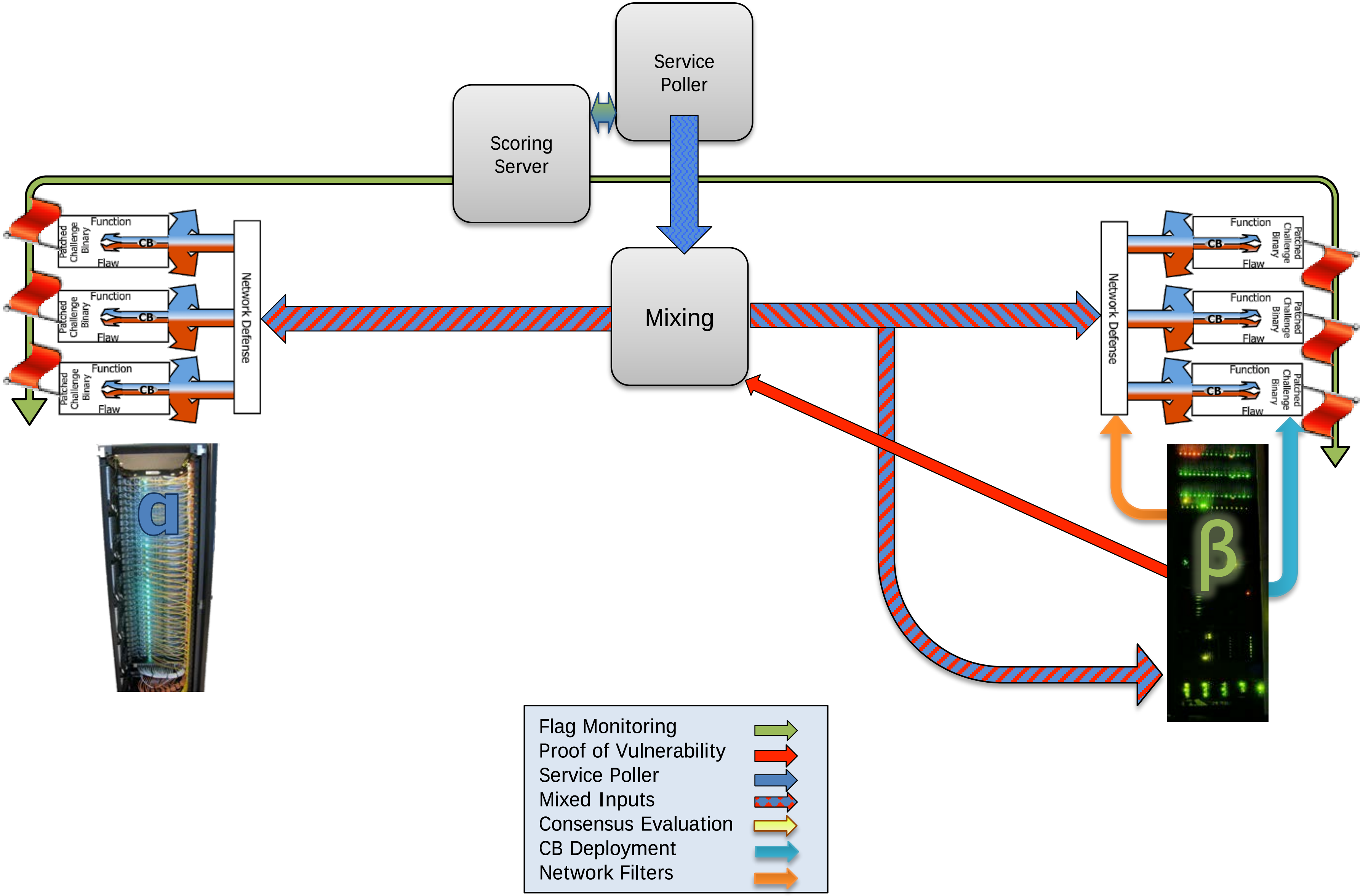
Vulnerability Title	Fix Avail?	Date Added	
XXXXXXXXXXXX XXXXXXXXXXXX Local Privilege Escalation Vulnerability	No	8/25/2010	
XXXXXXXXXXXX XXXXXXXXXXXX Denial of Service Vulnerability	Yes	8/24/2010	⬅
XXXXXXXXXXXX XXXXXXXXXXXX Buffer Overflow Vulnerability	No	8/20/2010	
XXXXXXXXXXXX XXXXXXXXXXXX Sanitization Bypass Weakness	No	8/18/2010	⬅
XXXXXXXXXXXX XXXXXXXXXXXX Security Bypass Vulnerability	No	8/17/2010	⬅
XXXXXXXXXXXX XXXXXXXXXXXX Multiple Security Vulnerabilities	Yes	8/16/2010	⬅
XXXXXXXXXXXX XXXXXXXXXXXX Remote Code Execution Vulnerability	No	8/16/2010	
XXXXXXXXXXXX XXXXXXXXXXXX Use-After-Free Memory Corruption Vulnerability	No	8/12/2010	⬅
XXXXXXXXXXXX XXXXXXXXXXXX Remote Code Execution Vulnerability	No	8/10/2010	
XXXXXXXXXXXX XXXXXXXXXXXX Multiple Buffer Overflow Vulnerabilities	No	8	
XXXXXXXXXXXX XXXXXXXXXXXX Stack Buffer Overflow Vulnerability	Yes	8	
XXXXXXXXXXXX XXXXXXXXXXXX Security-Bypass Vulnerability	No	8	
XXXXXXXXXXXX XXXXXXXXXXXX Multiple Security Vulnerabilities	No	8	
XXXXXXXXXXXX XXXXXXXXXXXX Buffer Overflow Vulnerability	No	7/29/2010	
XXXXXXXXXXXX XXXXXXXXXXXX Remote Privilege Escalation Vulnerability	No	7/28/2010	⬅
XXXXXXXXXXXX XXXXXXXXXXXX Cross Site Request Forgery Vulnerability	No	7/26/2010	
XXXXXXXXXXXX XXXXXXXXXXXX Multiple Denial Of Service Vulnerabilities	No	7/22/2010	

6 of the vulnerabilities are in security software

Color Code Key:

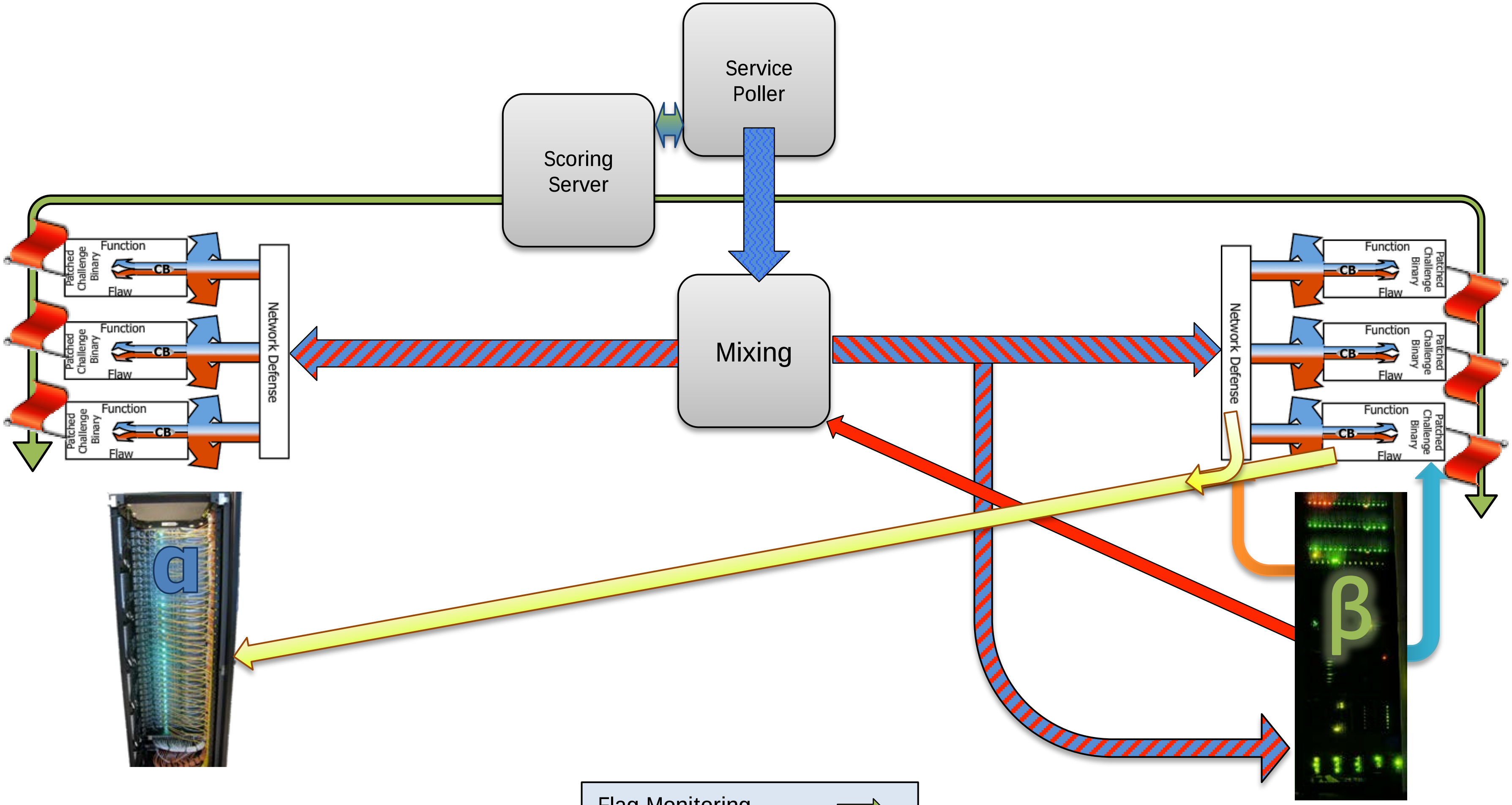
🟩 Vendor Replied – Fix in development	🟦 Awaiting Vendor Reply/Confirmation	🟧 Awaiting CC/S/A use validation

Consensus Evaluation



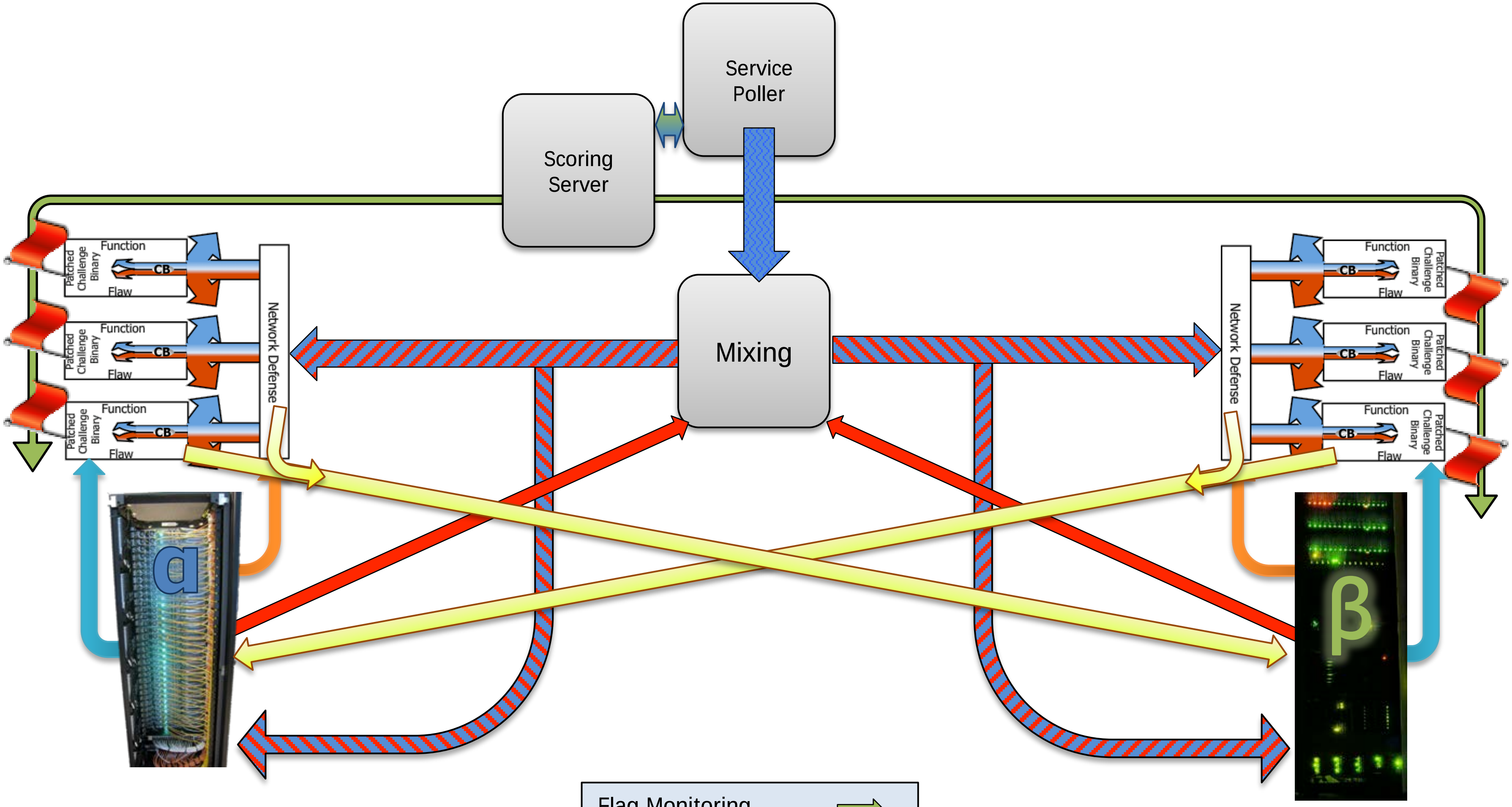
- Flag Monitoring 🟩
- Proof of Vulnerability 🟥
- Service Poller 🟦
- Mixed Inputs ▨
- Consensus Evaluation 🟨
- CB Deployment 🟦
- Network Filters 🟧

Consensus Evaluation



Flag Monitoring 🟩
Proof of Vulnerability 🟥
Service Poller 🟦
Mixed Inputs ▨
Consensus Evaluation 🟨
CB Deployment 🟦
Network Filters 🟧

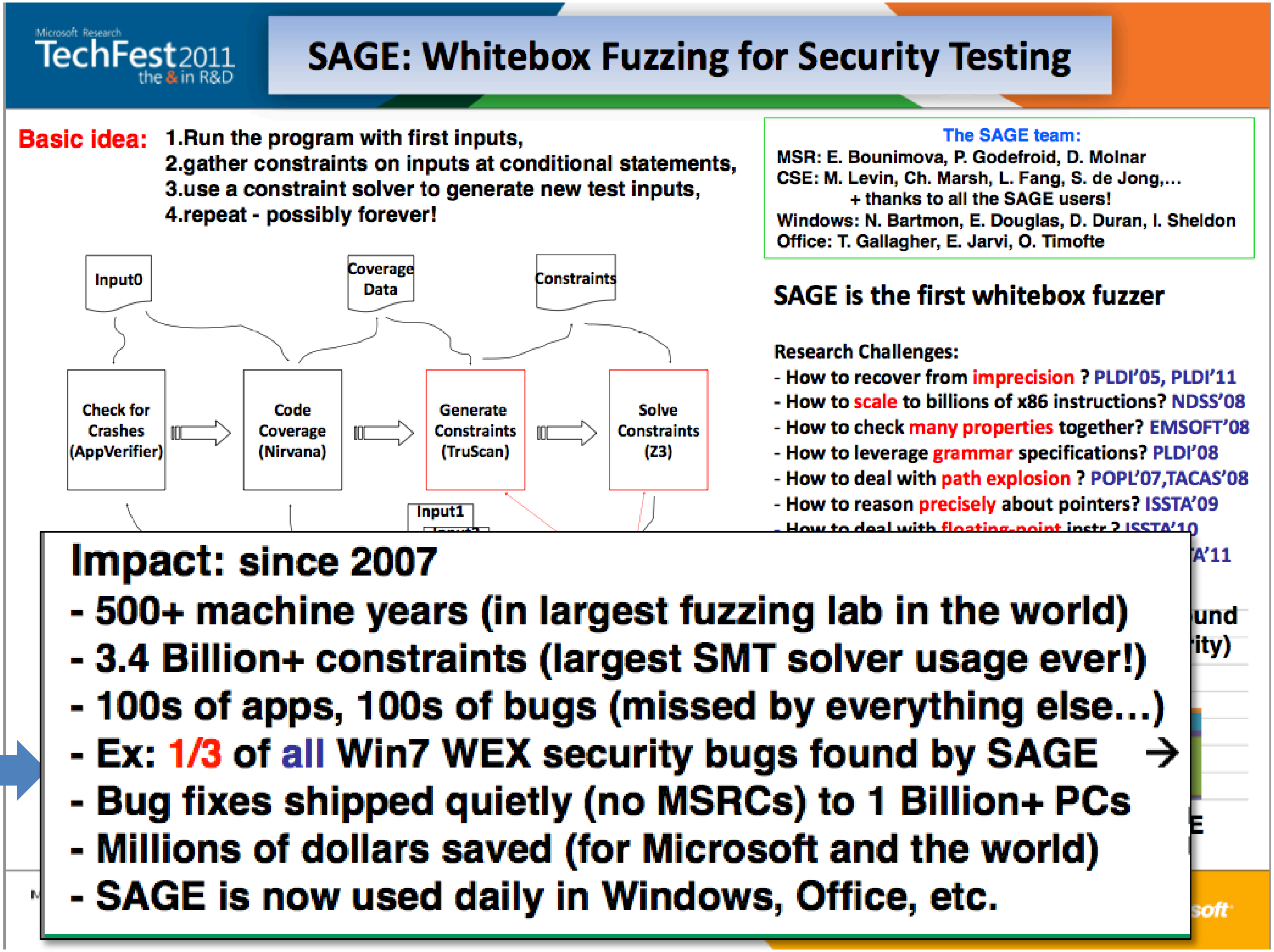
Consensus Evaluation



- Flag Monitoring
- Proof of Vulnerability
- Service Poller
- Mixed Inputs
- Consensus Evaluation
- CB Deployment
- Network Filters

At Microsoft, a Precursor

Machine Reasoning now accounts for many security flaws removed from Windows systems.



Microsoft Research TechFest2011 the & in R&D

SAGE: Whitebox Fuzzing for Security Testing

Basic idea: 1.Run the program with first inputs,
2.gather constraints on inputs at conditional statements,
3.use a constraint solver to generate new test inputs,
4.repeat - possibly forever!

Input0
Coverage Data
Constraints
Check for Crashes (AppVerifier)
Code Coverage (Nirvana)
Generate Constraints (TruScan)
Solve Constraints (Z3)
Input1

The SAGE team:
MSR: E. Bounimova, P. Godefroid, D. Molnar
CSE: M. Levin, Ch. Marsh, L. Fang, S. de Jong,...
+ thanks to all the SAGE users!
Windows: N. Bartmon, E. Douglas, D. Duran, I. Sheldon
Office: T. Gallagher, E. Jarvi, O. Timofte

SAGE is the first whitebox fuzzer

Research Challenges:
- How to recover from imprecision ? PLDI'05, PLDI'11
- How to scale to billions of x86 instructions? NDSS'08
- How to check many properties together? EMSOFT'08
- How to leverage grammar specifications? PLDI'08
- How to deal with path explosion ? POPL'07,TACAS'08
- How to reason precisely about pointers? ISSTA'09
- How to deal with floating-point instr ? ISSTA'10

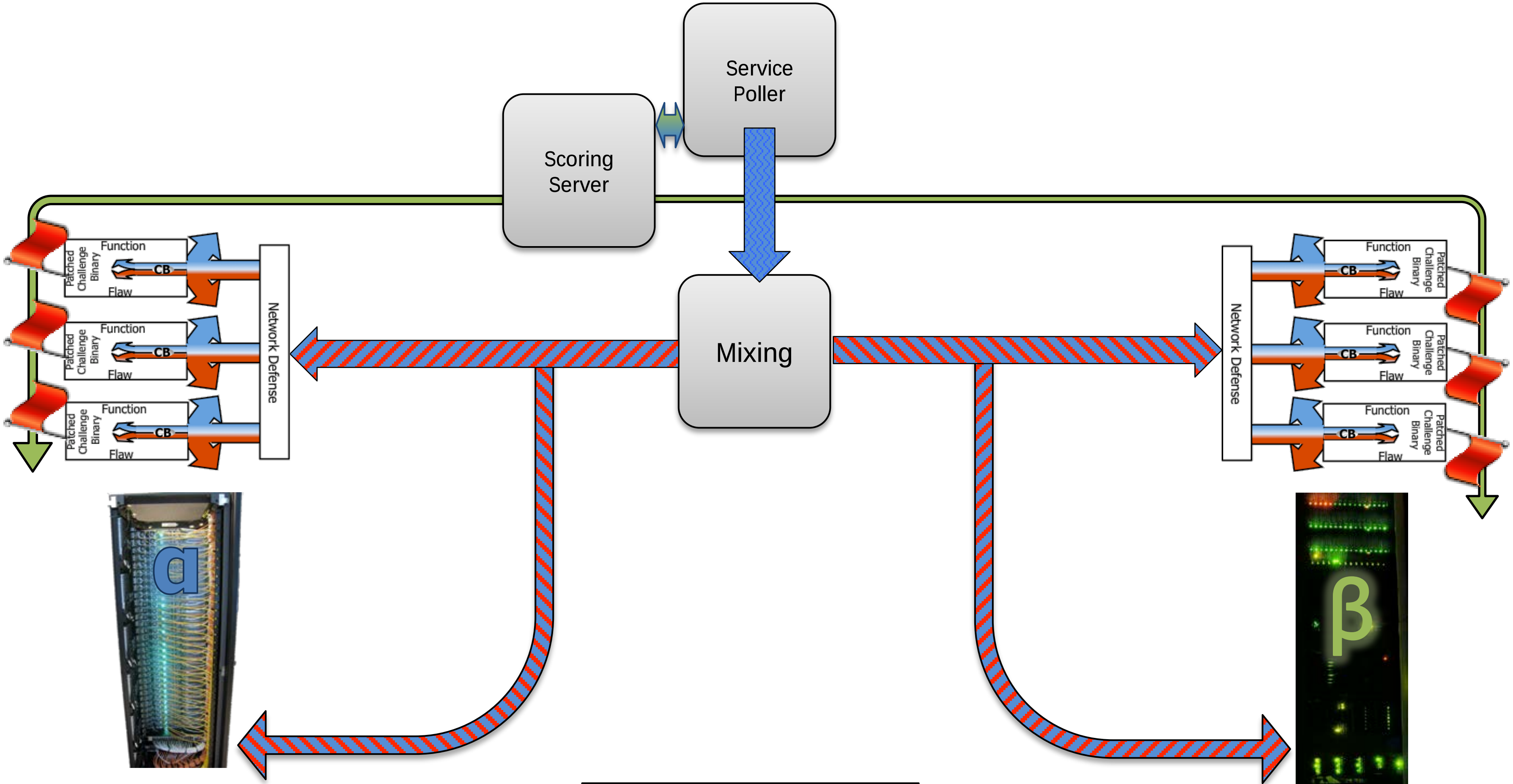
A'11

und

ity)

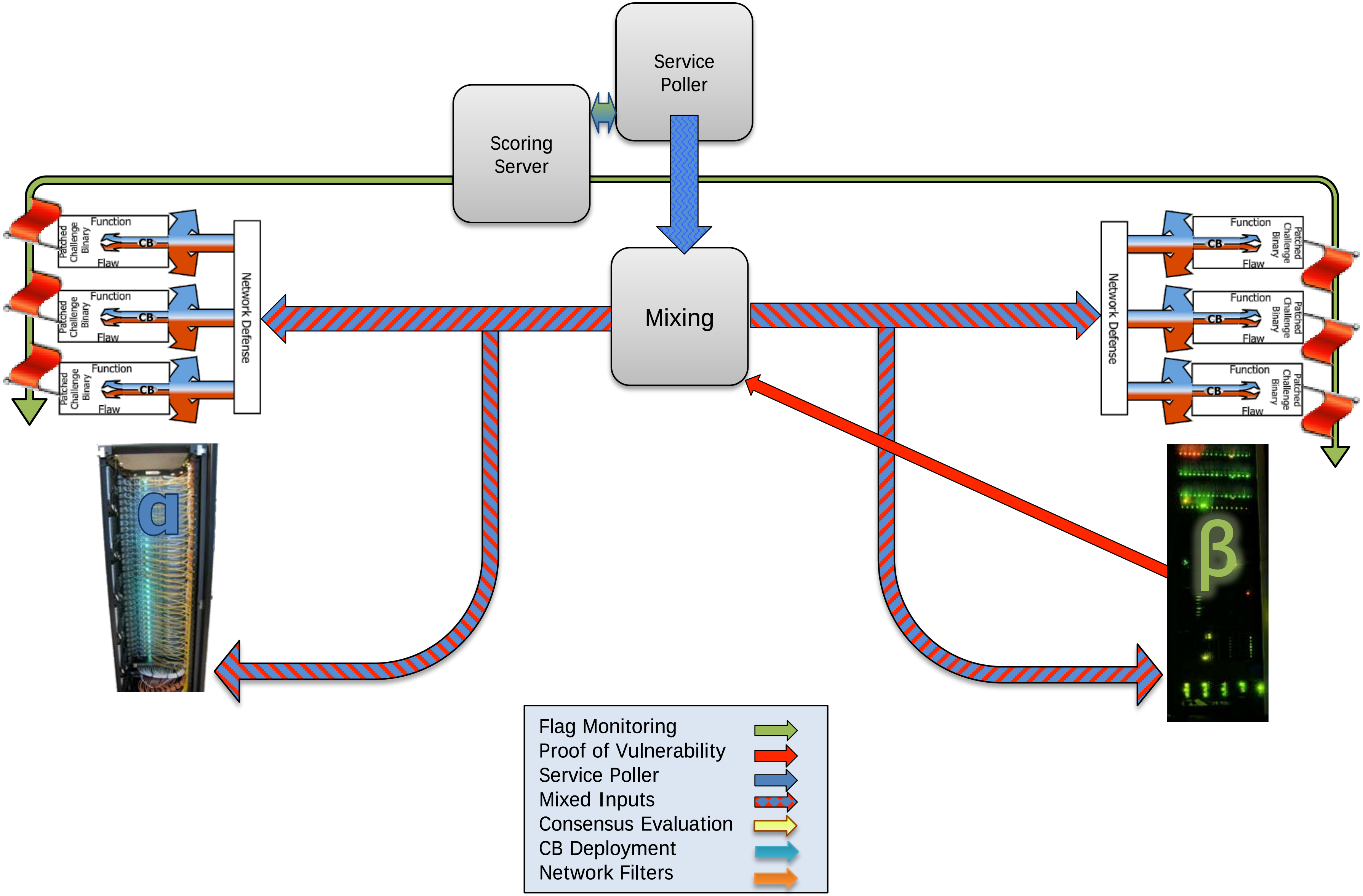
Impact: since 2007
- 500+ machine years (in largest fuzzing lab in the world)
- 3.4 Billion+ constraints (largest SMT solver usage ever!)
- 100s of apps, 100s of bugs (missed by everything else…)
- Ex: 1/3 of all Win7 WEX security bugs found by SAGE →
- Bug fixes shipped quietly (no MSRCs) to 1 Billion+ PCs
- Millions of dollars saved (for Microsoft and the world)
- SAGE is now used daily in Windows, Office, etc.

Consensus Evaluation



- Flag Monitoring
- Proof of Vulnerability
- Service Poller
- Mixed Inputs
- Consensus Evaluation
- CB Deployment
- Network Filters

Consensus Evaluation



Service Poller

Scoring Server

Mixing

Network Defense

Network Defense

Patched Challenge Binary

Function

CB

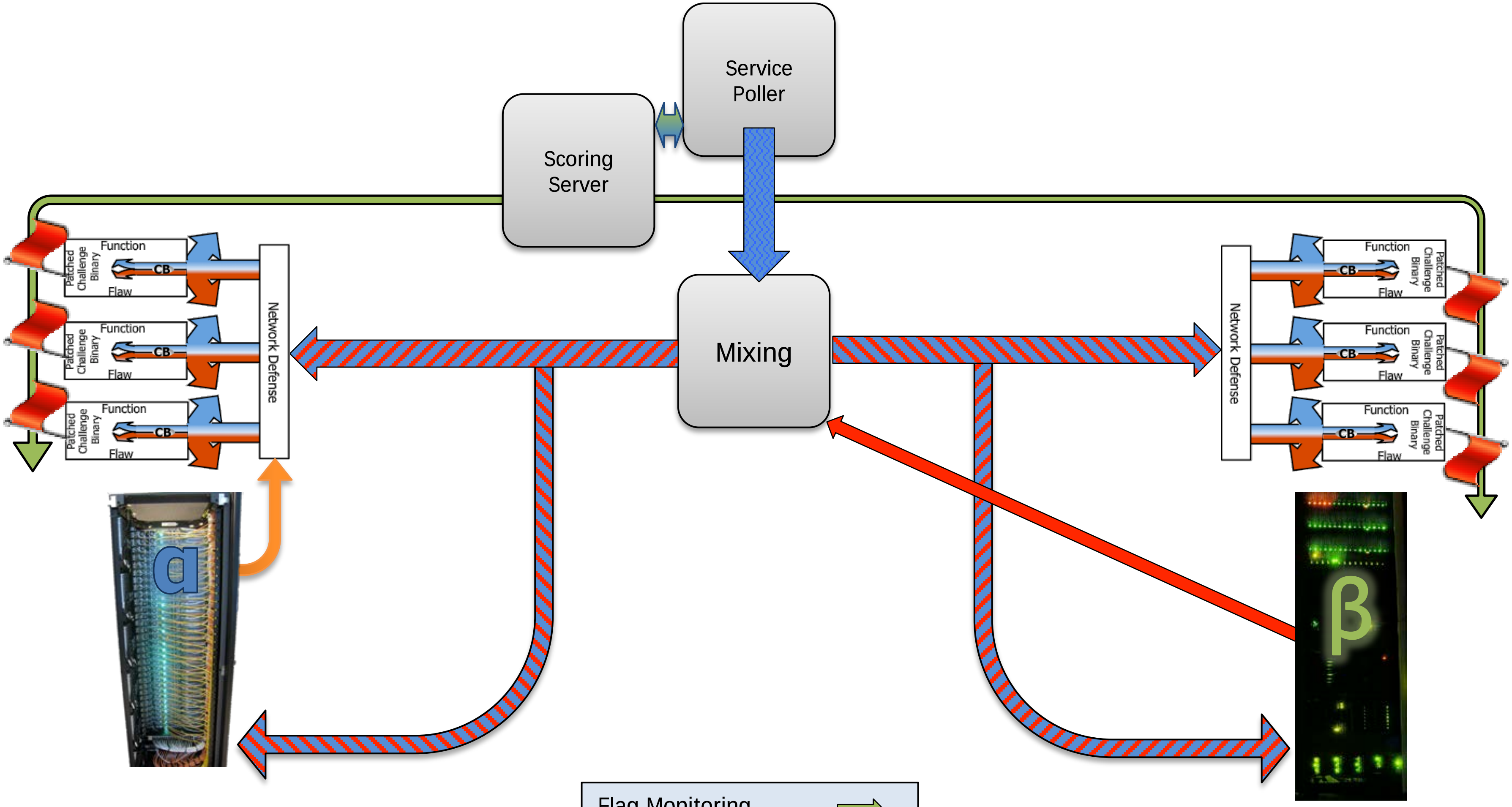
Flaw

α

β

Flag Monitoring 🟩
Proof of Vulnerability 🟥
Service Poller 🟦
Mixed Inputs ▨
Consensus Evaluation 🟨
CB Deployment 🟦
Network Filters 🟧

Consensus Evaluation



Service Poller

Scoring Server

Mixing

Network Defense

Function

CB

Flaw

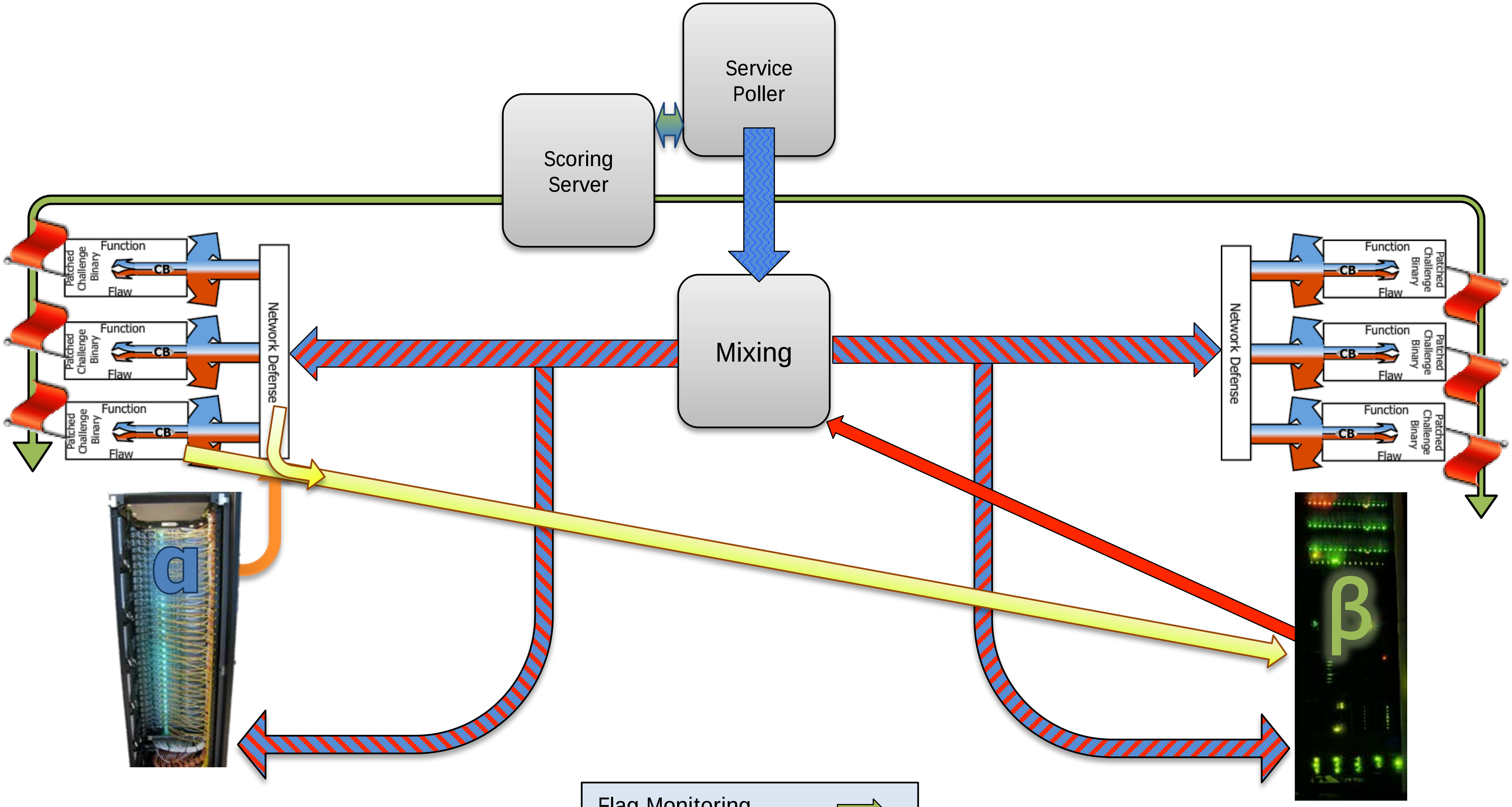
Patched Challenge Binary

α

β

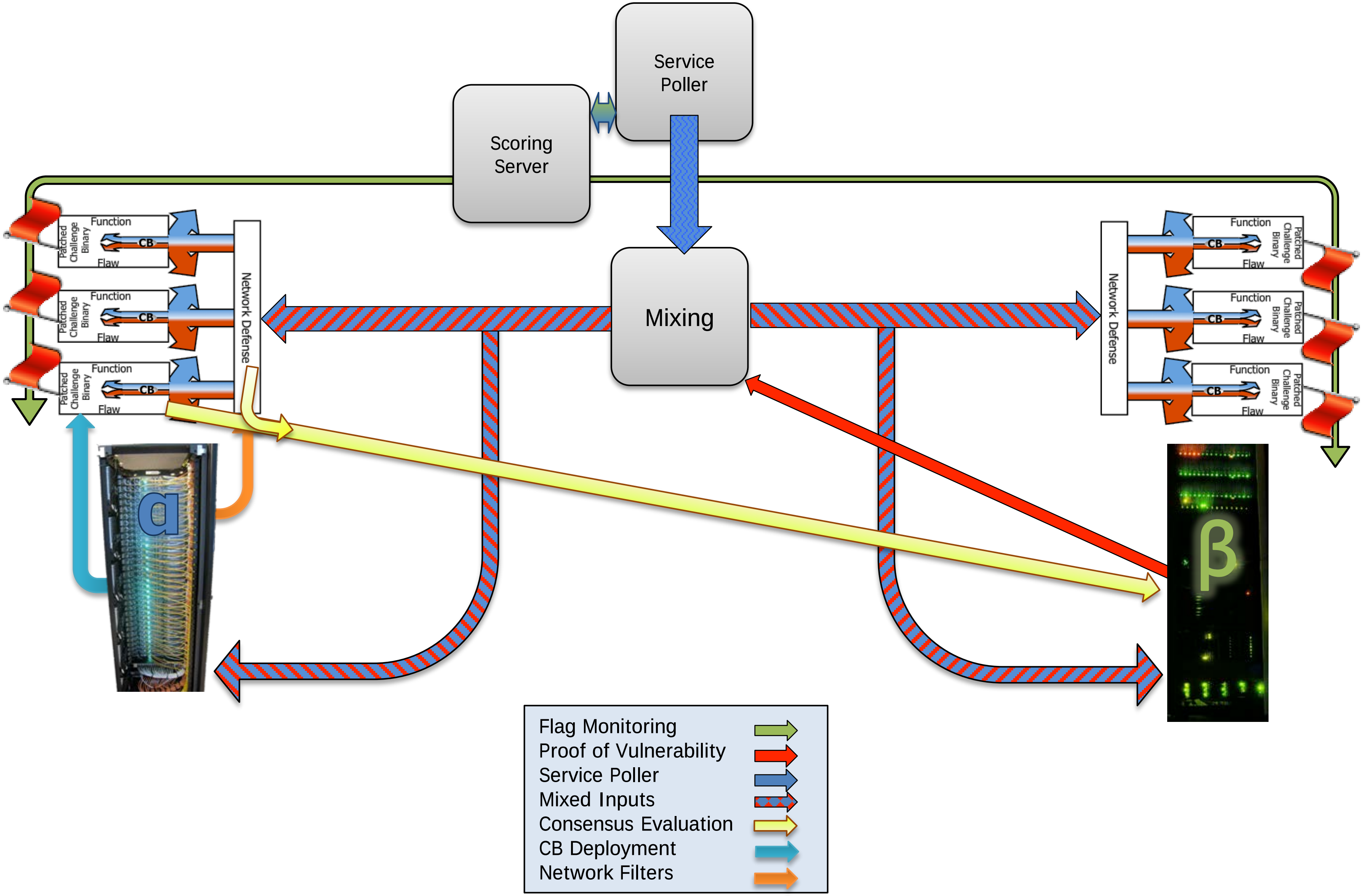
Flag Monitoring
Proof of Vulnerability
Service Poller
Mixed Inputs
Consensus Evaluation
CB Deployment
Network Filters

Consensus Evaluation

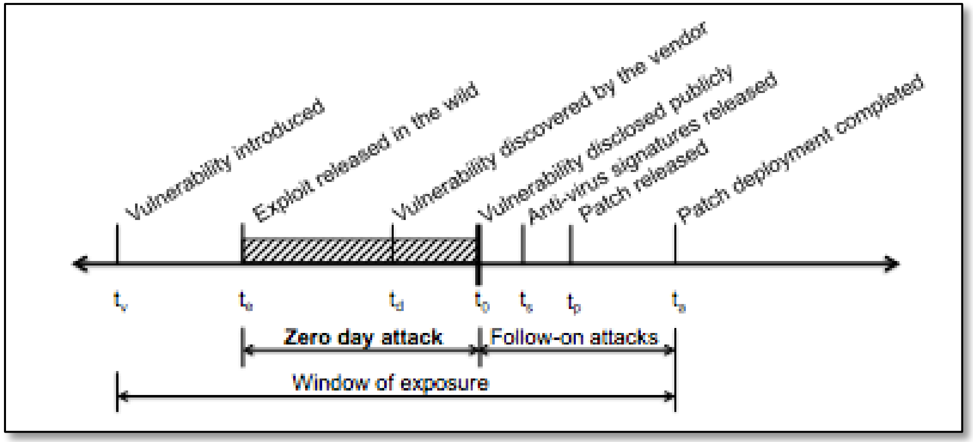


Flag Monitoring
Proof of Vulnerability
Service Poller
Mixed Inputs
Consensus Evaluation
CB Deployment
Network Filters

Consensus Evaluation



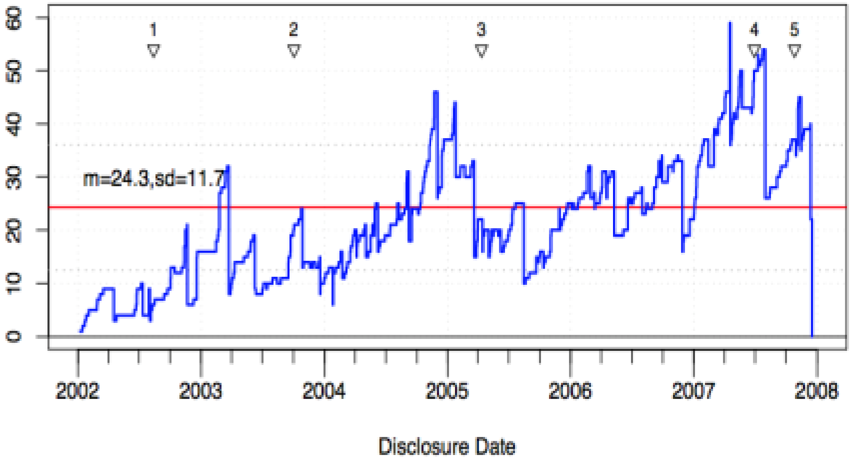
Defensive Adaptation Speed



Vulnerability introduced
Exploit released in the wild
Vulnerability discovered by the vendor
Vulnerability disclosed publicly
Anti-virus signatures released
Patch released
Patch deployment completed
t_v
t_e
t_d
t_0
t_s
t_p
t_a
Zero day attack
Follow-on attacks
Window of exposure

*

"a typical zero-day attack lasts 312 days" *



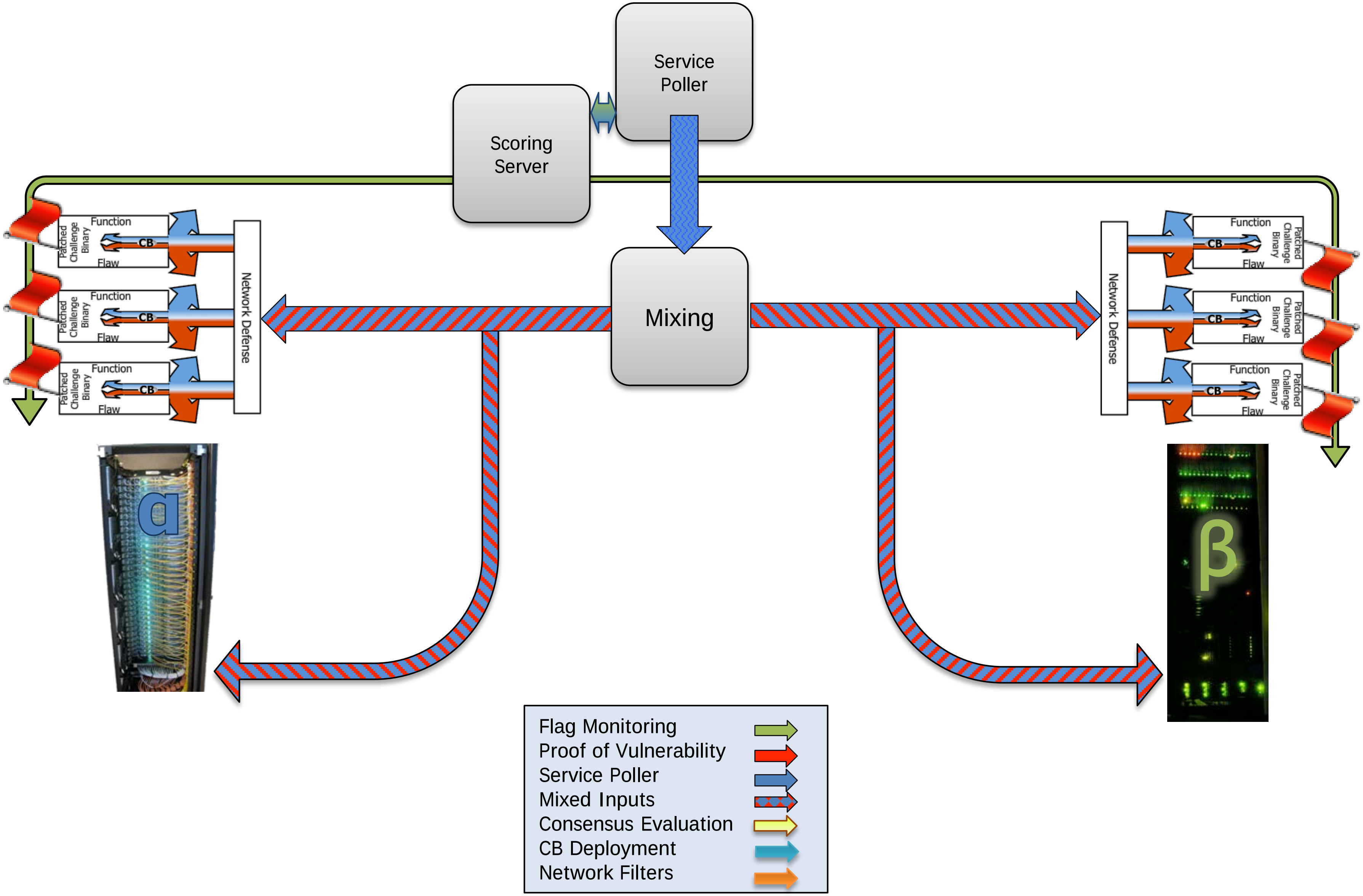
**

...and takes 24 days to patch.

** Security Econometrics, Stefan Frei

* Before We Knew It Bilge,Dumitras

Consensus Evaluation



Service Poller

Scoring Server

Mixing

Network Defense

Network Defense

Function

CB

Flaw

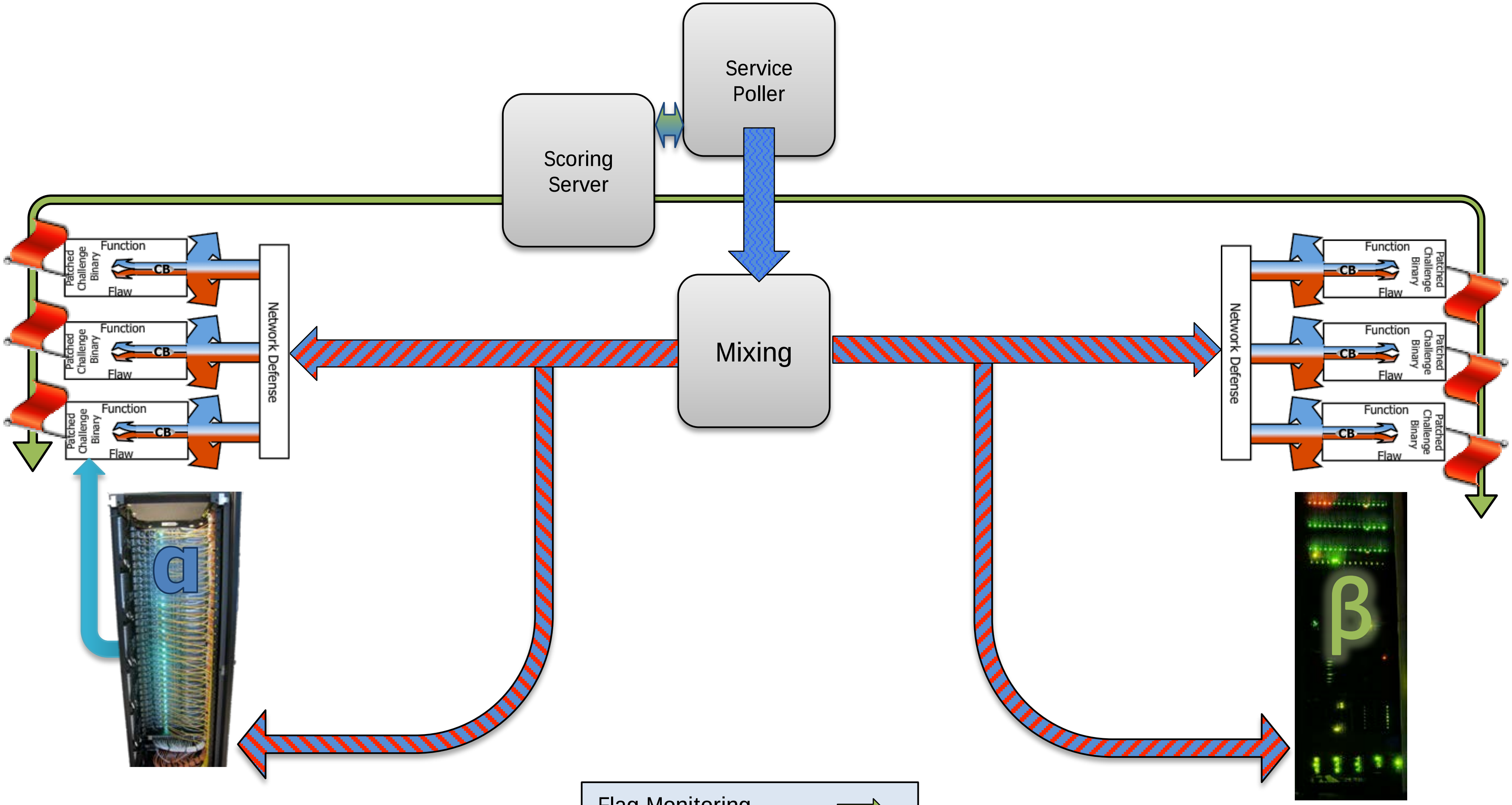
Patched Challenge Binary

α

β

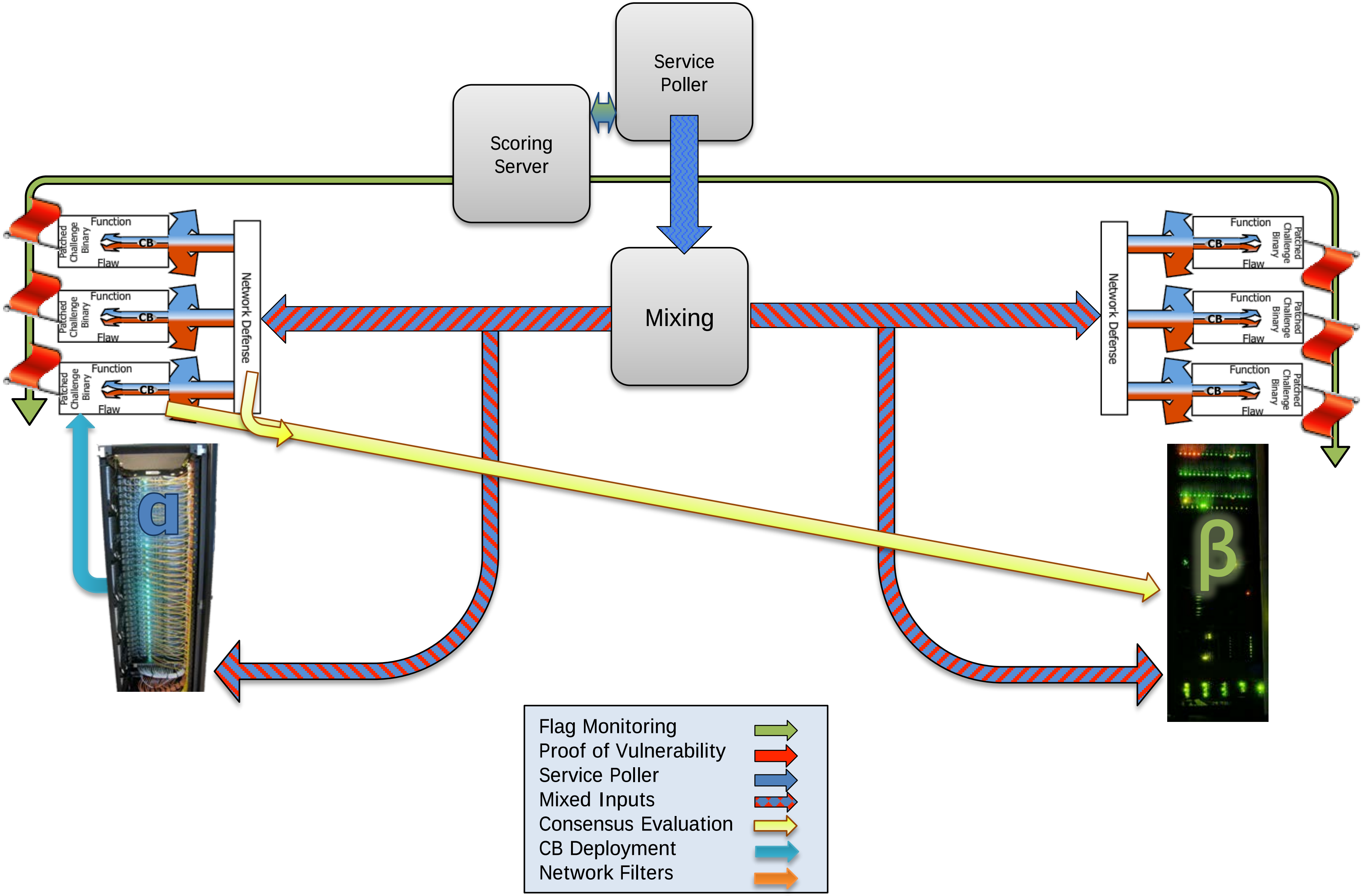
Flag Monitoring
Proof of Vulnerability
Service Poller
Mixed Inputs
Consensus Evaluation
CB Deployment
Network Filters

Consensus Evaluation

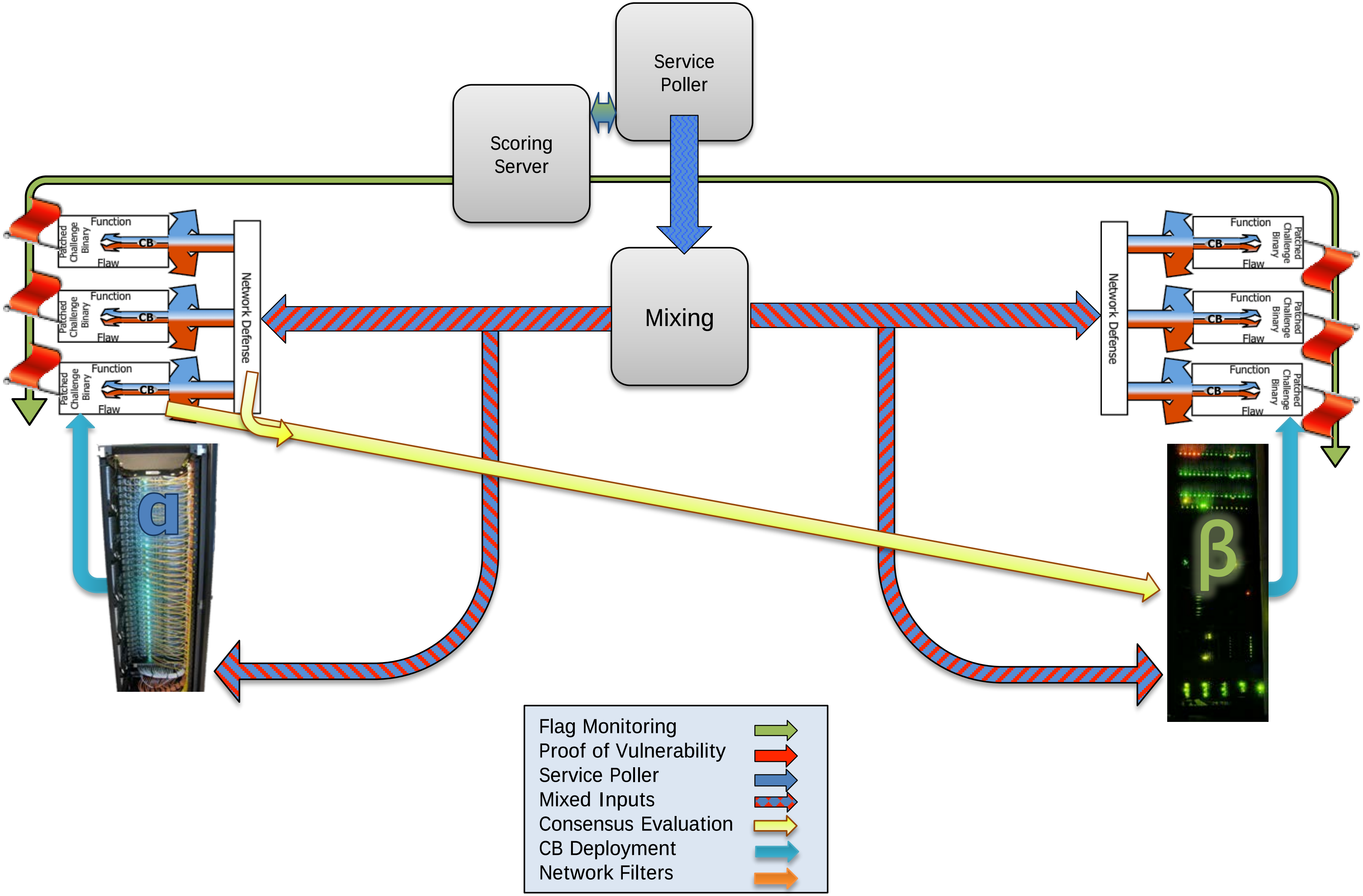


- Flag Monitoring 🟩
- Proof of Vulnerability 🟥
- Service Poller 🟦
- Mixed Inputs ▨
- Consensus Evaluation 🟨
- CB Deployment 🟦
- Network Filters 🟧

Consensus Evaluation

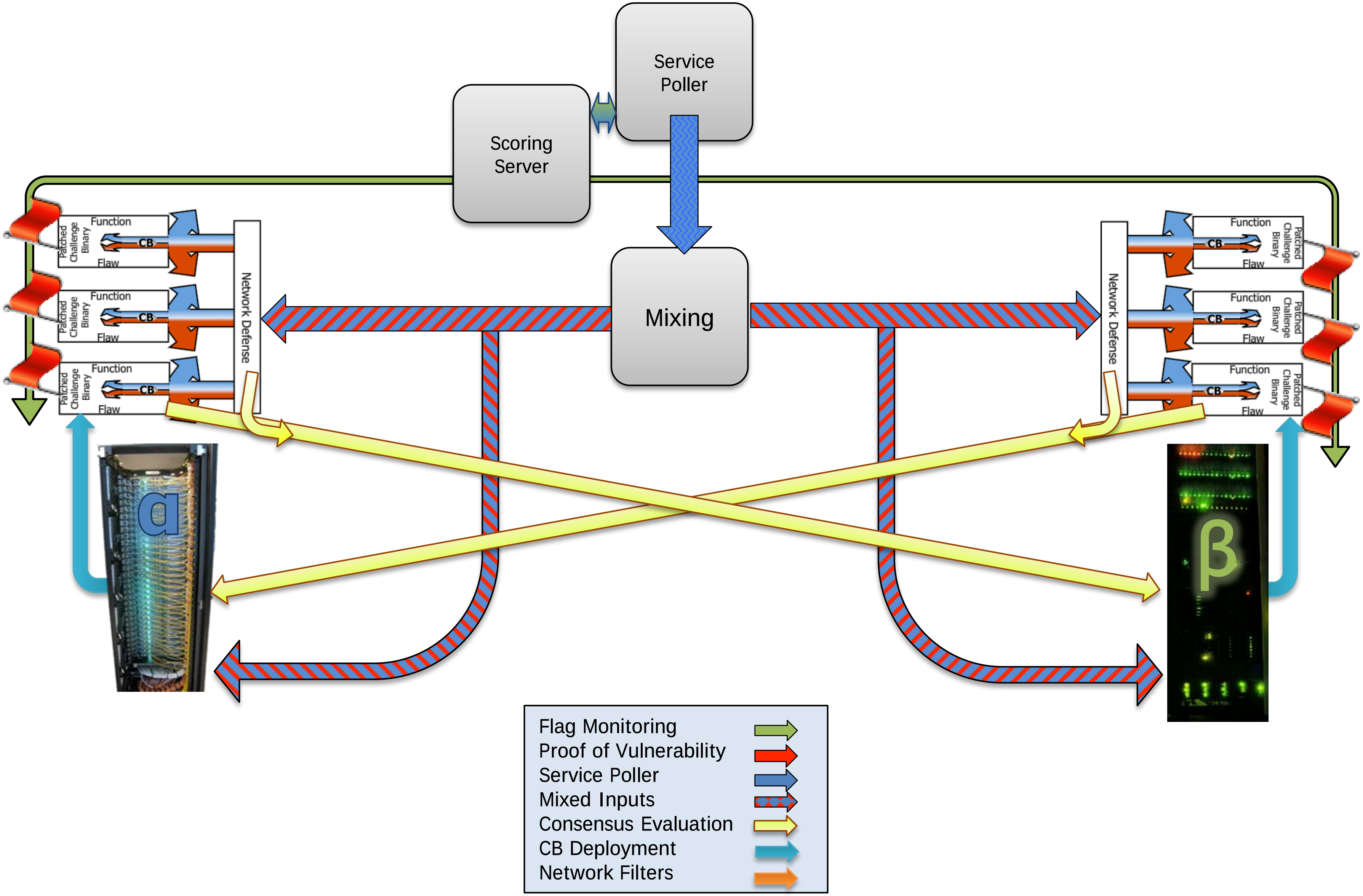


Consensus Evaluation



Flag Monitoring
Proof of Vulnerability
Service Poller
Mixed Inputs
Consensus Evaluation
CB Deployment
Network Filters

Consensus Evaluation



Service Poller

Scoring Server

Mixing

Network Defense

Network Defense

Patched Challenge Binary

Function

CB

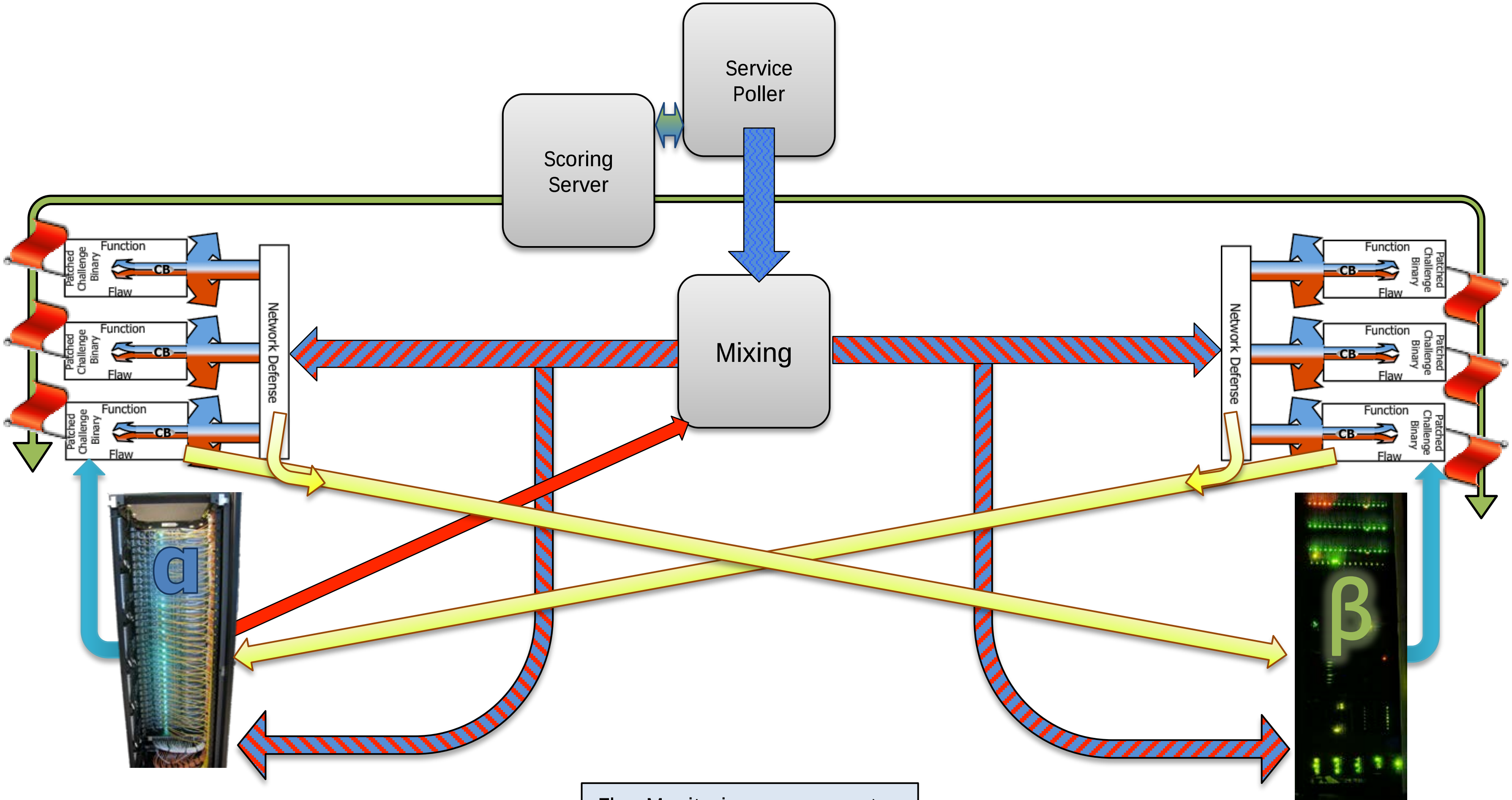
Flaw

α

β

Flag Monitoring
Proof of Vulnerability
Service Poller
Mixed Inputs
Consensus Evaluation
CB Deployment
Network Filters

Consensus Evaluation



Service Poller

Scoring Server

Mixing

Network Defense

Network Defense

Patched Challenge Binary

Function

CB

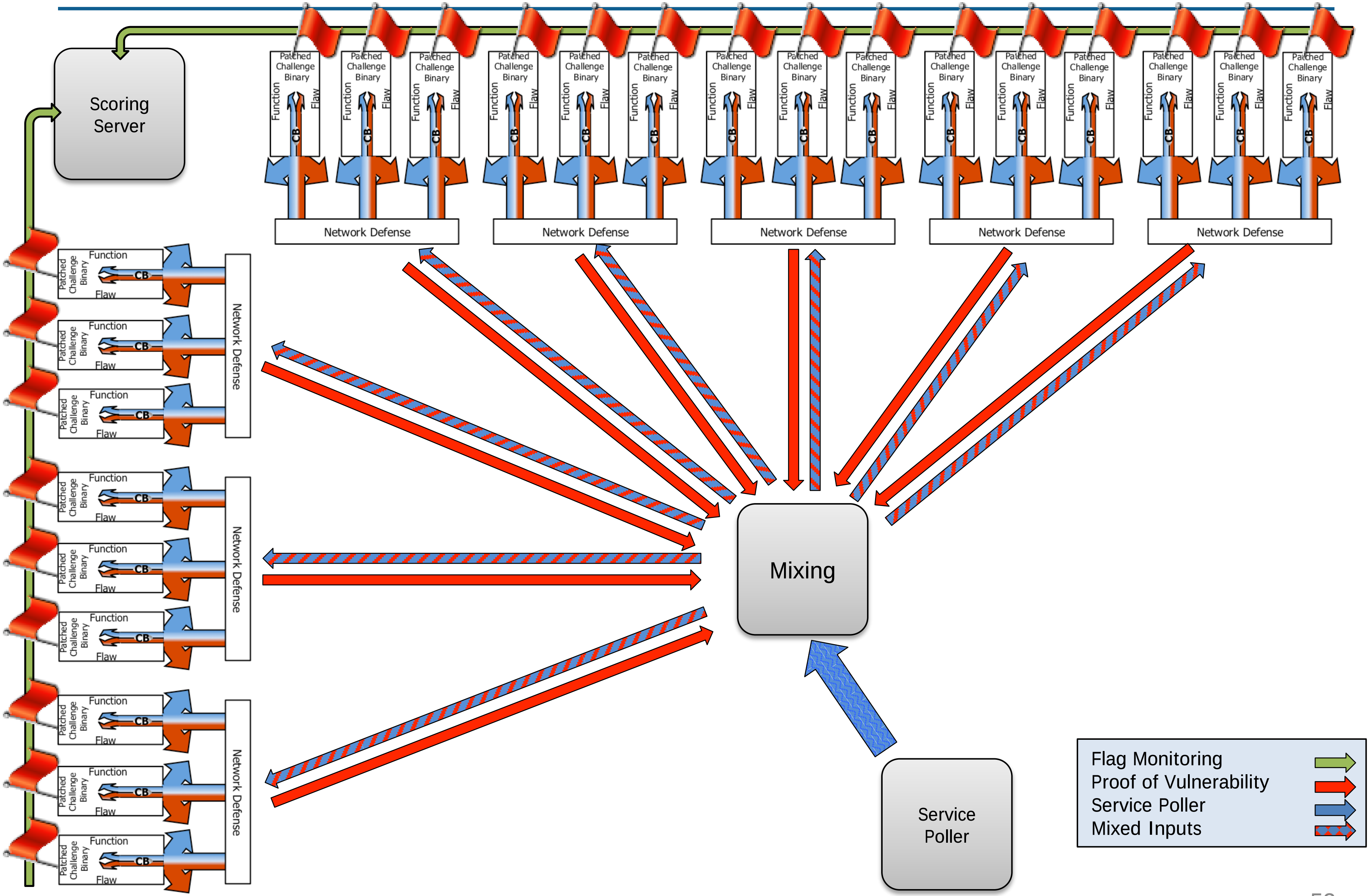
Flaw

α

β

Flag Monitoring
Proof of Vulnerability
Service Poller
Mixed Inputs
Consensus Evaluation
CB Deployment
Network Filters

Scheduled Final Event: Multi-Team Real Time Tournament



Flag Monitoring 🟩
Proof of Vulnerability 🟥
Service Poller 🟦
Mixed Inputs ▨

Approved for Public Release, Distribution Unlimited

Participation

- Build a team and sign up @ https://cgc.darpa.mil
- Lots of relevant work in ISSTA 2014.
  - Session 1: Concurrency and Verification
  - Session 3: Artifact Studies
  - Session 4: Static Analyses and Transformations
  - Session 5: Test Selection and Reduction
  - Session 6: Localization and Repair
  - Session 7: Security

Artifact * AEC * Evaluated * ISSTA *

Consistent * Complete * Well Documented * Easy to Reuse *

Annual Competition / Conference?

- You have the infrastructure
- You *will* have the challenge binaries, all source code, POVs ...
- You have the expertise
- You have the power to keep CGC alive after August 2016
  - Games have the International Computer Games Association (http://www.icga.org/)
  - Robotics have the RoboCup (http://www.robocup2014.org)
  - Turing Test has the Loebner Prize Competition (http://www.loebner.net/Prizef/loebner-prize.html)
  - Artificial Intelligence has the AAAI Annual Computer Poker Competition (http://www.computerpokercompetition.org/)
  - Satisfiability has the SAT Competition (http://www.satcompetition.org/)
  - Satisfiability Modulo Theories has the SMT Competition (http://smtcomp.sourceforge.net/2014/index.shtml)
  - Software Testing and Security have ???

CGC CYBER
GRAND_CHALLENGE

For more information:

www.darpa.mil/cybergrandchallenge