



```
addiu $sp, -0x18
sw $ra, 0x18+var_4($sp)
sw $a0, 0x18+arg_0($sp)
lui $1, 3
jal sub_2DAB8
lw $a0, dword_35A6C
lui $1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t9, $t8, 4
sltu $1, $v0, $t9
beqz $1, loc_2DA24
nop
sub 2DAB8
```

Developments in Cisco IOS Forensics

Felix 'FX' Lindner

BlackHat Briefings

Las Vegas, August 2008

```
move $a0, $t7
lw $a0, dword_35A6C
jal sub_2DAD4
addiu $a1, $v0, 0x10
beqz $v0, loc_2DA44
move $v0, $0
la $1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($1)
subu $t2, $t0, $t5
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($1)
sw $v0, dword_35A6C
```

Invent & Verify

Agenda

- IP Routing Infrastructure and Cisco IOS
- Cisco IOS Internals
- Debugging and Post Mortem Analysis Today
- A New Analysis Approach
 - Proposal
 - Features
 - Challenges
- Public Offer
- Future Work

Invent & Verify



```

move $a0, $t7
lw $a0, dword_35A70
jal sub_2DAD8
addiu $a1, $v0, 0x10
beqz $v0, $v0, $v0
move $v0, $v0
lw $t1, dword_35A70
lw $t1, dword_35A6C
lw $t1, $v0
subu $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

```

addiu $sp, $ra, -4
sw $ra, $a0
sw $a0, $a0
lwi $t1, 2
jal sub_2DAB8
lw $a0, dword_35A6C
lwi $t1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllv $t1, $v0, $t8
lwi $t1, 10x_2DA24
sub $t1, $t1

```

IP Routing Infrastructure

- The Internet and corporate networks almost exclusively run on the Internet Protocol
 - IP Version 4 is still prevalent protocol
 - IP Version 6 coming up very slowly
- The design of IP requires intelligent nodes in the network to make routing decisions
 - This is a design principle of the protocol and cannot be changed
 - “Flat” networks have their own issues

```
move $a0, $t7
lw $a0, dword_35A6C
jal sub_35AD4
addiu $a1, $v0, 0x10
beqz $v0, loc_2DA44
move $v0, $0
la $t1, dword_35A70
lw $t0, dword_35A6C
lw $t2, $t0, 2
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

```
addiu $ap, $ra, 0x10
sw $ra, $a0
sw $a0, $a0
lwf $t1, 2
fsl $f1, 35AB8
lwf $t2, dword_35A6C
lwf $t7, dword_35A6C
lwf $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllw $t1, $v0, $t8
```

Invent & Verify



IP Infrastructure & Security

- All security protocols on top of IP share common design goals:
 - Guarantee end-to-end integrity (some also confidentiality) of the traffic
 - Detect modification, replay, injection and holding back of traffic
 - Inform the upper protocol layers
- None of them can recover from attacks rooted in the routing infrastructure
 - Security protocols cannot influence routing

Invent & Verify



```
move $a0, $t7
lw $a1, dword_35A6C
jal $a0, 0
addiu $a0, $v0, 0x10
beqz $v0, loc_35A44
move $v0, 0
la $t1, dword_35A70
lw $t0, 0($t1)
subu $t2, $t0, 1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

```
addiu $sp, $ra, -4
sw $ra, 0($sp)
sw $a0, 4($sp)
lwf $t1, 2($sp)
lwf $t2, 2($sp)
lw $a0, dword_35A6C
lwf $t1, 2($sp)
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllr $t1, $v0, $t8
sll $t3, $v0, $t8
sub $t3, $t3, $t1
```

Infrastructure Monoculture

- Cisco Systems' routing platforms form the single largest population of networking equipment today
 - Equivalently distributed in the Internet core, government and corporate networks
 - Many different hardware platforms with different CPUs
 - Large investment sums bound to the equipment
 - Hard to replace
 - All run basically the same operating system
- Protecting this infrastructure is critical
- Therefore, in-depth analysis and diagnostics are of paramount importance

Invent & Verify



```
move $a0, $v7
lw $a1, dword_35A6C
jal sw $a1, $v0, $v0
addiu $a1, $v0, $v0
beqz $v0, loc_2DA44
move $v0, $v0
la $t1, dword_35A6C
lw $t1, dword_35A6C
lw $t1, dword_35A70
subu $t1, $t1, $t7
sw $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```



Cisco IOS

```
addiu $sp, $ra, -40
sw $ra, 0($sp)
sw $a0, 4($sp)
lui $t1, 2
jal sub_2DAB8
lw $a0, dword_35A6C
lui $t1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllr $t1, $v0, $t8
beqz $t1, loc_2DA24
nop
sub $t1, $t1, 1
```

- Cisco® Internetwork Operating System®
- Monolithic operating system
- Compile-time linked functionality – the 3 dimensional complexity of IOS
 - Platform dependent code
 - Feature-set dependent code
 - Major, Minor and Release version dependent code
- Several ***tens of thousands different*** IOS images used in today's networks
 - Over 10.000 still officially supported

```
move $a0, $v0
lw $a0, dword_35A6C
jal sub_2DAB8
addiu $a0, $a0, 4
beqz $v0, loc_2DA24
move $v0, $0
la $t1, 0($t7)
lw $t1, 0($t1)
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

Invent & Verify





Inside Cisco IOS

```

addiu $sp, $ra, 4
sw $ra, 0($sp)
sw $a0, 4($sp)
lui $t1, 3
jal sub_2DAB8
lw $a0, dword_35A6C
lui $t1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllr $t1, $t0, $t8
beqz $t1, loc_2DA24
nop
sub $t1, $t1, 1

```

- One large ELF binary
- Essentially a large, statically linked UNIX program
 - Loaded by ROMMON, a kind-of BIOS
- Runs directly on the router's main CPU
 - If the CPU provides virtual memory and privilege separation (for example Supervisor and User mode on MIPS), it will not be used

```

move $a0, $t7
lw $a0, dword_35A6C
jal sub_2DAB8
addiu $a1, $v0, 4
beqz $v0, loc_2DA44
move $v0, $0
la $t1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

Invent & Verify





Inside Cisco IOS

```
addiu $sp, $ra, -4  
sw $ra, 4($sp)  
sw $a0, 4($sp)  
lui $t1, 3  
jal sub_2DAB8  
lw $a0, dword_35A6C  
lui $t1, 3  
lw $t7, dword_35A6C  
lw $t6, dword_35A70  
subu $t8, $t6, $t7  
addiu $t2, $t6, 4  
sllr $t1, $v0, $t8  
beqz $t1, loc_2DA24  
nop  
sub $t1, $t1, 1
```

- Processes are rather like threads
 - No virtual memory mapping per process
- Run-to-completion, cooperative multitasking
 - Interrupt driven handling of critical events
- System-wide global data structures
 - Common heap
 - Very little abstraction around the data structures
 - No way to force abstraction

```
move $a0, $v1  
lw $a0, dword_35A6C  
jal sub_2DAB8  
addiu $a1, $v0, 4  
beqz $v0, loc_2DA24  
move $v0, $0  
la $t1, dword_35A70  
lw $t1, dword_35A6C  
lw $t0, 0($t1)  
subu $t2, $t0, $t1  
sra $t3, $t2, 2  
sll $t4, $t3, 2  
addu $t5, $v0, $t4  
sw $t5, 0($t1)  
sw $v0, dword_35A6C
```

Invent & Verify



The IOS Code Security Issue

- 12.4(16a) with enterprise base feature set consists of 25.316.780 bytes binary code!
 - This is a 2600 with PowerPC CPU
 - Not including 505.900 bytes firmware for E1T1 and initialization
- All written in plain C
- Sharing the same address space
- Sharing the same heap
- Sharing the same data structures
- Sharing millions of pointers

Invent & Verify



```
move $a0, $v0
lw $a0, dword_35A6C
jal $a0
addiu $a0, $v0, 2
beqz $v0, loc_2DA44
move $v0, $0
la $t0, dword_35A6C
lw $t0, 0($t0)
subu $t0, $t0, $v0
sra $t0, $t0, 2
sll $t5, $v0, $t4
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

```
addiu $ep, $ra, 4
sw $ra, 4($ep)
sw $a0, 8($ep)
lwi $t1, 2($ep)
subu $t1, $t1, 2
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllr $t1, $t1, $t8
```

The IOS Code Security Issue

- A single mistake in the most unimportant piece of code can influence anything on the system, including kernel, security subsystems and cryptographic code.
- Therefore, **everything** on IOS is a good target for remote code execution exploits in kernel context.

```
move $a0, $t7
lw $a1, 0($a0)
jal sub_2DAB8
addiu $a1, $v0, 0x10
beqz $v0, loc_2DA44
move $v0, $0
la $t1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

```
addiu $sp, $ra, -4
sw $ra, 0($sp)
sw $a0, 4($sp)
lw $t1, 8($sp)
sub $t1, $t1, 2
lw $t1, dword_35A68
lw $t1, dword_35A6C
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllr $t1, $v0, $t8
beqz $t1, loc_2DA24
sub $t1, $t1, 2
```





Isn't Cisco aware of that?

- Cisco recently started the distribution of the next generation IOS-XR
 - Commercial QNX microkernel
 - Real processes (memory protection?)
 - Concurrent scheduling
 - Significantly higher hardware requirements (as in Cisco 12000 !)
- People never use the latest IOS
 - Production corporate networks usually run on 12.1 or 12.2, which 12.5 is already available
 - Not even Cisco's own engineers would recommend the latest IOS release to a customer
 - That only covers people actively maintaining their network, not everyone running one

```

move $a0, $t0
lw $a0, 0($a0)
jal sub_2DA44
addiu $a1, $v0, 2DA44
beqz $v0, $t0, 2DA44
move $v0, $t0
la $t1, dword_35A70
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

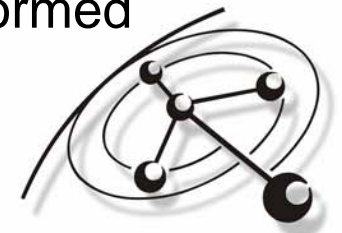
```

addiu $sp, $ra, -4
sw $ra, 0($sp)
sw $a0, 4($sp)
lwf $t1, 0($sp)
lwf $t2, 4($sp)
lwf $t3, 8($sp)
lwf $t4, 12($sp)
lwf $t5, 16($sp)
lwf $t6, 20($sp)
lwf $t7, 24($sp)
lwf $t8, 28($sp)
lwf $t9, 32($sp)
lwf $t10, 36($sp)
lwf $t11, 40($sp)
lwf $t12, 44($sp)
lwf $t13, 48($sp)
lwf $t14, 52($sp)
lwf $t15, 56($sp)
lwf $t16, 60($sp)
lwf $t17, 64($sp)
lwf $t18, 68($sp)
lwf $t19, 72($sp)
lwf $t20, 76($sp)
lwf $t21, 80($sp)
lwf $t22, 84($sp)
lwf $t23, 88($sp)
lwf $t24, 92($sp)
lwf $t25, 96($sp)
lwf $t26, 100($sp)
lwf $t27, 104($sp)
lwf $t28, 108($sp)
lwf $t29, 112($sp)
lwf $t30, 116($sp)
lwf $t31, 120($sp)
lwf $t32, 124($sp)
lwf $t33, 128($sp)
lwf $t34, 132($sp)
lwf $t35, 136($sp)
lwf $t36, 140($sp)
lwf $t37, 144($sp)
lwf $t38, 148($sp)
lwf $t39, 152($sp)
lwf $t40, 156($sp)
lwf $t41, 160($sp)
lwf $t42, 164($sp)
lwf $t43, 168($sp)
lwf $t44, 172($sp)
lwf $t45, 176($sp)
lwf $t46, 180($sp)
lwf $t47, 184($sp)
lwf $t48, 188($sp)
lwf $t49, 192($sp)
lwf $t50, 196($sp)
lwf $t51, 200($sp)
lwf $t52, 204($sp)
lwf $t53, 208($sp)
lwf $t54, 212($sp)
lwf $t55, 216($sp)
lwf $t56, 220($sp)
lwf $t57, 224($sp)
lwf $t58, 228($sp)
lwf $t59, 232($sp)
lwf $t60, 236($sp)
lwf $t61, 240($sp)
lwf $t62, 244($sp)
lwf $t63, 248($sp)
lwf $t64, 252($sp)
lwf $t65, 256($sp)
lwf $t66, 260($sp)
lwf $t67, 264($sp)
lwf $t68, 268($sp)
lwf $t69, 272($sp)
lwf $t70, 276($sp)
lwf $t71, 280($sp)
lwf $t72, 284($sp)
lwf $t73, 288($sp)
lwf $t74, 292($sp)
lwf $t75, 296($sp)
lwf $t76, 300($sp)
lwf $t77, 304($sp)
lwf $t78, 308($sp)
lwf $t79, 312($sp)
lwf $t80, 316($sp)
lwf $t81, 320($sp)
lwf $t82, 324($sp)
lwf $t83, 328($sp)
lwf $t84, 332($sp)
lwf $t85, 336($sp)
lwf $t86, 340($sp)
lwf $t87, 344($sp)
lwf $t88, 348($sp)
lwf $t89, 352($sp)
lwf $t90, 356($sp)
lwf $t91, 360($sp)
lwf $t92, 364($sp)
lwf $t93, 368($sp)
lwf $t94, 372($sp)
lwf $t95, 376($sp)
lwf $t96, 380($sp)
lwf $t97, 384($sp)
lwf $t98, 388($sp)
lwf $t99, 392($sp)
lwf $t100, 396($sp)
lwf $t101, 400($sp)
lwf $t102, 404($sp)
lwf $t103, 408($sp)
lwf $t104, 412($sp)
lwf $t105, 416($sp)
lwf $t106, 420($sp)
lwf $t107, 424($sp)
lwf $t108, 428($sp)
lwf $t109, 432($sp)
lwf $t110, 436($sp)
lwf $t111, 440($sp)
lwf $t112, 444($sp)
lwf $t113, 448($sp)
lwf $t114, 452($sp)
lwf $t115, 456($sp)
lwf $t116, 460($sp)
lwf $t117, 464($sp)
lwf $t118, 468($sp)
lwf $t119, 472($sp)
lwf $t120, 476($sp)
lwf $t121, 480($sp)
lwf $t122, 484($sp)
lwf $t123, 488($sp)
lwf $t124, 492($sp)
lwf $t125, 496($sp)
lwf $t126, 500($sp)
lwf $t127, 504($sp)
lwf $t128, 508($sp)
lwf $t129, 512($sp)
lwf $t130, 516($sp)
lwf $t131, 520($sp)
lwf $t132, 524($sp)
lwf $t133, 528($sp)
lwf $t134, 532($sp)
lwf $t135, 536($sp)
lwf $t136, 540($sp)
lwf $t137, 544($sp)
lwf $t138, 548($sp)
lwf $t139, 552($sp)
lwf $t140, 556($sp)
lwf $t141, 560($sp)
lwf $t142, 564($sp)
lwf $t143, 568($sp)
lwf $t144, 572($sp)
lwf $t145, 576($sp)
lwf $t146, 580($sp)
lwf $t147, 584($sp)
lwf $t148, 588($sp)
lwf $t149, 592($sp)
lwf $t150, 596($sp)
lwf $t151, 600($sp)
lwf $t152, 604($sp)
lwf $t153, 608($sp)
lwf $t154, 612($sp)
lwf $t155, 616($sp)
lwf $t156, 620($sp)
lwf $t157, 624($sp)
lwf $t158, 628($sp)
lwf $t159, 632($sp)
lwf $t160, 636($sp)
lwf $t161, 640($sp)
lwf $t162, 644($sp)
lwf $t163, 648($sp)
lwf $t164, 652($sp)
lwf $t165, 656($sp)
lwf $t166, 660($sp)
lwf $t167, 664($sp)
lwf $t168, 668($sp)
lwf $t169, 672($sp)
lwf $t170, 676($sp)
lwf $t171, 680($sp)
lwf $t172, 684($sp)
lwf $t173, 688($sp)
lwf $t174, 692($sp)
lwf $t175, 696($sp)
lwf $t176, 700($sp)
lwf $t177, 704($sp)
lwf $t178, 708($sp)
lwf $t179, 712($sp)
lwf $t180, 716($sp)
lwf $t181, 720($sp)
lwf $t182, 724($sp)
lwf $t183, 728($sp)
lwf $t184, 732($sp)
lwf $t185, 736($sp)
lwf $t186, 740($sp)
lwf $t187, 744($sp)
lwf $t188, 748($sp)
lwf $t189, 752($sp)
lwf $t190, 756($sp)
lwf $t191, 760($sp)
lwf $t192, 764($sp)
lwf $t193, 768($sp)
lwf $t194, 772($sp)
lwf $t195, 776($sp)
lwf $t196, 780($sp)
lwf $t197, 784($sp)
lwf $t198, 788($sp)
lwf $t199, 792($sp)
lwf $t200, 796($sp)
lwf $t201, 800($sp)
lwf $t202, 804($sp)
lwf $t203, 808($sp)
lwf $t204, 812($sp)
lwf $t205, 816($sp)
lwf $t206, 820($sp)
lwf $t207, 824($sp)
lwf $t208, 828($sp)
lwf $t209, 832($sp)
lwf $t210, 836($sp)
lwf $t211, 840($sp)
lwf $t212, 844($sp)
lwf $t213, 848($sp)
lwf $t214, 852($sp)
lwf $t215, 856($sp)
lwf $t216, 860($sp)
lwf $t217, 864($sp)
lwf $t218, 868($sp)
lwf $t219, 872($sp)
lwf $t220, 876($sp)
lwf $t221, 880($sp)
lwf $t222, 884($sp)
lwf $t223, 888($sp)
lwf $t224, 892($sp)
lwf $t225, 896($sp)
lwf $t226, 900($sp)
lwf $t227, 904($sp)
lwf $t228, 908($sp)
lwf $t229, 912($sp)
lwf $t230, 916($sp)
lwf $t231, 920($sp)
lwf $t232, 924($sp)
lwf $t233, 928($sp)
lwf $t234, 932($sp)
lwf $t235, 936($sp)
lwf $t236, 940($sp)
lwf $t237, 944($sp)
lwf $t238, 948($sp)
lwf $t239, 952($sp)
lwf $t240, 956($sp)
lwf $t241, 960($sp)
lwf $t242, 964($sp)
lwf $t243, 968($sp)
lwf $t244, 972($sp)
lwf $t245, 976($sp)
lwf $t246, 980($sp)
lwf $t247, 984($sp)
lwf $t248, 988($sp)
lwf $t249, 992($sp)
lwf $t250, 996($sp)
lwf $t251, 1000($sp)
lwf $t252, 1004($sp)
lwf $t253, 1008($sp)
lwf $t254, 1012($sp)
lwf $t255, 1016($sp)
lwf $t256, 1020($sp)
lwf $t257, 1024($sp)
lwf $t258, 1028($sp)
lwf $t259, 1032($sp)
lwf $t260, 1036($sp)
lwf $t261, 1040($sp)
lwf $t262, 1044($sp)
lwf $t263, 1048($sp)
lwf $t264, 1052($sp)
lwf $t265, 1056($sp)
lwf $t266, 1060($sp)
lwf $t267, 1064($sp)
lwf $t268, 1068($sp)
lwf $t269, 1072($sp)
lwf $t270, 1076($sp)
lwf $t271, 1080($sp)
lwf $t272, 1084($sp)
lwf $t273, 1088($sp)
lwf $t274, 1092($sp)
lwf $t275, 1096($sp)
lwf $t276, 1100($sp)
lwf $t277, 1104($sp)
lwf $t278, 1108($sp)
lwf $t279, 1112($sp)
lwf $t280, 1116($sp)
lwf $t281, 1120($sp)
lwf $t282, 1124($sp)
lwf $t283, 1128($sp)
lwf $t284, 1132($sp)
lwf $t285, 1136($sp)
lwf $t286, 1140($sp)
lwf $t287, 1144($sp)
lwf $t288, 1148($sp)
lwf $t289, 1152($sp)
lwf $t290, 1156($sp)
lwf $t291, 1160($sp)
lwf $t292, 1164($sp)
lwf $t293, 1168($sp)
lwf $t294, 1172($sp)
lwf $t295, 1176($sp)
lwf $t296, 1180($sp)
lwf $t297, 1184($sp)
lwf $t298, 1188($sp)
lwf $t299, 1192($sp)
lwf $t300, 1196($sp)
lwf $t301, 1200($sp)
lwf $t302, 1204($sp)
lwf $t303, 1208($sp)
lwf $t304, 1212($sp)
lwf $t305, 1216($sp)
lwf $t306, 1220($sp)
lwf $t307, 1224($sp)
lwf $t308, 1228($sp)
lwf $t309, 1232($sp)
lwf $t310, 1236($sp)
lwf $t311, 1240($sp)
lwf $t312, 1244($sp)
lwf $t313, 1248($sp)
lwf $t314, 1252($sp)
lwf $t315, 1256($sp)
lwf $t316, 1260($sp)
lwf $t317, 1264($sp)
lwf $t318, 1268($sp)
lwf $t319, 1272($sp)
lwf $t320, 1276($sp)
lwf $t321, 1280($sp)
lwf $t322, 1284($sp)
lwf $t323, 1288($sp)
lwf $t324, 1292($sp)
lwf $t325, 1296($sp)
lwf $t326, 1300($sp)
lwf $t327, 1304($sp)
lwf $t328, 1308($sp)
lwf $t329, 1312($sp)
lwf $t330, 1316($sp)
lwf $t331, 1320($sp)
lwf $t332, 1324($sp)
lwf $t333, 1328($sp)
lwf $t334, 1332($sp)
lwf $t335, 1336($sp)
lwf $t336, 1340($sp)
lwf $t337, 1344($sp)
lwf $t338, 1348($sp)
lwf $t339, 1352($sp)
lwf $t340, 1356($sp)
lwf $t341, 1360($sp)
lwf $t342, 1364($sp)
lwf $t343, 1368($sp)
lwf $t344, 1372($sp)
lwf $t345, 1376($sp)
lwf $t346, 1380($sp)
lwf $t347, 1384($sp)
lwf $t348, 1388($sp)
lwf $t349, 1392($sp)
lwf $t350, 1396($sp)
lwf $t351, 1400($sp)
lwf $t352, 1404($sp)
lwf $t353, 1408($sp)
lwf $t354, 1412($sp)
lwf $t355, 1416($sp)
lwf $t356, 1420($sp)
lwf $t357, 1424($sp)
lwf $t358, 1428($sp)
lwf $t359, 1432($sp)
lwf $t360, 1436($sp)
lwf $t361, 1440($sp)
lwf $t362, 1444($sp)
lwf $t363, 1448($sp)
lwf $t364, 1452($sp)
lwf $t365, 1456($sp)
lwf $t366, 1460($sp)
lwf $t367, 1464($sp)
lwf $t368, 1468($sp)
lwf $t369, 1472($sp)
lwf $t370, 1476($sp)
lwf $t371, 1480($sp)
lwf $t372, 1484($sp)
lwf $t373, 1488($sp)
lwf $t374, 1492($sp)
lwf $t375, 1496($sp)
lwf $t376, 1500($sp)
lwf $t377, 1504($sp)
lwf $t378, 1508($sp)
lwf $t379, 1512($sp)
lwf $t380, 1516($sp)
lwf $t381, 1520($sp)
lwf $t382, 1524($sp)
lwf $t383, 1528($sp)
lwf $t384, 1532($sp)
lwf $t385, 1536($sp)
lwf $t386, 1540($sp)
lwf $t387, 1544($sp)
lwf $t388, 1548($sp)
lwf $t389, 1552($sp)
lwf $t390, 1556($sp)
lwf $t391, 1560($sp)
lwf $t392, 1564($sp)
lwf $t393, 1568($sp)
lwf $t394, 1572($sp)
lwf $t395, 1576($sp)
lwf $t396, 1580($sp)
lwf $t397, 1584($sp)
lwf $t398, 1588($sp)
lwf $t399, 1592($sp)
lwf $t400, 1596($sp)
lwf $t401, 1600($sp)
lwf $t402, 1604($sp)
lwf $t403, 1608($sp)
lwf $t404, 1612($sp)
lwf $t405, 1616($sp)
lwf $t406, 1620($sp)
lwf $t407, 1624($sp)
lwf $t408, 1628($sp)
lwf $t409, 1632($sp)
lwf $t410, 1636($sp)
lwf $t411, 1640($sp)
lwf $t412, 1644($sp)
lwf $t413, 1648($sp)
lwf $t414, 1652($sp)
lwf $t415, 1656($sp)
lwf $t416, 1660($sp)
lwf $t417, 1664($sp)
lwf $t418, 1668($sp)
lwf $t419, 1672($sp)
lwf $t420, 1676($sp)
lwf $t421, 1680($sp)
lwf $t422, 1684($sp)
lwf $t423, 1688($sp)
lwf $t424, 1692($sp)
lwf $t425, 1696($sp)
lwf $t426, 1700($sp)
lwf $t427, 1704($sp)
lwf $t428, 1708($sp)
lwf $t429, 1712($sp)
lwf $t430, 1716($sp)
lwf $t431, 1720($sp)
lwf $t432, 1724($sp)
lwf $t433, 1728($sp)
lwf $t434, 1732($sp)
lwf $t435, 1736($sp)
lwf $t436, 1740($sp)
lwf $t437, 1744($sp)
lwf $t438, 1748($sp)
lwf $t439, 1752($sp)
lwf $t440, 1756($sp)
lwf $t441, 1760($sp)
lwf $t442, 1764($sp)
lwf $t443, 1768($sp)
lwf $t444, 1772($sp)
lwf $t445, 1776($sp)
lwf $t446, 1780($sp)
lwf $t447, 1784($sp)
lwf $t448, 1788($sp)
lwf $t449, 1792($sp)
lwf $t450, 1796($sp)
lwf $t451, 1800($sp)
lwf $t452, 1804($sp)
lwf $t453, 1808($sp)
lwf $t454, 1812($sp)
lwf $t455, 1816($sp)
lwf $t456, 1820($sp)
lwf $t457, 1824($sp)
lwf $t458, 1828($sp)
lwf $t459, 1832($sp)
lwf $t460, 1836($sp)
lwf $t461, 1840($sp)
lwf $t462, 1844($sp)
lwf $t463, 1848($sp)
lwf $t464, 1852($sp)
lwf $t465, 1856($sp)
lwf $t466, 1860($sp)
lwf $t467, 1864($sp)
lwf $t468, 1868($sp)
lwf $t469, 1872($sp)
lwf $t470, 1876($sp)
lwf $t471, 1880($sp)
lwf $t472, 1884($sp)
lwf $t473, 1888($sp)
lwf $t474, 1892($sp)
lwf $t475, 1896($sp)
lwf $t476, 1900($sp)
lwf $t477, 1904($sp)
lwf $t478, 1908($sp)
lwf $t479, 1912($sp)
lwf $t480, 1916($sp)
lwf $t481, 1920($sp)
lwf $t482, 1924($sp)
lwf $t483, 1928($sp)
lwf $t484, 1932($sp)
lwf $t485, 1936($sp)
lwf $t486, 1940($sp)
lwf $t487, 1944($sp)
lwf $t488, 1948($sp)
lwf $t489, 1952($sp)
lwf $t490, 1956($sp)
lwf $t491, 1960($sp)
lwf $t492, 1964($sp)
lwf $t493, 1968($sp)
lwf $t494, 1972($sp)
lwf $t495, 1976($sp)
lwf $t496, 1980($sp)
lwf $t497, 1984($sp)
lwf $t498, 1988($sp)
lwf $t499, 1992($sp)
lwf $t500, 1996($sp)
lwf $t501, 2000($sp)
lwf $t502, 2004($sp)
lwf $t503, 2008($sp)
lwf $t504, 2012($sp)
lwf $t505, 2016($sp)
lwf $t506, 2020($sp)
lwf $t507, 2024($sp)
lwf $t508, 2028($sp)
lwf $t509, 2032($sp)
lwf $t510, 2036($sp)
lwf $t511, 2040($sp)
lwf $t512, 2044($sp)
lwf $t513, 2048($sp)
lwf $t514, 2052($sp)
lwf $t515, 2056($sp)
lwf $t516, 2060($sp)
lwf $t517, 2064($sp)
lwf $t518, 2068($sp)
lwf $t519, 2072($sp)
lwf $t520, 2076($sp)
lwf $t521, 2080($sp)
lwf $t522, 2084($sp)
lwf $t523, 2088($sp)
lwf $t524, 2092($sp)
lwf $t525, 2096($sp)
lwf $t526, 2100($sp)
lwf $t527, 2104($sp)
lwf $t528, 2108($sp)
lwf $t529, 2112($sp)
lwf $t530, 2116($sp)
lwf $t531, 2120($sp)
lwf $t532, 2124($sp)
lwf $t533, 2128($sp)
lwf $t534, 2132($sp)
lwf $t535, 2136($sp)
lwf $t536, 2140($sp)
lwf $t537, 2144($sp)
lwf $t538, 2148($sp)
lwf $t539, 2152($sp)
lwf $t540, 2156($sp)
lwf $t541, 2160($sp)
lwf $t542, 2164($sp)
lwf $t543, 2168($sp)
lwf $t544, 2172($sp)
lwf $t545, 2176($sp)
lwf $t546, 2180($sp)
lwf $t547, 2184($sp)
lwf $t548, 2188($sp)
lwf $t549, 2192($sp)
lwf $t550, 2196($sp)
lwf $t551, 2200($sp)
lwf $t552, 2204($sp)
lwf $t553, 2208($sp)
lwf $t554, 2212($sp)
lwf $t555, 2216($sp)
lwf $t556, 2220($sp)
lwf $t557, 2224($sp)
lwf $t558, 2228($sp)
lwf $t559, 2232($sp)
lwf $t560, 2236($sp)
lwf $t561, 2240($sp)
lwf $t562, 2244($sp)
lwf $t563, 2248($sp)
lwf $t564, 2252($sp)
lwf $t565, 2256($sp)
lwf $t566, 2260($sp)
lwf $t567, 2264($sp)
lwf $t568, 2268($sp)
lwf $t569, 2272($sp)
lwf $t570, 2276($sp)
lwf $t571, 2280($sp)
lwf $t572, 2284($sp)
lwf $t573, 2288($sp)
lwf $t574, 2292($sp)
lwf $t575, 2296($sp)
lwf $t576, 2300($sp)
lwf $t577, 2304($sp)
lwf $t578, 2308($sp)
lwf $t579, 2312($sp)
lwf $t580, 2316($sp)
lwf $t581, 2320($sp)
lwf $t582, 2324($sp)
lwf $t583, 2328($sp)
lwf $t584, 2332($sp)
lwf $t585, 2336($sp)
lwf $t586, 2340($sp)
lwf $t587, 2344($sp)
lwf $t588, 2348($sp)
lwf $t589, 2352($sp)
lwf $t590, 2356($sp)
lwf $t591, 2360($sp)
lwf $t592, 2364($sp)
lwf $t593, 2368($sp)
lwf $t594, 2372($sp)
lwf $t595, 2376($sp)
lwf $t596, 2380($sp)
lwf $t597, 2384($sp)
lwf $t598, 2388($sp)
lwf $t599, 2392($sp)
lwf $t600, 2396($sp)
lwf $t601, 2400($sp)
lwf $t602, 2404($sp)
lwf $t603, 2408($sp)
lwf $t604, 2412($sp)
lwf $t605, 2416($sp)
lwf $t606, 2420($sp)
lwf $t607, 2424($sp)
lwf $t608, 2428($sp)
lwf $t609, 2432($sp)
lwf $t610, 2436($sp)
lwf $t611, 2440($sp)
lwf $t612, 2444($sp)
lwf $t613, 2448($sp)
lwf $t614, 2452($sp)
lwf $t615, 2456($sp)
lwf $t616, 2460($sp)
lwf $t617, 2464($sp)
lwf $t618, 2468($sp)
lwf $t619, 2472($sp)
lwf $t620, 2476($sp)
lwf $t621, 2480($sp)
lwf $t622, 2484($sp)
lwf $t623, 2488($sp)
lwf $t624, 2492($sp)
lwf $t625, 2496($sp)
lwf $t626, 2500($sp)
lwf $t627, 2504($sp)
lwf $t628, 2508($sp)
lwf $t629, 2512($sp)
lwf $t630, 2516($sp)
lwf $t631, 2520($sp)
lwf $t632, 2524($sp)
lwf $t633, 2528($sp)
lwf $t634, 2532($sp)
lwf $t635, 2536($sp)
lwf $t636, 2540($sp)
lwf $t637, 2544($sp)
lwf $t638, 2548($sp)
lwf $t639, 2552($sp)
lwf $t640, 2556($sp)
lwf $t641, 2560($sp)
lwf $t642, 2564($sp)
lwf $t643, 2568($sp)
lwf $t644, 2572($sp)
lwf $t645, 2576($sp)
lwf $t646, 2580($sp)
lwf $t647, 2584($sp)
lwf $t648, 2588($sp)
lwf $t649, 2592($sp)
lwf $t650, 2596($sp)
lwf $t651, 2600($sp)
lwf $t652, 2604($sp)
lwf $t653, 2608($sp)
lwf $t654, 2612($sp)
lwf $t655, 2616($sp)
lwf $t656, 2620($sp)
lwf $t657, 2624($sp)
lwf $t658, 2628($sp)
lwf $t659, 2632($sp)
lwf $t660, 2636($sp)
lwf $t661, 2640($sp)
lwf $t662, 2644($sp)
lwf $t663, 2648($sp)
lwf $t664, 2652($sp)
lwf $t665, 2656($sp)
lwf $t666, 2660($sp)
lwf $t667, 2664($sp)
lwf $t668, 2668($sp)
lwf $t669, 2672($sp)
lwf $t670, 2676($sp)
lwf $t671, 2680($sp)
lwf $t672, 2684($sp)
lwf $t673, 2688($sp)
lwf $t674, 2692($sp)
lwf $t675, 2696($sp)
lwf $t676, 2700($sp)
lwf $t677, 2704($sp)
lwf $t678, 2708($sp)
lwf $t679, 2712($sp)
lwf $t680, 2716($sp)
lwf $t681, 2720($sp)
lwf $t682, 2724($sp)
lwf $t683, 2728($sp)
lwf $t684, 2732($sp)
lwf $t685, 2736($sp)
lwf $t686, 2740($sp)
lwf $t687, 2744($sp)
lwf $t688, 2748($sp)
lwf $t689, 2752($sp)
lwf $t690, 2756($sp)
lwf $t691, 2760($sp)
lwf $t692, 2764($sp)
lwf $t693, 2768($sp)
lwf $t694, 2772($sp)
lwf $t695, 2776($sp)
lwf $t696, 2780($sp)
lwf $t697, 2784($sp)
lwf $t698, 2788($sp)
lwf $t699, 2792($sp)
lwf $t700, 2796($sp)
lwf $t701, 2800($sp)
lwf $t702, 2804($sp)
lwf $t703, 2808($sp)
lwf $t704, 2812($sp)
lwf $t705, 2816($sp)
lwf $t706, 2820($sp)
lwf $t707, 2824($sp)
lwf $t708, 2828($sp)
lwf $t709, 2832($sp)
lwf $t710, 2836($sp)
lwf $t711, 2840($sp)
lwf $t712, 2844($sp)
lwf $t713, 2848($sp)
lwf $t714, 2852
```

Just, how often are routers hacked?

- Keynote speaker Jerry Dixon at BlackHat Washington DC mentioned not updated routers as a cause for concern
 - Do you know how expensive that is?
- Old vulnerabilities like the HTTP level 16 bug are still actively scanned for
 - The router is used as a jump pad for further attacks
- TCL backdoors are commonly used
- Patched images are not rare
 - IOS images cost money
 - People will use images from anywhere
 - Patching images is not hard
- Lawful Interception is its own can of worms
 - The router's operator is not supposed to know that LI is performed
 - Who watches the watchers?

Invent & Verify



move \$a0, \$t0
lw \$a0, dword_35A6C
jal sub_2DAB8
addiu \$a1, \$v0, 0x10
beqz \$v0, \$t0, \$t1
move \$t1, \$t0
la \$t1, dword_35A70
lw \$t1, dword_35A70
lw \$t0, 0(\$t1)
subu \$t2, \$t0, 1
sra \$t3, \$t2, 2
sll \$t4, \$t3, 2
addu \$t5, \$v0, \$t4
sw \$t5, 0(\$t1)
sw \$v0, dword_35A6C

```
addiu $sp, $ra, -4  
sw $ra, 0($sp)  
sw $a0, 4($sp)  
lui $t1, 3  
jal sub_2DAB8  
lw $a0, dword_35A6C  
lui $t1, 3  
lw $t7, dword_35A6C  
lw $t6, dword_35A70  
subu $t8, $t6, $t7  
addiu $t2, $t6, 4  
li $t9, 0  
li $t9, 0
```

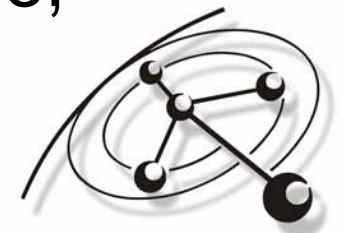


And the future?

- Ever noticed attackers take on the target with the lowest efforts required and the highest return of invest?
 - Windows became just a lot harder
 - UNIXes are hardened, even OS X
 - Infected PCs leave obvious traces
- The question is not:
“Will routers become a target?”
- The question should be:
“Do we want to know when they did?”
- Check the speaking schedule: 3 IOS talks here, 2 of them on attack methods

```
move $a1, $t1  
lw $a0, dword_35A6C  
jal sub_2DAB8  
addiu $a1, $a0, 4  
beqz $v0, loc_2DA44  
move $v0, $a1  
la $t1, dword_35A6C  
lw $t0, 0($t1)  
subu $t1, $t0, $t1  
sra $t3, $t2, 2  
sll $t4, $t3, 2  
addu $t5, $v0, $t4  
sw $t5, 0($t1)  
sw $v0, dword_35A6C
```

Invent & Verify





Summary – Part I

```

addiu $sp, $ra, -4
sw $ra, 0($sp)
sw $a0, 4($sp)
lui $t1, 3
jal sub_2DAB8
lw $a0, dword_35A6C
lui $t1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllw $t1, $v0, $t8
beqz $t1, loc_2DA24
nop
sub $t1, $t1, 1

```

- A significant share of the Internet, governmental and corporate networks runs on:
 - one out of several tens of thousands of builds
 - of more or less the same code base
 - in a single process environment

... and we cannot bypass it, even if we could tell that it's compromised

```

move $a1, $a0, dword_35A6C
jal sub_2DAD4
addiu $a1, $v0, 0x10
beqz $v0, loc_2DA44
move $v0, $0
la $t1, dword_35A70
lw $t2, $t1, 0
lw $t3, $t1, 4
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

Next question: **How can we even tell?**

Invent & Verify





Error Handling and Recovery

- The software architecture of IOS dictates how exception handling has to be done
 - Remember, IOS is like a large UNIX process
 - What happens when a UNIX process segfaults?
- Upon an exception, IOS can only restart the entire system
 - Even on-board, scheduled diagnostic processes can only forcefully crash the system

```
move $a0, $t7
lw $a0, dword_35A6C
jal sub_35A68
addiu $a1, $v0, 0x10
beqz $v0, loc_2DA44
move $v0, $0
la $t1, dword_35A70
li $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

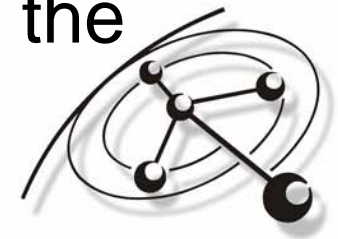
Invent & Verify



Crash Cause Evidence

- Reboot is a clean recovery method
- Reboot destroys all volatile evidence of the crash cause
 - Everything on the router is volatile!
 - Exception: startup configuration and IOS image
- Later IOS releases write an information file called “crashinfo”
 - Crashinfo contains very little information
 - Contents depend on what IOS thought was the cause of the crash

Invent & Verify



```
move $a1, $v0
lw $a0, dword_35A6C
jal sub_2DAD4
addiu $a1, $v0, 4
beqz $v0, $0
move $v0, $0
la $t1, dword_35A70
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

```
addiu $ep, $ra, 4
sw $ra, 4($ep)
sw $a0, $a0
lwi $t1, 2
sub_2DAB8 $t1, $t1, 3
lw $a0, dword_35A6C
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllw $t1, $v0, $t8
beqz $t1, 1cc_2DA24
nop
sub_2DAB8
```


Runtime Evidence

- Crashinfo is only written upon device crashes
- Successful attacks don't cause device crashes
- The available methods are:
 - Show commands
 - Debug commands
 - SNMP monitoring
 - Syslog monitoring

```
move $a0, $v0
lw $a0, dword_35A6C
jal sub_2DAB8
addiu $a1, $v0
beqzl $v0, loc_2D624
move $v0, $0
la $t1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

```
addiu $sp, $ra, 4
sw $ra, 0($sp)
sw $a0, 4($sp)
lui $t1, 3
jal sub_2DAB8
lw $a0, dword_35A6C
lui $t1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllr $t1, $v0, $t8
beqzl $t1, loc_2D624
```



Show Commands

```

addiu $sp, $ra, -4
sw $ra, 0($sp)
sw $a0, 4($sp)
lui $t1, 3
jal sub_2DAB8
lw $a0, dword_35A6C
lui $t1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sltu $t1, $t0, $t8
beqz $t1, loop_35A74

```

- IOS offers a plethora of inspection commands known as the “show” commands
 - Requires access to the command line interface
- Geared towards network engineers
- Thousands of different options and versions
- Almost no access to code
- 12.4 even limits memory show commands

```

move $a1, $v0
lw $a0, dword_35A6C
jal sub_2DAD4
addiu $a1, $v0, 0
beqz $v0, loop_35A74
move $v0, $0
la $t1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

Invent & Verify



Debug Commands

```
addiu $sp, $ra, -4  
sw $ra, 0($sp)  
sw $a0, 4($sp)  
lui $t1, 3  
jal sub_2DAB8  
lw $a0, dword_35A6C  
lui $t1, 3  
lw $t7, dword_35A6C  
lw $t6, dword_35A70  
subu $t8, $t6, $t7  
addiu $t2, $t6, 4  
stwu $t1, 0($t8)  
beqz $t1, loc_2DA24  
nop  
sub $t1, $t1, 1
```

- “debug” enables in-code debugging output
- Debug output has scheduler precedence
 - Too much debug output halts the router
 - Not an option in production environments
- Enabling the right debug output is an art
 - Turn on the wrong ones and you see very little
 - Turn on too many and the router stops working
- Commands depend on the IOS version
- For debug commands to be useful, you have to know what you are looking for **before it happens**
 - Not very useful for security analysis

```
move $a0, $t0  
lw $a0, 0($a0)  
jal sub_2DAD4  
addiu $a0, $a0, 0x10  
beqz $v0, loc_2DA24  
move $v0, $t0  
la $t1, 0x10000000  
lw $t0, 0($t1)  
lw $t0, 0($t1)  
subu $t2, $t0, $t0  
sra $t3, $t2, 2  
sll $t4, $t3, 2  
addu $t5, $v0, $t4  
sw $t5, 0($t1)  
sw $v0, dword_35A6C
```

Invent & Verify



SNMP and Syslog Monitoring

- Commonly accepted method for monitoring networking equipment
- SNMP depending on the implemented MIB
 - Geared towards networking functionality
 - Very little process related information
- Syslog is about as useful for security monitoring on IOS as it is on UNIX systems
- Both generate continuous network traffic
- Both consume system resources on the router
- Then again, someone has to read the logs.

Invent & Verify



```
move $a0, $v0
lw $a1, 0($a0)
jal sub_2DAD4
addiu $a0, $v0, 0x10
beqz $v0, 0($a0)
move $v0, 10
la $t0, dword_35A70
lw $t1, 0($t0)
lw $t2, 4($t0)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

```
addiu $sp, $ra, -4
sw $ra, 0($sp)
sw $a0, 4($sp)
lwi $t1, 2($sp)
jal sub_2DAD8
lwi $a0, dword_35A6C
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllr $t1, $v0, $t2
beqz $t1, 0($t1)
sub $t1, $t1, 1
```

Summary – Part II

```
addiu $sp, $ra, -4  
sw $ra, 0($sp)  
sw $a0, 4($sp)  
lui $t1, 3  
jal sub_2DAB8  
lw $a0, dword_35A6C  
lui $t1, 3  
lw $t7, dword_35A6C  
lw $t6, dword_35A70  
subu $t8, $t6, $t7  
addiu $t2, $t6, 4  
sltu $t1, $v0, $t8  
bne $t1, $t2, 0x00000000  
nop
```

- Identifying compromised routers using today's tools and methods is hard, if not impossible.
- There is not enough data to perform any post mortem analysis of router crashes, security related or not.
- We cannot distinguish between a functional problem, an attempted attack and a successful attack on infrastructure running IOS.

```
move $a0, $t7  
lw $a0, dword_35A6C  
jal sub_2DAB8  
addiu $a0, $v0, 1  
beqz $v0, loc_2DA44  
move $v0, $v0  
la $t1, dword_35A6C  
lw $t1, dword_35A6C  
lw $t0, 0($t1)  
subu $t2, $t0, $t1  
sra $t3, $t2, 2  
sll $t4, $t3, 2  
addu $t5, $v0, $t4  
sw $t5, 0($t1)  
sw $v0, dword_35A6C
```

Invent & Verify



A (not so) New Approach

- We need the maximum amount of evidence
 - A full snapshot of the device is just enough
- We don't need it continuously
 - We need it on-demand
 - We need it when the device crashes
- We need an independent and solid analysis framework to process the evidence
 - We need to be able to extend and adjust it

```
move $a0, $v0
lw $a0, dword_35A6C
jal $a0
addiu $a0, $v0, 0
beqz $v0, Toc_2DA44
move $v0, $0
la $t1, word_35A68
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

```
addiu $sp, $sp, -4
sw $ra, 4($sp)
sw $a0, 8($sp)
lui $t1, 0
lui $t1, 0DAB8
sw $t1, dword_35A6C
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t9, $t6, 4
sllr $t1, $v0, $t9
lui $t1, 0CDA24
```

Invent & Verify



Getting the Evidence

- Cisco IOS can write complete core dumps
 - Memory dump of the main memory
 - Memory dump of the IO memory
 - Memory dump of the PCI memory (if applicable)
- Core dumps are written in two cases
 - The device crashes
 - The user issues the “write core” command

```
move $a0, $t7
lw $a0, dword_35A6C
jal sub_2DAB8
addiu $a1, $v0, 2
beqz $v0, loc_2DA44
move $v0, $0
la $t1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

```
addiu $sp, $ra, -4
sw $ra, 4($sp)
sw $a0, 8($sp)
lw $t1, 4($sp)
jal sub_2DAB8
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllr $t1, $v0, $t2
beqz $t1, loc_2DA24
```



Core Dump Destinations

- IOS supports various destinations
 - TFTP server (bug!)
 - FTP server
 - RCP server
 - Flash file system (later IOS releases)
- Core dumps are enabled by configuration
 - Configuration commands do not differ between IOS versions
 - Configuration change has no effect on the router's operation or performance

```
move $a0, $r0
lw $a0, dword_35A6C
jal sub_2DAD0
addiu $a1, $v0, 0
beqz $v0, loc_2DA44
move $v0, $0
la $t1, word_35A6C
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, 2
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

```
addiu $ep, $ra, 4
sw $ra, 0($ep)
sw $a0, 4($ep)
lwf $t1, 2($ep)
sub $t1, $t1, 2DAB8
lw $t1, dword_35A6C
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllw $t1, $t1, $t8
beqz $t1, loc_2DA24
nop
sub $t1, $t1, 4
```

Invent & Verify





Core Dump Enabled Infrastructure

- Configure all IOS devices to dump core onto one or more centrally located FTP servers
 - Minimizes required monitoring of devices: A router crashed if you find a core dump on the FTP server
 - Preserves evidence
 - Allows crash correlation between different routers
- Why wasn't it used before?
 - Core dumps were useless, except for Cisco developers and exploit writers.



Invent & Verify



Analyzing Core Dumps

Disclaimer:

- Any of the following methods can be implemented in whatever your preferred programming language is.
- This presentation will be centric to our implementation: Recurity Labs CIR.

```

addiu $sp, $ra, -4
sw $ra, 0($sp)
sw $a0, 4($sp)
lui $t1, 2
fe7 sub_2DAB8
lw $a0, dword_35A6C
sw $t1, 8($sp)
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllr $t1, $v0, $t8
beqz $t1, loc_2DA24
nop
sub_2DAB8

```

```

move $a0, $t7
lw $a0, dword_35A6C
jal sub_2DAD4
addiu $a1, $v0, 0x10
beqz $v0, loc_2DA44
move $v0, $0
la $t1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

Invent & Verify



Core Dump Analyzer Requirements

```

addiu $sp, $ra, -4
sw $a0, 0($sp)
lwr $t1, 0($sp)
jal sub_2DAB8
lwr $a0, dword_35A6C
lwr $t1, 3
lwr $t7, dword_35A6C
lwr $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
stwu $t1, 0($t2)
beqz $t1, loc_2DA24
nop
sub $t1, $t1, 4

```

- Must be 100% independent
 - No Cisco code
 - No disassembly based analysis
- Must gradually recover abstraction
 - No assumptions about anything
 - Ability to cope with massively corrupted data
- Should not be exploitable itself
 - Preferably not written in C

```

move $a0, $t7
lwr $a0, dword_35A6C
jal sub_2DAB8
addiu $a0, 0
beqz $v0, loc_2DA44
move $v0, $0
la $t1, dword_35A70
lwr $t1, 0($t1)
lwr $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

Invent & Verify



The Image Blueprint

- The IOS image (ELF file) contains all required information about the memory mapping on the router.
 - The image serves as the memory layout blueprint, to be applied to the core files
 - We wish it were as easy as it sounds
- Using a known-to-be-good image also allows verification of the code and read-only data segments
 - Now we can easily and reliably detect runtime patched images

```

move $a0, $t7
lw $a1, 0($a0)
jal sub_20A74
addiu $a1, $v0, 0x10
beqz $v0, 7($a1)
move $v0, 10($a1)
la $t1, dword_35A70
lw $t1, dword_35A70
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

```

addiu $ep, $ra, 4
sw $a0, 0($ep)
sw $a0, 4($ep)
lw $t1, 0($ep)
fa $t1, 2
sub_2DAB8 $a0, dword_35A6C
lw $t1, 0($ep)
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sll $t1, $v0, $t8
lw $t1, 0($t1)
sub_20A24 $t1, 0($t1)

```

Invent & Verify



Heap Reconstruction

```

addiu $sp, $ra, -4
sw $a0, 0($sp)
sw $a0, 4($sp)
lui $t1, 2
jal sub_2DAB8
$ra, dword_35A6C
$1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllr $t1, $v0, $t8
beqz $t1, loc_2DA24
nop
sub $t1, $t1, $t2

```

- IOS uses one large heap
- The IOS heap contains plenty of meta-data for debugging purposes
 - 40 bytes overhead per heap block in IOS up to 12.3
 - 48 bytes overhead per heap block in IOS 12.4
- Reconstructing the entire heap allows extensive integrity and validity checks
 - Exceeding by far the on-board checks IOS performs during runtime
 - Showing a number of things that would have liked to stay hidden in the shadows ☹️

```

move $a0, $v0
lw $a0, dword_35A6C
jal sub_2DAB8
addiu $a1, $v0, 4
beqz $v0, loc_2DA44
move $v0, $0
la $t1, dword_35A6C
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

Invent & Verify



Heap Verification

```

addiu $sp, $ra, -4
sw $a0, 0($sp)
sw $a0, 4($sp)
lui $t1, 3
jal sub_2DAB8
lw $a0, dword_35A6C
lui $t1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllr $t1, $t6, $t8
beqz $t1, loc_2DA24
nop
sub $t1, $t1, 1

```

- Full functionality of “CheckHeaps”
 - Verify the integrity of the allocated and free heap block doubly linked lists
- Find holes in addressable heap
 - Invisible to CheckHeaps
- Identify heap overflow footprints
 - Values not verified by CheckHeaps
 - Heuristics on rarely used fields
- Map heap blocks to referencing processes
- Identify formerly allocated heap blocks
 - Catches memory usage peaks from the recent past

```

move $a0, $t1
lw $a0, dword_35A6C
jal sub_2DAD4
addiu $a0, $a0, 1
beqz $a0, loc_2DA44
move $v0, $0
la $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $v0
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

Invent & Verify



Process List

```
addiu $sp, $ra, -4  
sw $ra, 0($sp)  
sw $a0, 4($sp)  
lui $t1, 2  
jal sub_2DAB8  
lw $a0, dword_35A6C  
lui $t1, 3  
lw $t7, dword_35A6C  
lw $t6, dword_35A70  
subu $t8, $t6, $t7  
addiu $t2, $t6, 4  
sltu $t1, $v0, $t8  
beqz $t1, loc_2DA24  
nop  
sub $t1, $t1, 1
```

- Extraction of the IOS Process List
 - Identify the processes' stack block
 - Create individual, per process back-traces
 - Identify return address overwrites
 - Obtain the processes' scheduling state
 - Obtain the processes' CPU usage history
 - Obtain the processes' CPU context
- Almost any post mortem analysis method known can be applied, given the two reconstructed data structures.

```
move $a0, $t7  
lw $a0, dword_35A6C  
jal sub_2DAB8  
addiu $a0, $a0, 4  
beqz $v0, loc_2DA44  
move $v0, $a0  
la $t1, 0($v0)  
lw $t1, dword_35A6C  
lw $t0, 0($t1)  
subu $t3, $t1, $t0  
sra $t3, $t3, 4  
sll $t4, $t3, 2  
addu $t5, $v0, $t4  
sw $t5, 0($t1)  
sw $v0, dword_35A6C
```

Invent & Verify



TCL Backdoor Detection

- TCL scripting is available on later Cisco IOS versions
- TCL scripts listening on TCP sockets
 - Well known method
 - Used to simplify automated administration
 - Used to silently keep privileged access to routers
 - Known bug:
 - not terminated when the VTY session ends (fixed)
 - Simple TCL backdoor scripts published
- CIR can extract all TCP script chunks from IOS heap and dump them for further analysis
 - There is still some reversing work to do

Invent & Verify



```
move $a0, $v0
lw $a0, dword_35A6C
jal $a0, 0
addiu $a0, $v0, 0
beqz $v0, loc_35A44
move $v0, $v0
la $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, 1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

```
addiu $sp, $ra, -4
sw $ra, $ra, 4
sw $a0, $a0, 4
lwf $t1, 2($t1)
fild $f12, 2DAB8
lwf $t0, dword_35A6C
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllw $t1, $t1, $t2
```


Random Applications

```

addiu $sp, $ra, -4
sw $a0, 0($sp)
sw $a0, 4($sp)
lui $t1, 3
jal sub_2DAB8
$ra, dword_35A6C
lui $t1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllr $t1, $v0, $t8
beqz $t1, loc_2DA24
nop
sub_2DAB8

```

- Find occasional CPU hogs
- Detect Heap fragmentation causes
- Determine what processes where doing
- Finding attacked processes
 - See examples (Semi-DEMO)
- Research tool
 - Pointer correlation becomes really easy
 - Essential in a shared memory environment

```

move $a1, $v0
lw $a0, dword_35A6C
jal sub_2DAD4
addiu $a1, $v0, 0x10
beqz $v0, loc_2DA24
move $v0, $0
la $t1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

Invent & Verify





IOS Packet Forwarding Memory

```

addiu $sp, $ra, -4
sw $a0, 0($sp)
sw $a0, 4($sp)
lui $t1, 3
jal sub_2DAB8
lw $a0, dword_35A6C
lui $t1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllr $t1, $t0, $t8
beqz $t1, loc_2DA24
nop
sub $t1, $t1, 1

```

- IOS performs routing either as:
 - Process switching
 - Fast switching
 - Particle systems
 - Hardware accelerated switching
- Except hardware switching, all use IO memory
 - IO memory is written as separate code dump
 - By default, about 6% of the router's memory is dedicated as IO memory
 - In real world installations, it is common to increase the percentage to speed up forwarding
- Hardware switched packets use PCI memory
 - PCI memory is written as separate core dump

```

move $a0, $t0
lw $a0, dword_35A6C
jal sub_2DAD4
addiu $a1, $v0, 0x31
beqz $v0, loc_2DA24
move $v0, $0
la $t1, dword_35A70
lw $t1, $t1
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

Invent & Verify



IO Memory Buffers

- Routing (switching) **ring buffers** are grouped by packet size
 - Small
 - Medium
 - Big
 - Huge
- Interfaces have their own buffers for locally handled traffic
- IOS tries really hard to not copy packets around in memory
- New traffic does not automatically erase older traffic in a linear way

Invent & Verify



```
addiu $sp, $ra, -4
sw $a0, 0($sp)
lui $t1, 3
jal sub_2DAB8
lw $a0, dword_35A6C
lui $t1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllv $t1, $t0, $t8
lui $t1, 3
sub $t1, $t1, $t2
```



```
move $a0, $t1
lw $a0, 0($a0)
jal sub_2DAB8
addiu $a0, $a0, 16
beqz $v0, loc_21A444
move $v0, $a0
la $t1, dword_35A6C
lw $t1, 0($t1)
lw $t1, 0($t1)
subu $t1, $t1, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```



Traffic Extraction

```

addiu $sp, $ra, -4
sw $a0, 0($sp)
lui $t1, 3
jal sub_2DAB8
lw $a0, dword_35A6C
lui $t1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sltu $t1, $v0, $t8

```

- CIR dumps packets that were process switched by the router from IO memory into a PCAP file
 - Traffic addressed to and from the router itself
 - Traffic that was process switching inspected
 - Access List matching
 - QoS routed traffic
- CIR could dump packets that were forwarded through the router too

- Reconstruction of packet fragments possible
- Is it desirable?

```

move $a1, $v0
lw $a0, 0($a1)
jal sub_2DAD4
addiu $a1, $v0, 0x10
beqzl $v0, 10
move $a1, $v0
la $t1, dword_35A70
lw $t1, 0($t1)
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

Invent & Verify



Advanced Traffic Extraction

- Writing core to a remote server uses IO memory
 - Overwrites part of the traffic evidence
- CIR can use a GDB link instead of a core dump
 - Serial GDB protocol allows direct access to router memory via the console
 - Uses Zynamics GDB debug link
- Disconnecting all network interfaces preserves IO and PCI memory contents
 - Using GDB halts the router
- All data is preserved – useful for emergency inspections

```
move $a0, dword_35A6C
lw $a0, dword_35A6C
jal sub_2DA44
addiu $a1, $v0
beqz $v0, loc_2DA44
move $v0, $v0
la $t1, dword_35A6C
lw $t1, dword_35A6C
lw $t1, dword_35A6C
subu $t3, $t2, 2
sra $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

```
addiu $ep, $ra, 4
sw $ra, 4($ep)
sw $a0, 8($ep)
lwf $t1, 4($ep)
lwf $t2, 8($ep)
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t9, $t8, 4
sllw $t9, $t9, 2
lwf $t9, 4($t9)
sub $t9, $t9, $t8
```



Traffic Extraction Applications

```

addiu $sp, $ra, -4
sw $ra, 0($sp)
sw $a0, 4($sp)
lui $t1, 2
jal $t1, 20A58
addiu $a0, $ra, 35A6C
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
stwu $t1, $t0, $t2
beqz $t1, 100_20A24
sub $t1, $t1, $t2

```

- Identification of attack jump pad routers
- 0day identification against systems on segmented network interfaces
 - If you got the packet, you got the 0day
- Spoofing attack backtracking
 - One hop at the time, obviously
- LE detection

```

move $a0, $t7
lw $a0, dword_35A6C
jal $t1, 20A58
addiu $a0, $ra, 35A6C
beqz $t1, 100_20A24
move $v0, $0
la $t1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

Invent & Verify





Reality Check: March's Vulnerabilities

```

addiu $sp, $ra, -4
sw $a0, 0($sp)
lui $t1, 3
jal sub_2DAB8
lw $a0, dword_35A6C
lui $t1, 3
st7, dword_35A6C
st6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
st7, $t2, $t8
sub $t2, $t2, 4

```

- “Cisco IOS Virtual Private Dial-up Network Denial of Service Vulnerability”
 - Memory exhaustion / leak
 - Visible by heap usage analysis
- “Cisco IOS User Datagram Protocol Delivery Issue For IPv4/IPv6 Dual-stack Routers”
 - “The show interfaces command can be used to view the input queue size to identify a blocked input interface.”
 - CIR could output all the packets that are still in the queue, even allowing source identification
- “Vulnerability in Cisco IOS with OSPF, MPLS VPN, and Supervisor 32, Supervisor 720, or Route Switch Processor 720”
 - see above

```

move $a0, $v0
lw $a0, 0($a0)
jal sub_2DAB8
addiu $a0, $a0, 4
beqz $v0, $a0, $a0
move $v0, $a0
la $t1, dword_35A70
lw $t1, 0($t1)
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

Invent & Verify





Challenges

- The analysis framework has to handle the complexity of the Cisco IOS landscape
 - Hardware platforms
 - Image versions
 - Any-to-Any relation!
- CIR is currently IOS feature set independent
- CIR successfully tested against IOS 12.1 – 12.4
- Official support starts with:
 - Cisco 2600
- Internal testing already covers:
 - Cisco 1700
 - Cisco 2691
 - Cisco 6200
- The platform is the major source of work, testing and verification

```

addiu $sp, $ra, -4
sw $ra, 0($sp)
sw $a0, 4($sp)
lui $t1, 2
jal sub_2DAB8
lw $a0, dword_35A6C
lui $t1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
lui $t0, 2
sw $t0, 0($t2)
subu $t2, $t0, $t1

```

```

move $a0, $t7
lw $a0, dword_35A70
jal sub_2DAD4
addiu $a1, $v0, 4
beqz $v0, Toc_2DAD4
move $t1, $v0
la $t1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

Invent & Verify



Summary – Part III

```

addiu $sp, $ra, 4
sw $ra, 0($sp)
sw $a0, 4($sp)
lui $t1, 2
jal sub_2DAB8
lw $a0, dword_35A6C
lui $t1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllw $t1, $t0, $t8
sub

```

- Writing core dumps is a viable method for obtaining IOS evidence when it is needed.
 - The evidence includes forwarded and received packets.
- An independent analysis framework can distinguish between bugs and attacks, enabling real forensics on IOS routers.
- Recurity Labs' CIR already reliably identifies many types of attacks and IOS backdoors.
 - CIR is work-in-progress
 - CIR's future depends on the feedback we receive from the community.

```

move $a1, $v0
lw $a0, dword_35A6C
jal sub_2DAB8
addiu $a1, $v0
beqz $v0, loc_212A44
move $v0, $0
la $t1, dword_35A6C
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

Invent & Verify





Availability

```

addiu $sp, $sp, -4
sw $ra, 4($sp)
sw $a0, 8($sp)
lui $t1, 2
jal sub_2DAB8
lw $a0, dword_35A6C
lui $t1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllr $t1, $t0, $t8
beqz $t1, loc_2DA24
nop
sub $t1, $t1, 1

```

1. CIR Online Service (free)
2. CIR Rootkit Detector (free)
3. CIR Professional (non-free)

```

move $a0, $t7
lw $a0, dword_35A6C
jal sub_2DAD4
addiu $a1, $v0, 0x10
beqz $v0, loc_2DA44
move $v0, $0
la $t1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

Invent & Verify





CIR Online

```

addiu $sp, $ra, -4
sw $ra, 0($sp)
sw $a0, 4($sp)
lui $t1, 2
jal sub_2DAB8
lw $a0, dword_35A6C
lui $t1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllw $t1, $t0, $t8
subu $t1, $t1, $t2

```

- An analysis framework's quality is directly related to the amount of cases it has seen
 - CIR needs a lot more food to grow up
 - We want to provide it to everyone while constantly developing and improving it
- Free Service: **<http://cir.recurity-labs.com>**
 - Processing on our servers
 - Always using the latest version
 - Right now, CIR Online runs in BETA state

```

move $a0, $t7
lw $a0, dword_35A6C
jal sub_2DAB8
addiu $a1, $v0, 0x10
beqz $v0, $t0, $t1
move $v0, $t0
la $t1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

Invent & Verify



```
addiu $sp, $ra, 4
sw $ra, 0($sp)
sw $a0, 4($sp)
lui $t1, 2
jal sub_2DAB8
lw $a0, dword_35A6C
lui $t1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllr $t1, $v0, $t8
beqz $t1, loc_2DA24
nop
sub $t1, $t1, 1
```



CIR Rootkit Detector

- Detection of image modification
- Detection of runtime code modification
- Support for all access layer platforms
- Freely available at <http://cir.recurity-labs.com>
- Currently in BETA state

```
move $a0, $t7
lw $a0, dword_35A6C
jal sub_2DAD4
addiu $a1, $v0, 0x10
beqz $v0, loc_2DA44
move $v0, $0
la $t1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

Invent & Verify



At the end, it's all up to you!

- We think CIR could be useful
 - For the networking engineer
 - For the forensics professional
 - To finally know the state of our infrastructure
- We know what we can do
- We need advise on where you want this tool to be in the future

```
move $a0, $v0
lw $a0, dword_35A6C
jal $a0, sub_20A68
addiu $a0, $a0, 2
beqz $v0, loc_20A44
move $v0, $0
la $t1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

```
addiu $ep, $ra, 4
sw $ra, 4($ep)
sw $a0, $a0
lw $t1, 2($a0)
jal $t1, sub_20A68
lw $t0, dword_35A6C
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllr $t1, $t0, $t8
beqz $t1, loc_20A24
nop
sub $t1, $t1
```



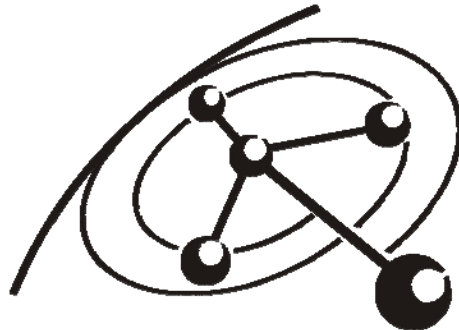


cir.recurity-labs.com

```

addiu $sp, $ra, -4
sw $a0, 0($sp)
lui $1, 3
subu $a0, $a0, $t7
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t2, $t6, 4
sllr $t1, $t0, $t8
beqz $t1, loc_2DA24
nop
sub $t1, $t1, $t2

```



Recurity Labs

Felix 'FX' Lindner
Head

fx@recurity-labs.com

Recurity Labs GmbH, Berlin, Germany
<http://www.recurity-labs.com>

```

move $a0, $t7
lw $a0, dword_35A6C
jal sub_2DAD4
addiu $a1, $v0, 0x10
beqz $v0, loc_2DA44
move $v0, $0
la $t1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

```

Invent & Verify

