

Sleepy Watermark Tracing: An Active Network-based Intrusion Response Framework

Xinyuan Wang[†] Douglas S. Reeves^{†‡} S. Felix Wu^{††} Jim Yuill[†]

[†]Department of Computer Science

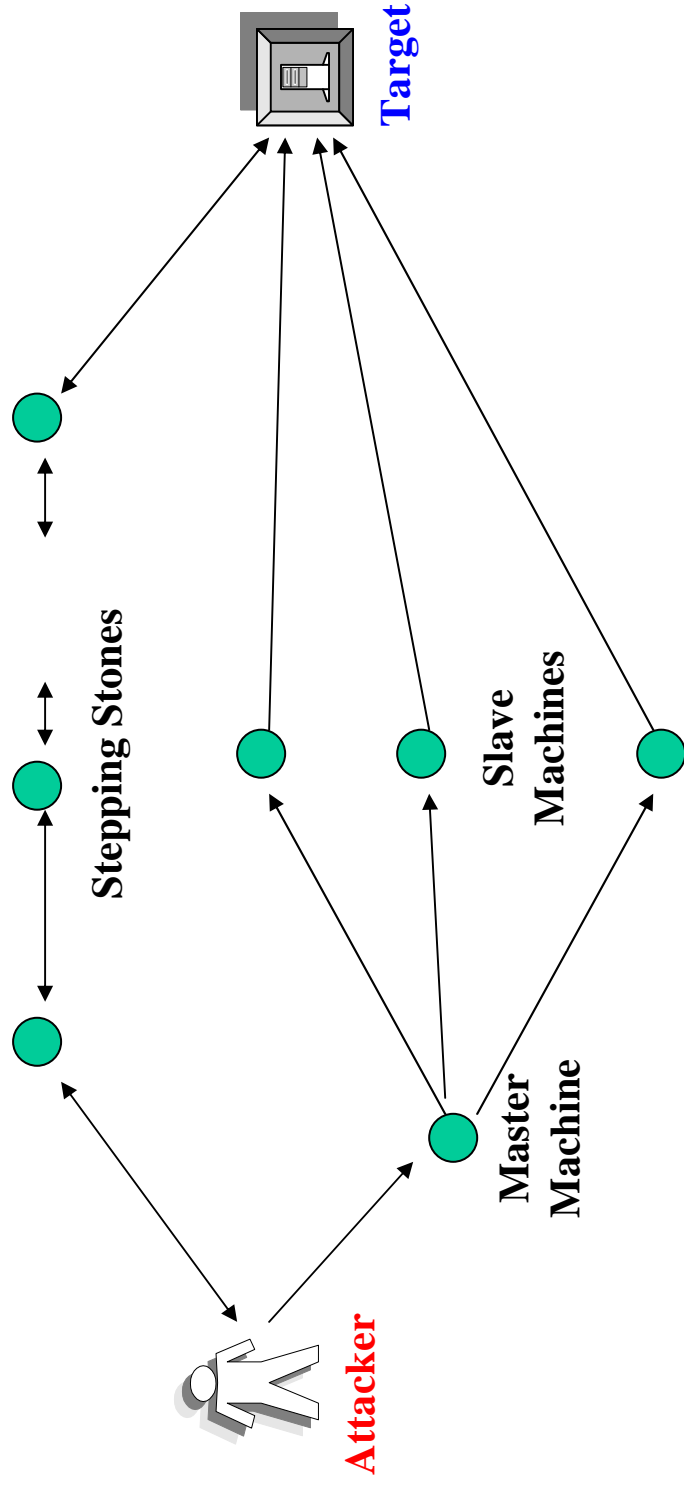
[‡]Department of Electrical and Computer Engineering
North Carolina State University

^{††}Department of Computer Science
University of California at Davis

IFIP/Sec'01 Paris, France

Network-Based Attacks

We have detected attacks from the network !!!



Where do these attacks come from ???

Tracing Problem and Its Challenges

- What is tracing problem ?
 - *To identify the source of network-based intrusion*
- Why tracing is important ?
 - *Network-based attacks can not be effectively repelled or eliminated until its source is known*
- Challenges in tracing
 - *Spoofed source IP address*
 - *Connections through “stepping stones”*
- One of the hardest network security problems
- Focus on tracing *chained connections* with stepping stones

Tracing Approaches

	Passive	Active
Host-based	DIDS CIS	CallerID
Network-based	Thumbpriting Timing-based Deviation-based	IDIP SWT

Classification of Existing Tracing Approaches and **SWT**

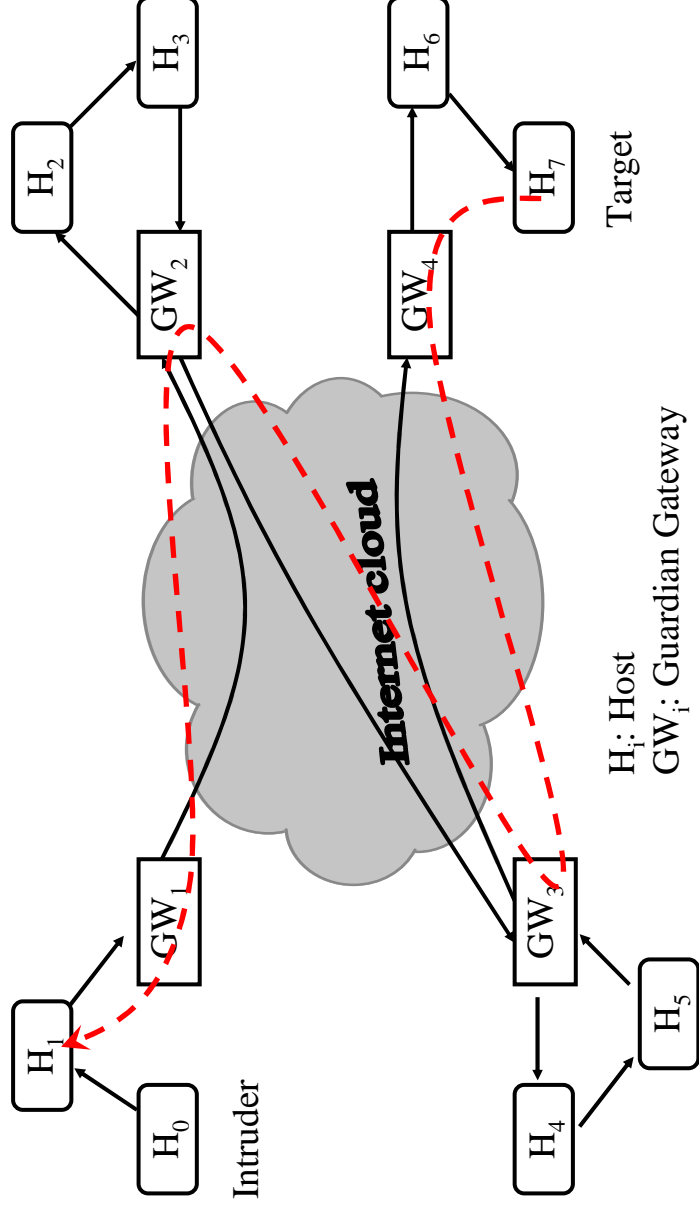
Tracing Approach Classification

- Host-based:
 - *tracing based on information collected from each host*
- Network-based:
 - *tracing based on the property of network connections: the application level content of chained connections is invariant*
- Passive:
 - *passively monitor and compare network traffic, need to compare every concurrent incoming connections with every concurrent outgoing connection. (clueless tracing)*
- Active:
 - *dynamically control what and how connections are to be correlated through customized packet processing. (tracing with clue)*

Sleepy Watermark Tracing (SWT)

- SWT is an active network-based tracing framework
 - *Active network seeks to increase the programmability of networks that enables user and application to dynamically control how packets are handled.*
- SWT is “sleepy” and yet “active”
- SWT exploits following observations
 - *Interactive intrusions with chained connections are bi-directional and symmetric at the granularity of connections*
 - *Application level contents are invariant across connection chains*

SWT Tracing Model

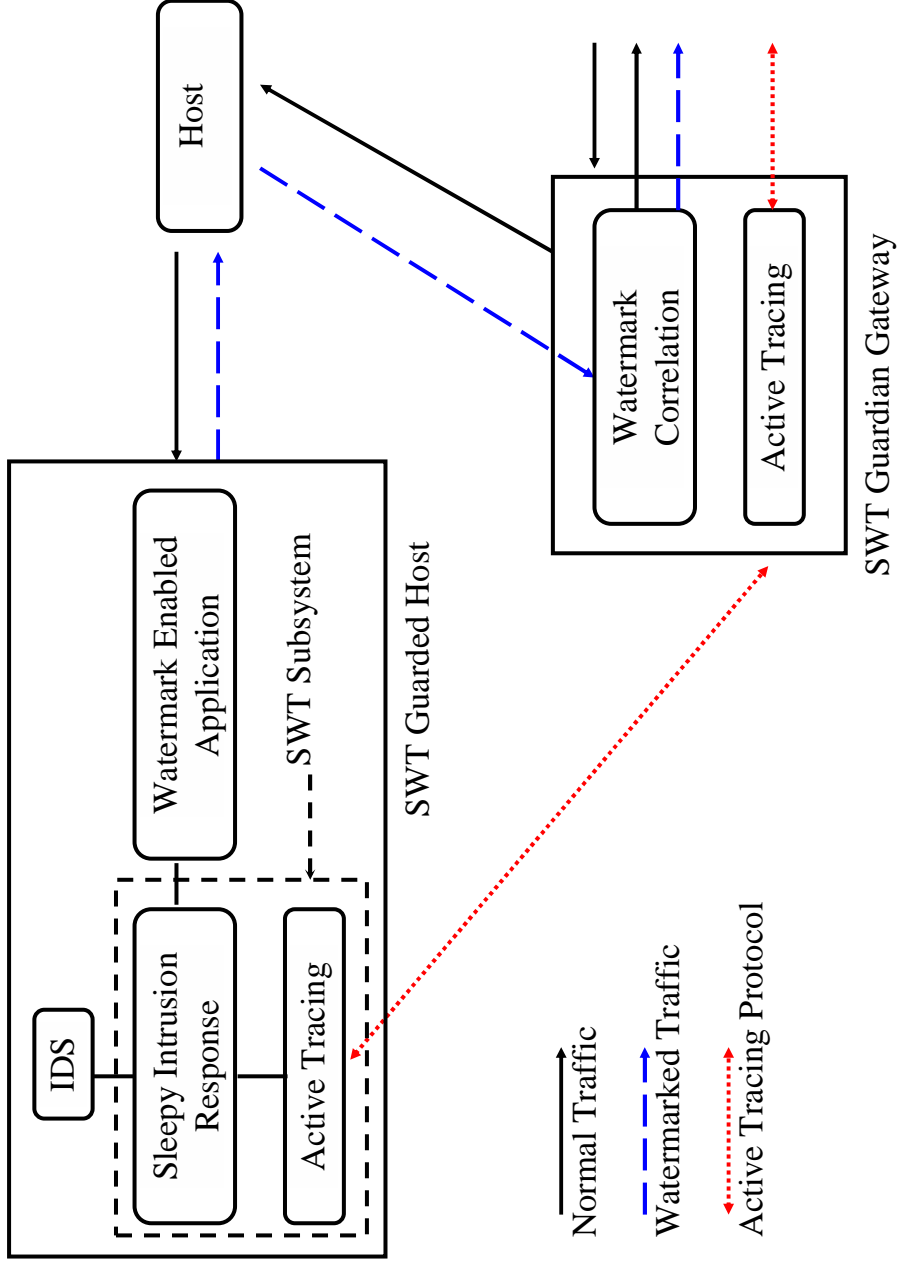


Target injects watermark into the backward connection and “wakes up” guardian gateways along the intrusion path

SWT Concepts and Assumptions

- Basic SWT concepts
 - *Guardian Gateway (nearest router)*
 - *Incoming Guardian Gateway*
 - *Outgoing Guardian Gateway*
 - *Guardian Gateway Set*
 - *Guarded Host*
- Basic SWT assumptions
 - *Intrusions are interactive and bi-directional*
 - *Routers are trust worthy and hosts are not trust worthy*
 - *Each host has a single SWT guardian gateway*
 - *There is no link-to-link encryption*

SWT Architecture



SWT Components

- SWT supporting components
 - *IDS*
 - *Application level interface to any Intrusion Detection System*
 - *Watermark-enabled application*
 - *Server applications that have been modified to be able to “inject” arbitrary watermark at request*
- SWT components
 - *Sleepy Intrusion Response (SIR)*
 - *Controls and coordinates overall SWT intrusion tracing*
 - *Watermark Correlation (WMC)*
 - *Matching adjacent connections through watermark*
 - *Active Tracing (AT)*
 - *“Wakes up” and coordinate SWT guardian gateways*

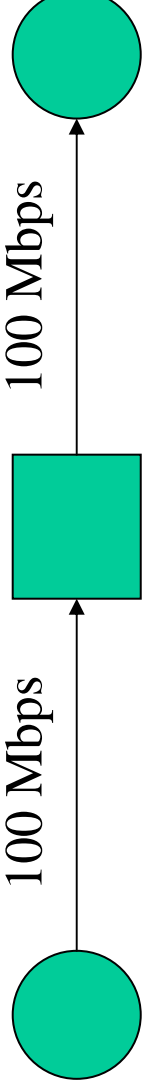
Watermark

- A small piece of information that can be used to uniquely identify a connection
- Application specific
- Invisible to end users (telnet, rlogin etc)
 - *[Identifying part] + [covering part]*
 - “**intruder**”
 - *Original*
 - “**Su**”
 - *[Original] + [watermark]*
 - “**Suintruder**”
- Collision probability

SWT Analysis

- SWT Advantages
 - *Separate intrusion tracing from intrusion detection*
 - *Does not need to record all the concurrent connections*
 - *Requires no clock synchronization*
 - *Trace through connection chain within single keystroke*
 - *Can trace through connection chain even when the intruder is silent*
- Robustness and security
- Efficiency
- Scalability
- Applicability
- Intrusiveness

SWT Performance

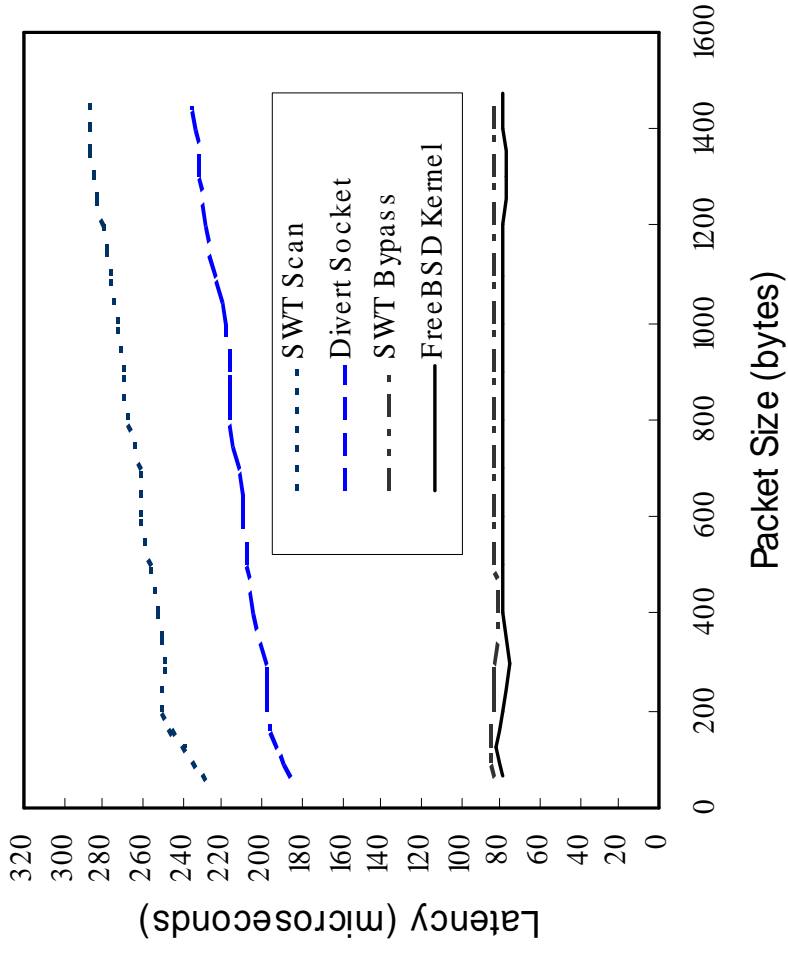


SWT Guardian GW
Pentium 233Mhz
FreeBSD 4.0

Measure latency

- FreeBSD kernel IP forwarding without SWT
- SWT configured to by pass traffic
- Divert socket IP forwarding without SWT
- SWT configured to scan traffic

SWT Latency



Latency overhead due to SWT itself is about 50 μ s

Future Work

- New form of watermark
- Correlate encrypted connection chains (ssh, IPSEC etc)
- More watermark-enabled applications
- Transparent proxy for watermark injection
- Tracing based active intrusion response
 - What can be done once we have identified the intrusion source ?