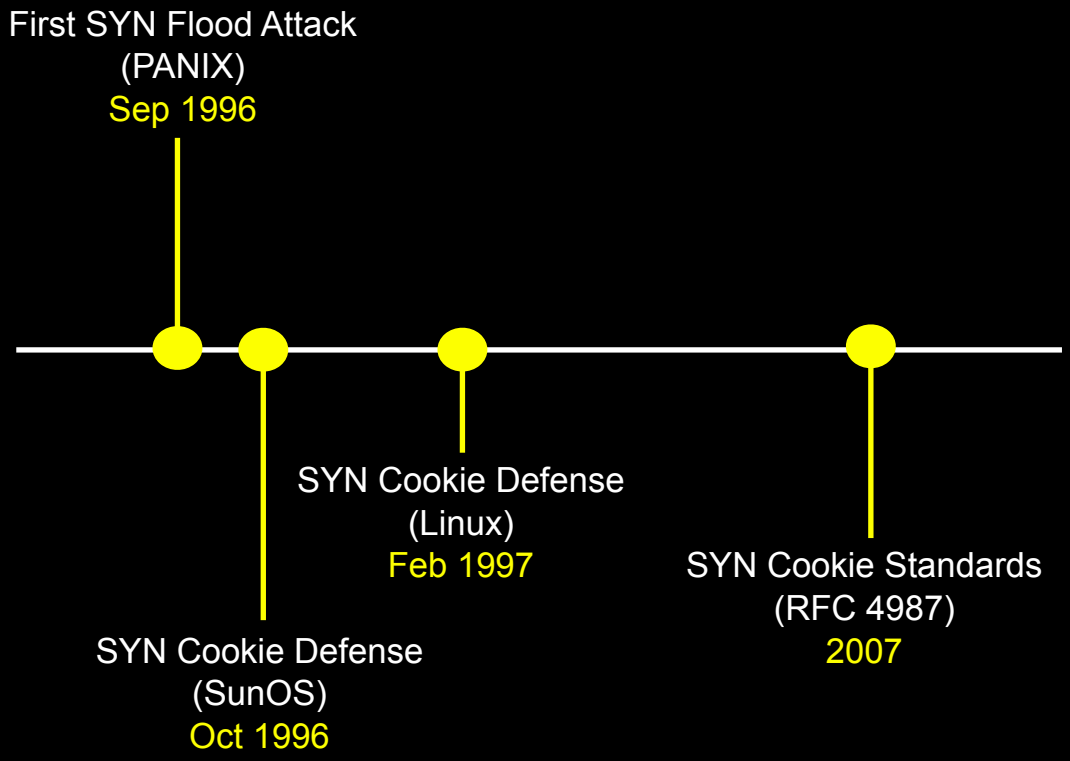
SmartCookie: Blocking Large-Scale SYN Floods with a Split-Proxy Defense on Programmable Data Planes

Sophia Yoo, Xiaoqi Chen, Jennifer Rexford

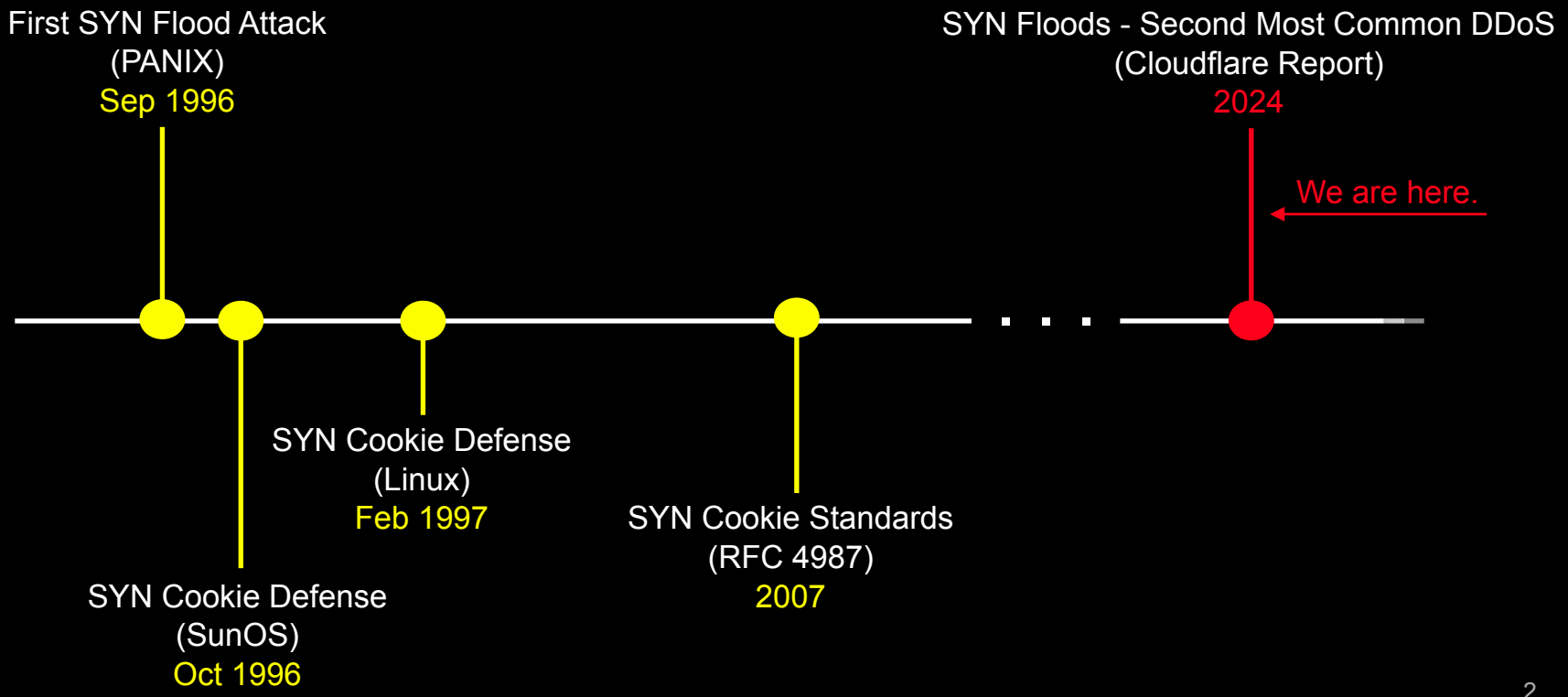
33rd USENIX Security Symposium
August 14, 2024

PRINCETON
UNIVERSITY

A brief timeline of SYN flooding attacks



A brief timeline of SYN flooding attacks



What modern SYN flood defenses really need…

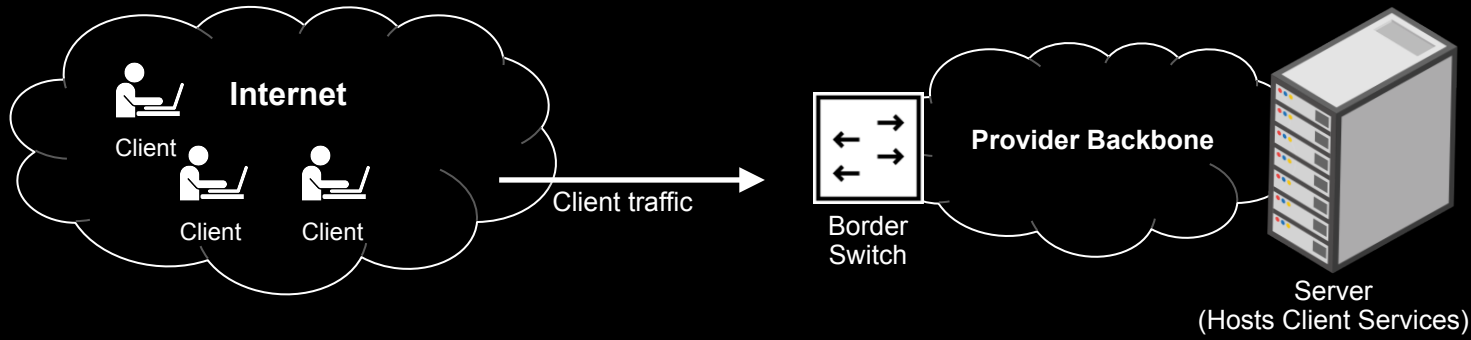
- ⁉ Security
- ⁉ Scalability
- ⁉ Performance

What modern SYN flood defenses really need…

- ⁉ Security …blocking attacks from adaptive adversaries
- ⁉ Scalability …handling large amounts of benign and attack traffic
- ⁉ Performance …maintaining low latency for benign clients

Network providers must performantly serve client traffic

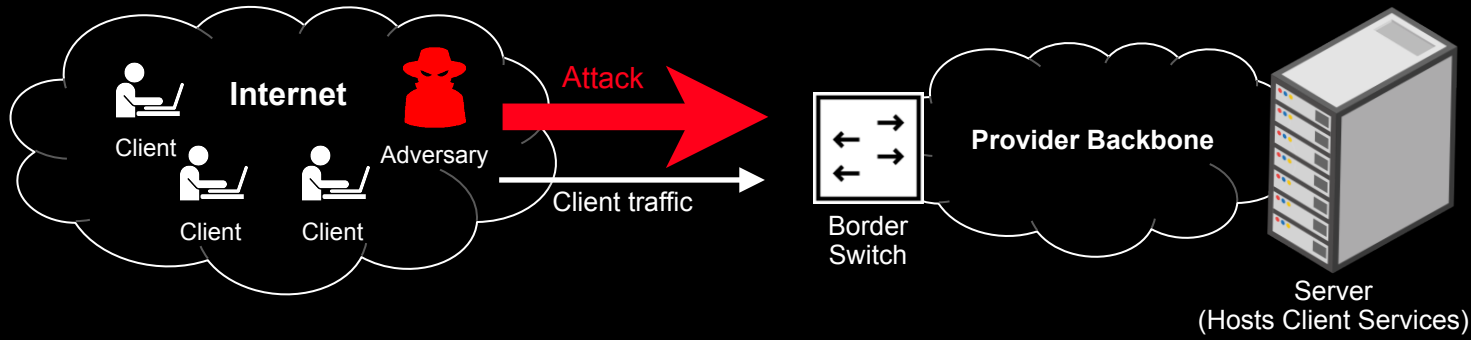
- ⁉ Security …blocking attacks from adaptive adversaries
- ⁉ Scalability …handling large amounts of benign and attack traffic
- ⁉ Performance …maintaining low latency for benign clients

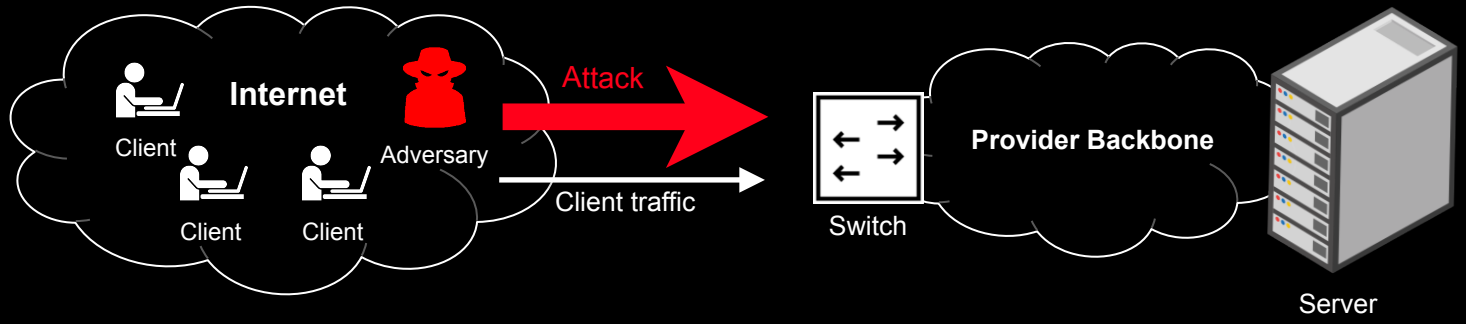


Network providers must performantly serve client traffic

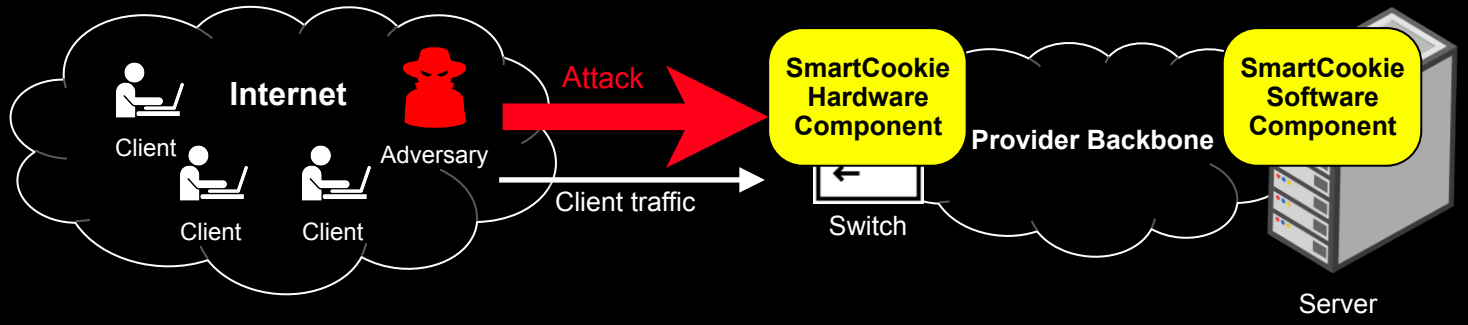
while blocking attack traffic as early as possible in the provider network

- ⁉ Security …blocking attacks from adaptive adversaries
- ⁉ Scalability …handling large amounts of benign and attack traffic
- ⁉ Performance …maintaining low latency for benign clients



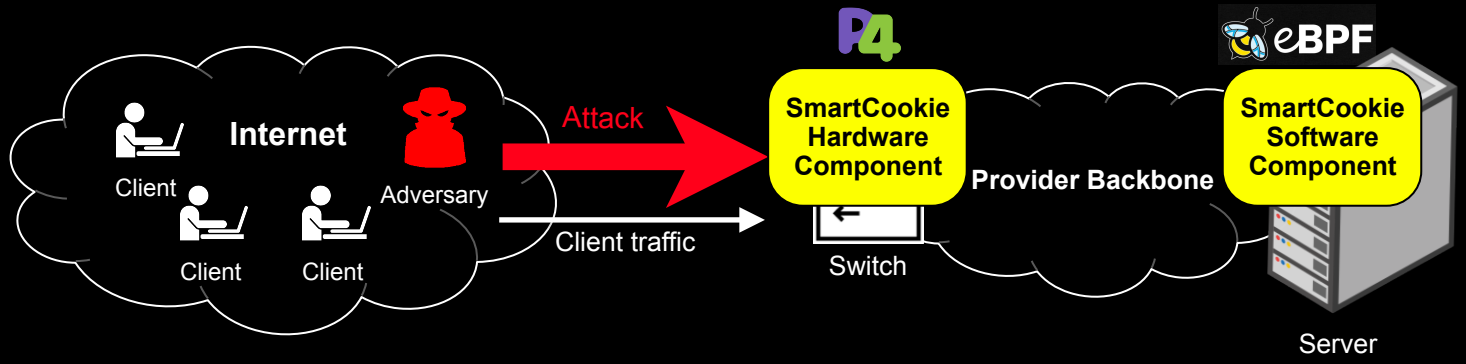

Internet
Client
Client
Client
Adversary
Attack
Client traffic
Switch
Provider Backbone
Server

SmartCookie: a *collaborative*, *split-layer design*



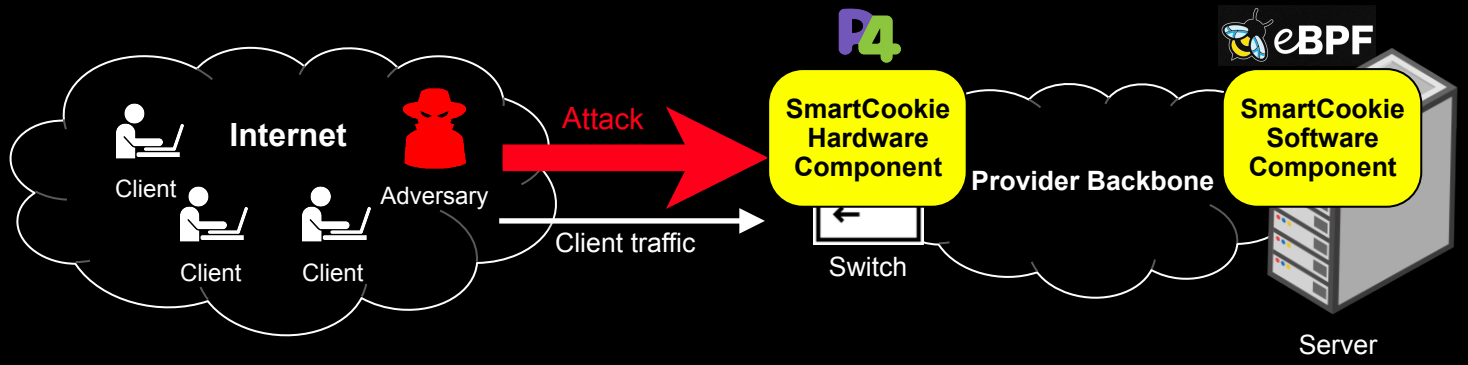
SmartCookie: a *collaborative*, *split-layer design*

…enabled by the power of programmable targets



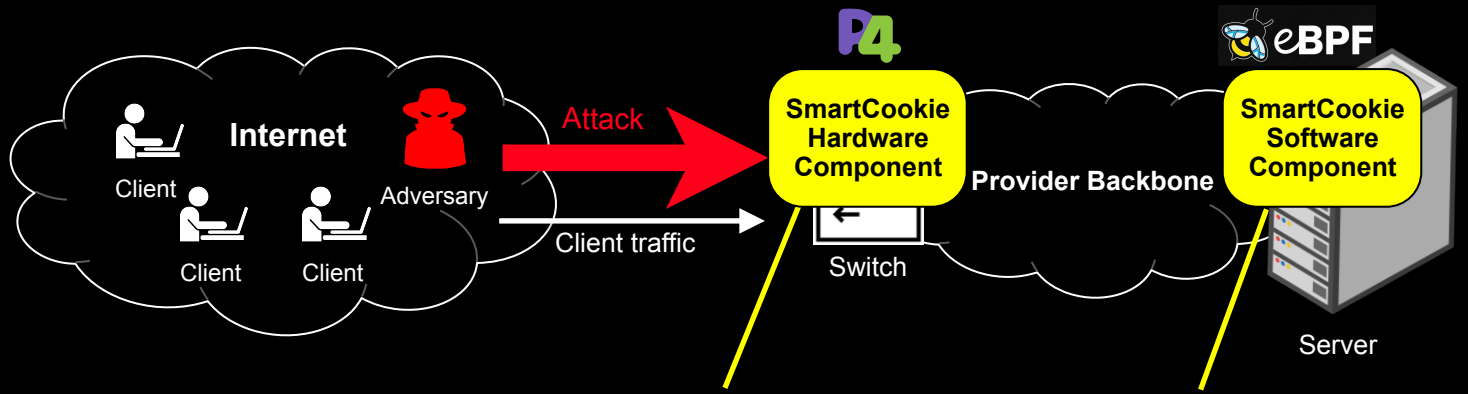
SmartCookie: a *collaborative*, *split-layer design*

where each component handles functionality based on its unique resources



SmartCookie: a *collaborative*, *split-layer design*

where each component handles functionality based on its unique resources



(i) Hardware blocks large bulk of attack traffic
+ passes through all client traffic

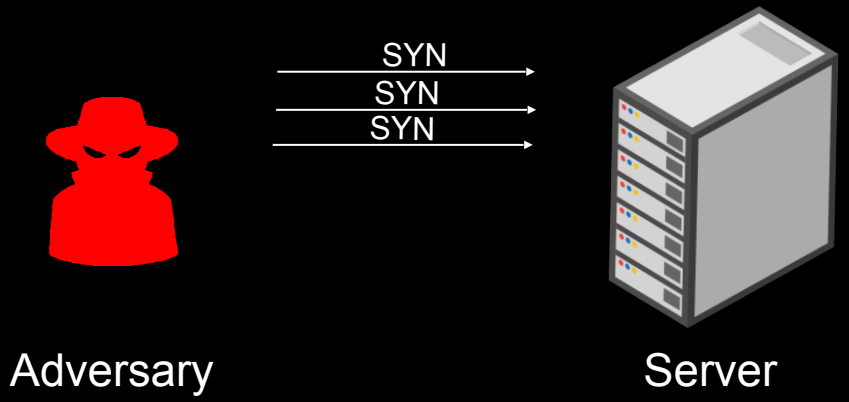
(ii) Software blocks any leftover attack traffic
+ serves all client traffic as normal

SYN Flooding: an asymmetric attack on resources

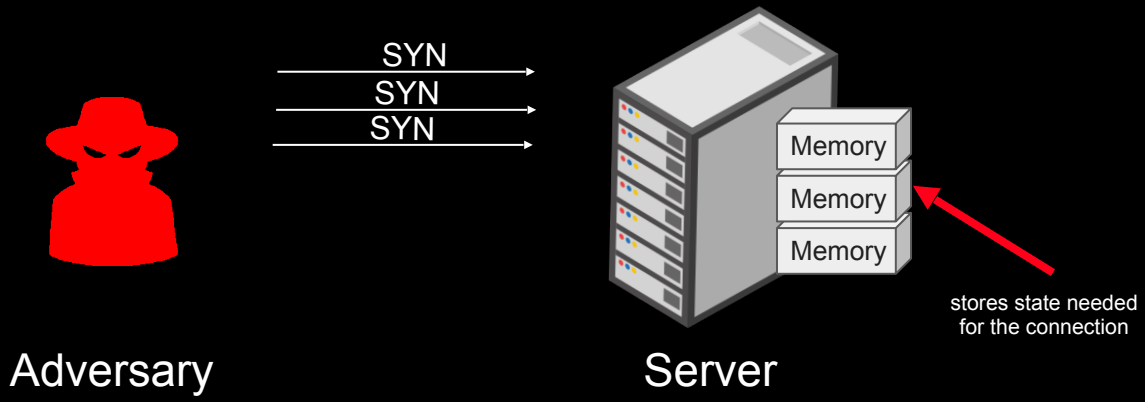
Adversary

Server

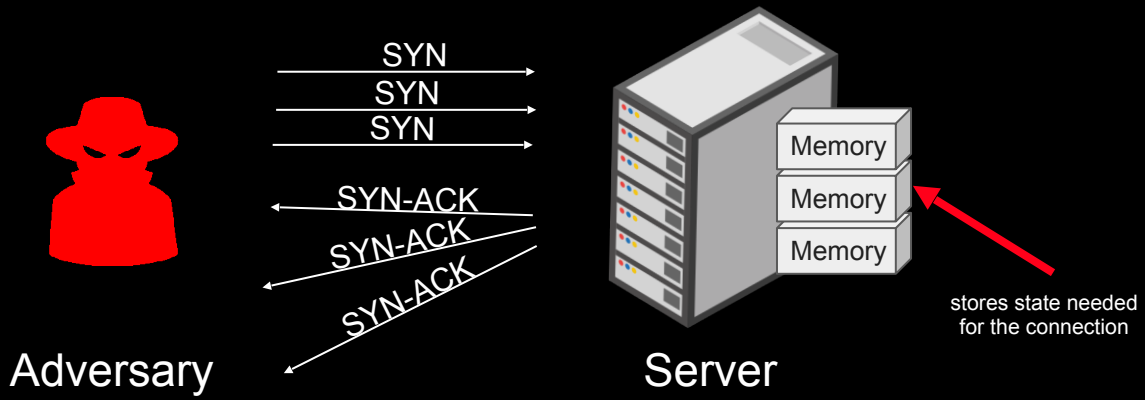
SYN Flooding: an asymmetric attack on resources



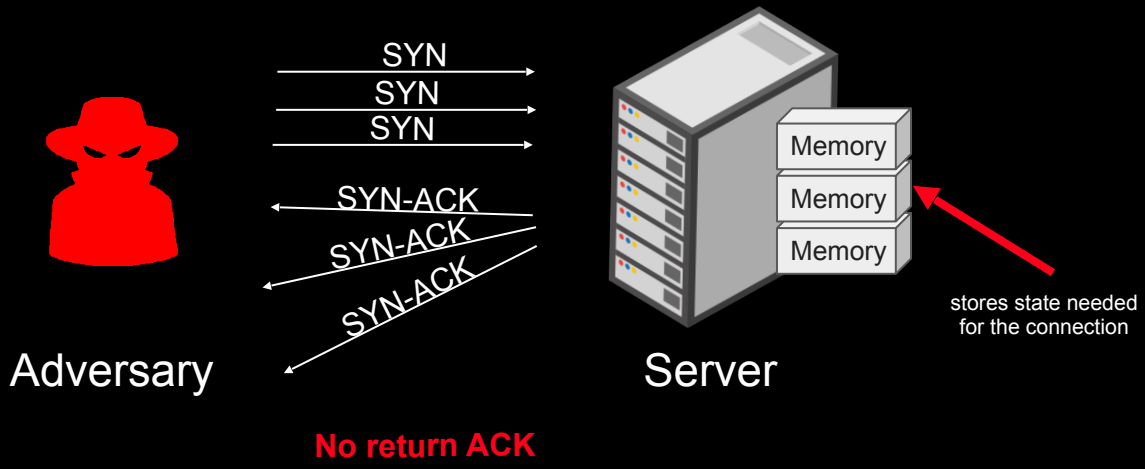
SYN Flooding: an asymmetric attack on resources



SYN Flooding: an asymmetric attack on resources

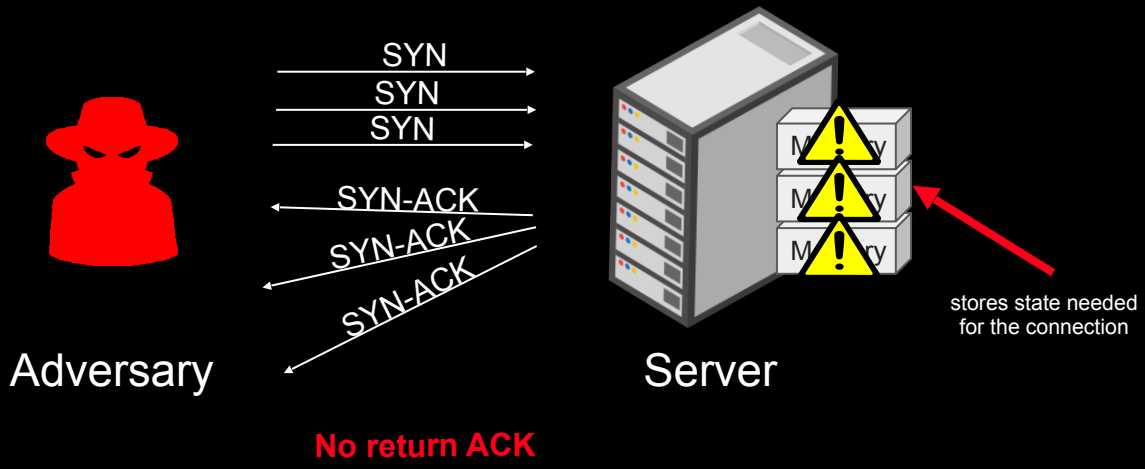


SYN Flooding: an asymmetric attack on resources



SYN Flooding: an asymmetric attack on resources

Server memory is depleted, leading to DoS for new requests



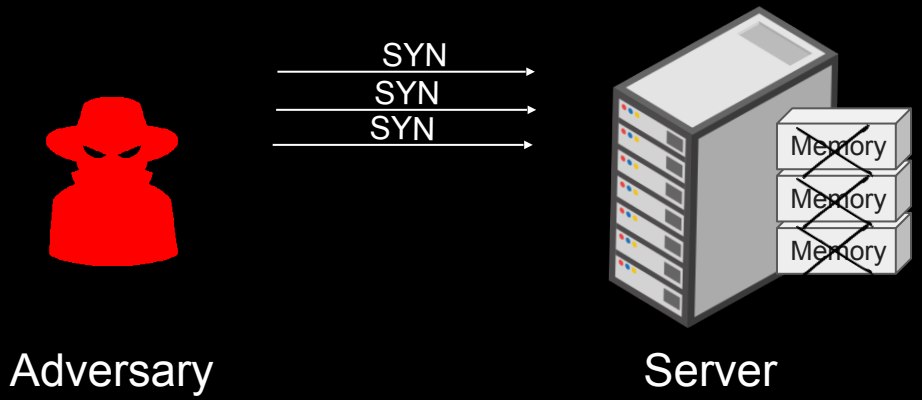
SYN Cookies: a stateless solution

Adversary

Server

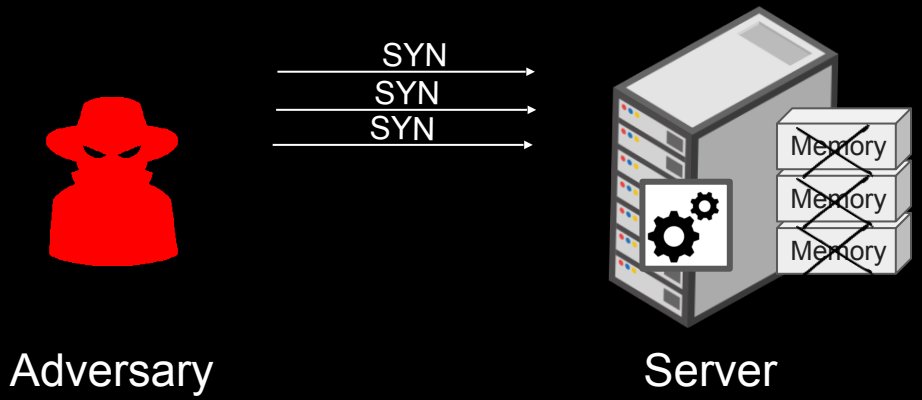
SYN Cookies: a stateless solution

Trading memory…



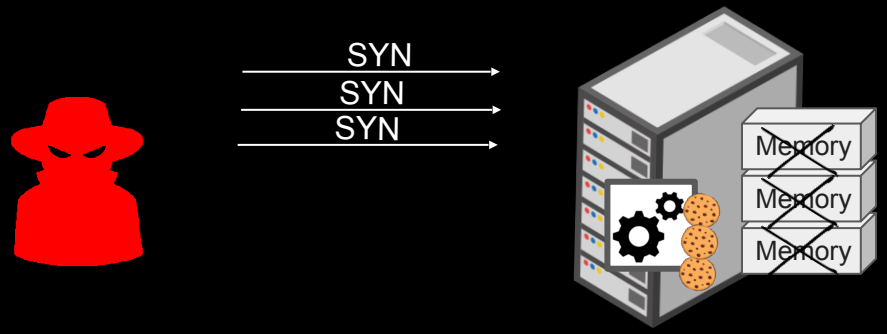
SYN Cookies: a stateless solution

…for compute



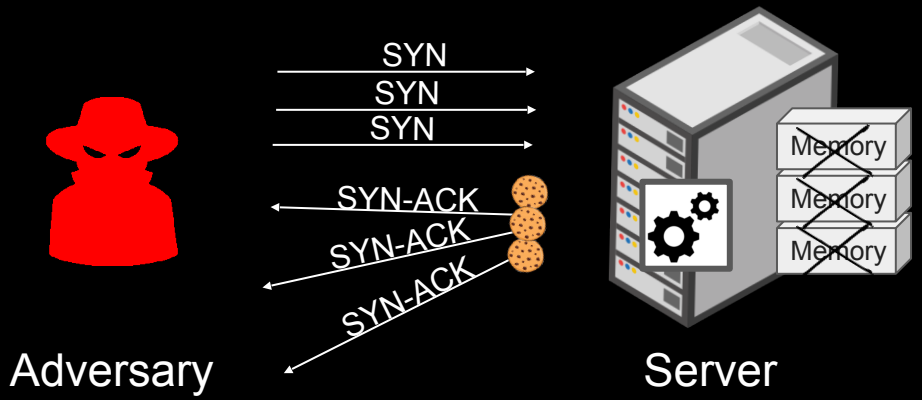
SYN Cookies: a stateless solution

Cryptographically secure “cookie” computed to store the relevant state



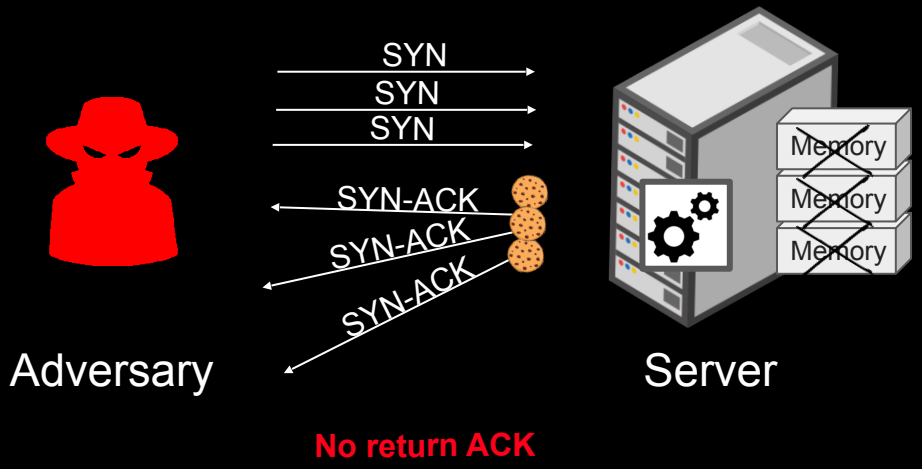
SYN Cookies: a stateless solution

Cryptographically secure “cookie” computed to store the relevant state



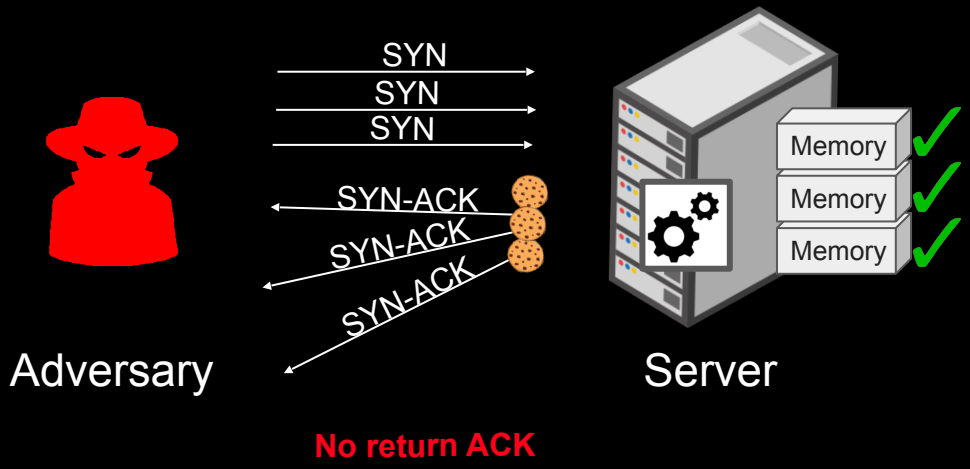
SYN Cookies: a stateless solution

Cryptographically secure “cookie” computed to store the relevant state



SYN Cookies: a stateless solution

Memory protected for legitimate connection requests

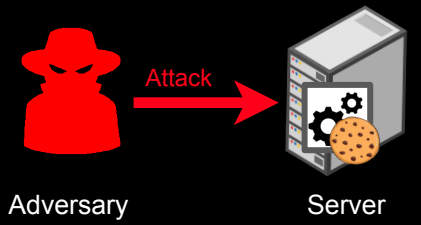


Why is a secure, scalable, and performant defense so hard?

Software-Only Solutions Can’t Scale Compute

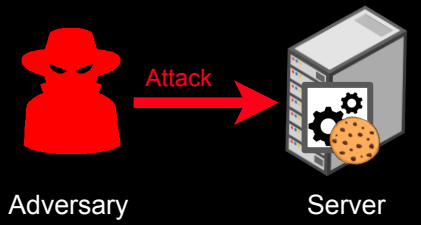
Computational cost to identify attacks leads to early CPU exhaustion.

Cryptographically computing cookies in software is costly



Cryptographically computing cookies in software is costly

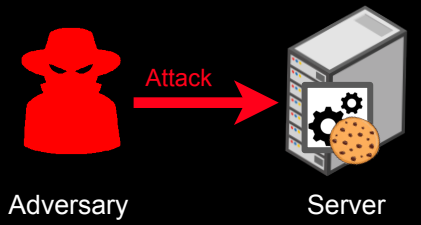
Cryptographic cookie computation and packet processing for every potential connection



Cryptographically computing cookies in software is costly

Cryptographic cookie computation and packet processing for every potential connection

Computational strain becomes new attack vector

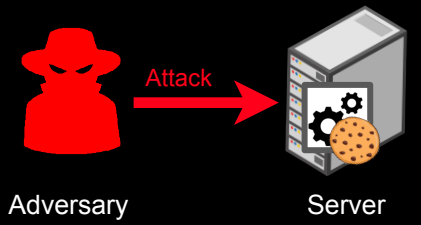


Cryptographically computing cookies in software is costly

Cryptographic cookie computation and packet processing for every potential connection

Computational strain becomes new attack vector

Server CPU capacity easily overwhelmed under heavy loads



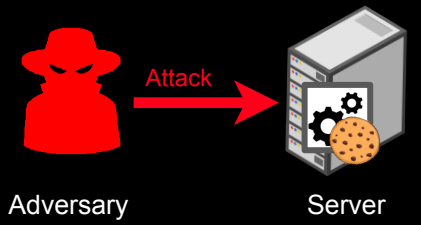
Cryptographically computing cookies in software is costly

Cryptographic cookie computation and packet processing for every potential connection

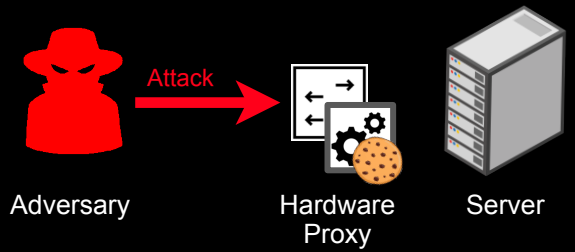
Computational strain becomes new attack vector

Server CPU capacity easily overwhelmed under heavy loads

Application performance degraded, clients experience DoS (again)

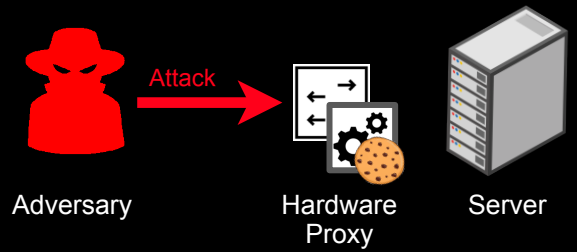


Why not move the cookie defense to a hardware proxy?



Why not move the cookie defense to a hardware proxy?

…like a high-speed programmable switch!



Why is a secure, scalable, and performant defense so hard?

Software-Only Solutions Can’t Scale Compute

Computational cost to identify attacks leads to early CPU exhaustion.

Hardware-Only Solutions Can’t Scale Memory

Hardware-Only Solutions Are Insecure

Why is a secure, scalable, and performant defense so hard?

Software-Only Solutions Can’t Scale Compute

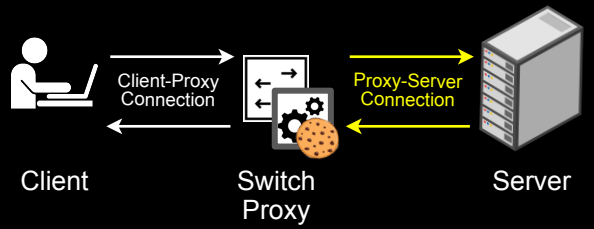
Computational cost to identify attacks leads to early CPU exhaustion.

Hardware-Only Solutions Can’t Scale Memory

Usage of limited memory compromises performance.

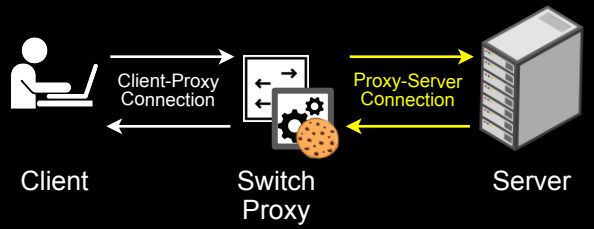
Hardware-Only Solutions Are Insecure

Tracking verified connections in hardware is costly



Tracking verified connections in hardware is costly

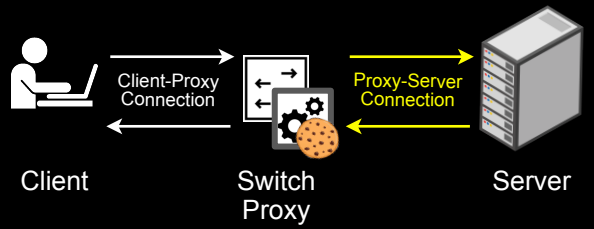
Header translation required between client-proxy and proxy-server



Tracking verified connections in hardware is costly

Header translation required between client-proxy and proxy-server

Switch proxy must keep per-flow state for ongoing connections

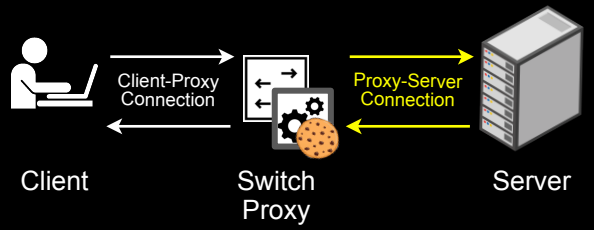


Tracking verified connections in hardware is costly

Header translation required between client-proxy and proxy-server

Switch proxy must keep per-flow state for ongoing connections

Exhausts limited memory of high-speed switch hardware

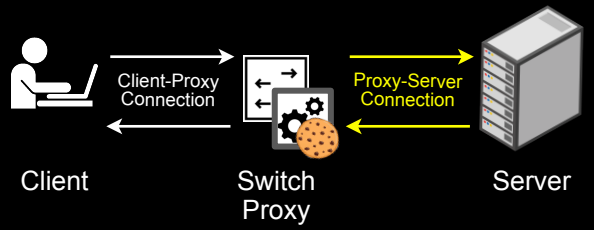


Tracking verified connections in hardware is costly

Header translation required between client-proxy and proxy-server

Switch proxy must keep per-flow state for ongoing connections

Exhausts limited memory of high-speed switch hardware



Jaqen[1] avoids memory usage, at a performance cost (extra RTT and added latency for all benign flows)

[1] Liu, et al. *Jaqen: A High-Performance Switch-Native Approach for Detecting and Mitigating Volumetric DDoS Attacks with Programmable Switches.* USENIX Security Symposium, 2021.

Why is a secure, scalable, and performant defense so hard?

Software-Only Solutions Can’t Scale Compute

Computational cost to identify attacks leads to early CPU exhaustion.

Hardware-Only Solutions Can’t Scale Memory

Usage of limited memory compromises performance.

Hardware-Only Solutions Are Insecure

Weak hashing for cookie generation breaks security.

Insecure hashing breaks cookie security

Insecure hashing breaks cookie security

Cookie = hash(4-tuple, secret)

```
4-tuple = [src_ip, dst_ip, src_port, dst_port]
```

Insecure hashing breaks cookie security

Cookie = hash(4-tuple, secret)

```
4-tuple = [src_ip, dst_ip, src_port, dst_port]
```

Hash must be strong enough to withstand manipulated hash collisions

Insecure hashing breaks cookie security

Cookie = hash(4-tuple, secret)

```
4-tuple = [src_ip, dst_ip, src_port, dst_port]
```

Hash must be strong enough to withstand manipulated hash collisions

Hardware solutions (Jaqen[1], Poseidon[2]) rely on CRC Checksum for hashing - insecure!

[1] Liu, et al. *Jaqen: A High-Performance Switch-Native Approach for Detecting and Mitigating Volumetric DDoS Attacks with Programmable Switches.* USENIX Security Symposium, 2021.
[2] Zhang, et al. *Poseidon: Mitigating volumetric DDoS attacks with programmable switches.* Network and Distributed System Security Symposium, 2020.

Insecure hashing breaks cookie security

Cookie = hash(4-tuple, secret)

```
4-tuple = [src_ip, dst_ip, src_port, dst_port]
```

Hash must be strong enough to withstand manipulated hash collisions

Hardware solutions (Jaqen[1], Poseidon[2]) rely on CRC Checksum for hashing - insecure!

Security abandoned, compute AND memory consumed

[1] Liu, et al. *Jaqen: A High-Performance Switch-Native Approach for Detecting and Mitigating Volumetric DDoS Attacks with Programmable Switches.* USENIX Security Symposium, 2021.
[2] Zhang, et al. *Poseidon: Mitigating volumetric DDoS attacks with programmable switches.* Network and Distributed System Security Symposium, 2020.

Why is a secure, scalable, and performant defense so hard?

Software-Only Solutions Can’t Scale Compute

Computational cost to identify attacks leads to early CPU exhaustion.

Hardware-Only Solutions Can’t Scale Memory

Usage of limited memory compromises performance.

Hardware-Only Solutions Are Insecure

Weak hashing for cookie generation breaks security.

Outline

Motivation

SmartCookie

Results

Conclusion

SmartCookie solves these challenges! 

Key Insight:

modern SYN flood defenses must be layered,

a *collaborative split-layer design* of hardware + software

SmartCookie solves these challenges! 

Intelligent division of labor

SmartCookie solves these challenges! 

Intelligent division of labor

- What functionality should be partitioned?
- How should it be partitioned?

What functionality should be partitioned?

What functionality should be partitioned?

We observe three key elements of SYN cookie proxy defenses

(F1) Cookie checks

(F2) Header translations

(F3) Keeping state for verified connections

How should this functionality be partitioned?

(F1) Cookie checks

(F2) Header translations

(F3) Keeping state for verified connections

How should this functionality be partitioned?

(F1) Cookie checks

(F2) Header translations

(F3) Keeping state for verified connections

Insight 1: Switches are highly performant, but severely memory-limited

How should this functionality be partitioned?

(F1) Cookie checks

(F2) Header translations

(F3) Keeping state for verified connections

Insight 1: Switches are highly performant, but severely memory-limited

…switches are an excellent first line of defense, but should not keep per-flow state!

How should this functionality be partitioned?

(F1) Cookie checks

(F2) Header translations

(F3) Keeping state for verified connections

Insight 1: Switches are highly performant, but severely memory-limited
…switches are an excellent first line of defense, but should not keep per-flow state!

Insight 2: Servers are provisioned with memory for benign flows, but they are slower

How should this functionality be partitioned?

(F1) Cookie checks

(F2) Header translations

(F3) Keeping state for verified connections

Insight 1: Switches are highly performant, but severely memory-limited
…switches are an excellent first line of defense, but should not keep per-flow state!

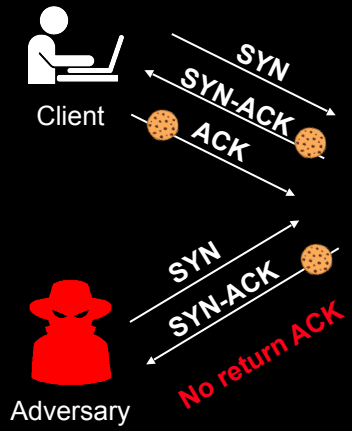
Insight 2: Servers are provisioned with memory for benign flows, but they are slow
…servers are ideal for exactly tracking benign flows, but should not block attacks!

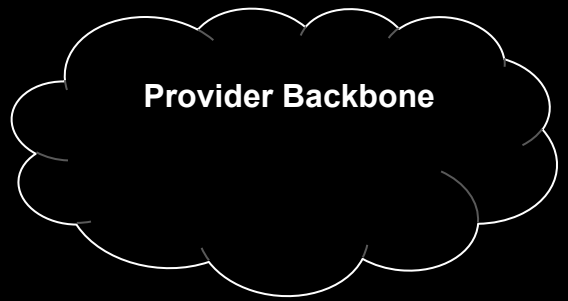
SmartCookie's Split-Proxy Architecture

(F1) Cookie checks

(F2) Header translations

(F3) Keeping state for verified connections

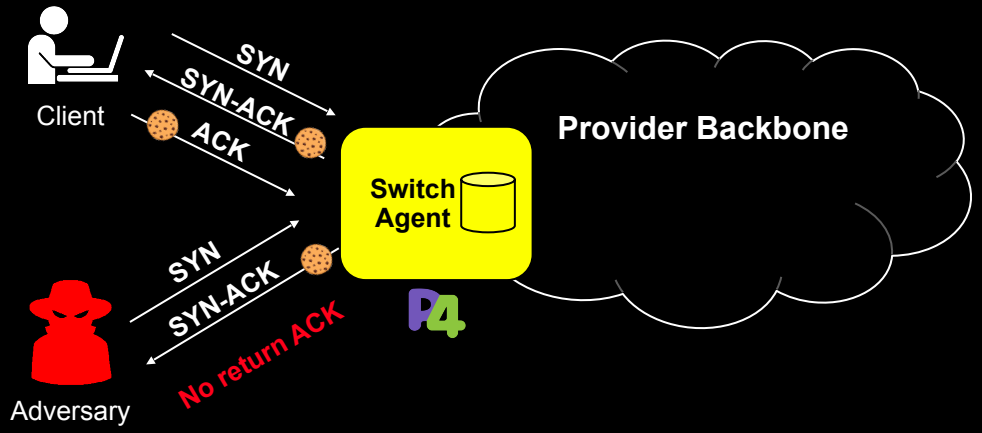




SmartCookie's Split-Proxy Architecture

Switch agent performs *secure* cookie checks with a robust hash, not CRC (F1)

(F1) Cookie checks	(F2) Header translations	(F3) Keeping state for verified connections

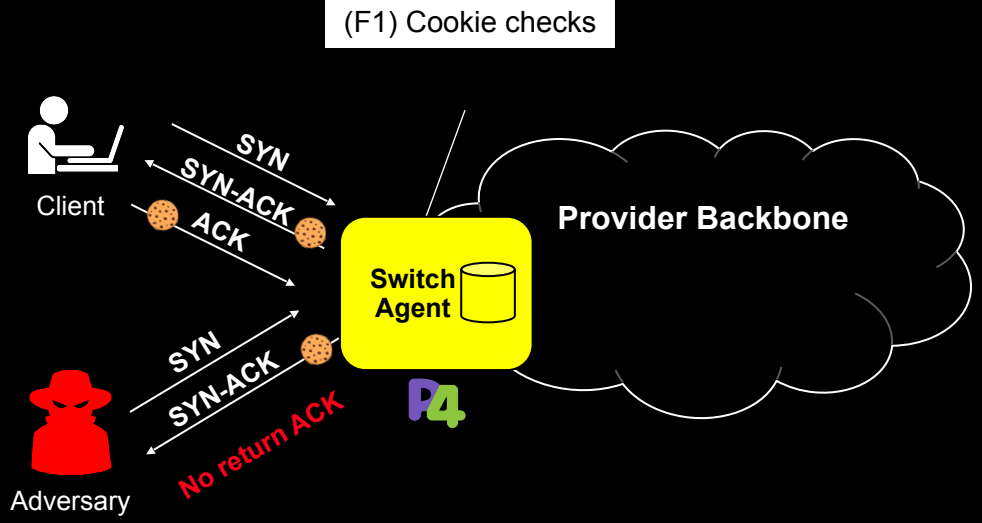


SmartCookie's Split-Proxy Architecture

Switch agent performs *secure* cookie checks with a robust hash, not CRC (F1)

(F2) Header translations

(F3) Keeping state for verified connections



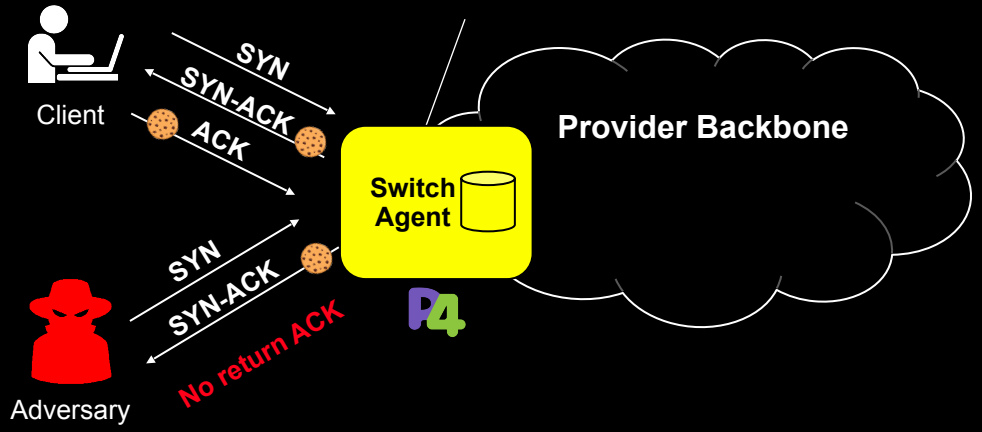
SmartCookie's Split-Proxy Architecture

Switch agent *approximately* tracks verified connections (F3.A)

(F2) Header translations

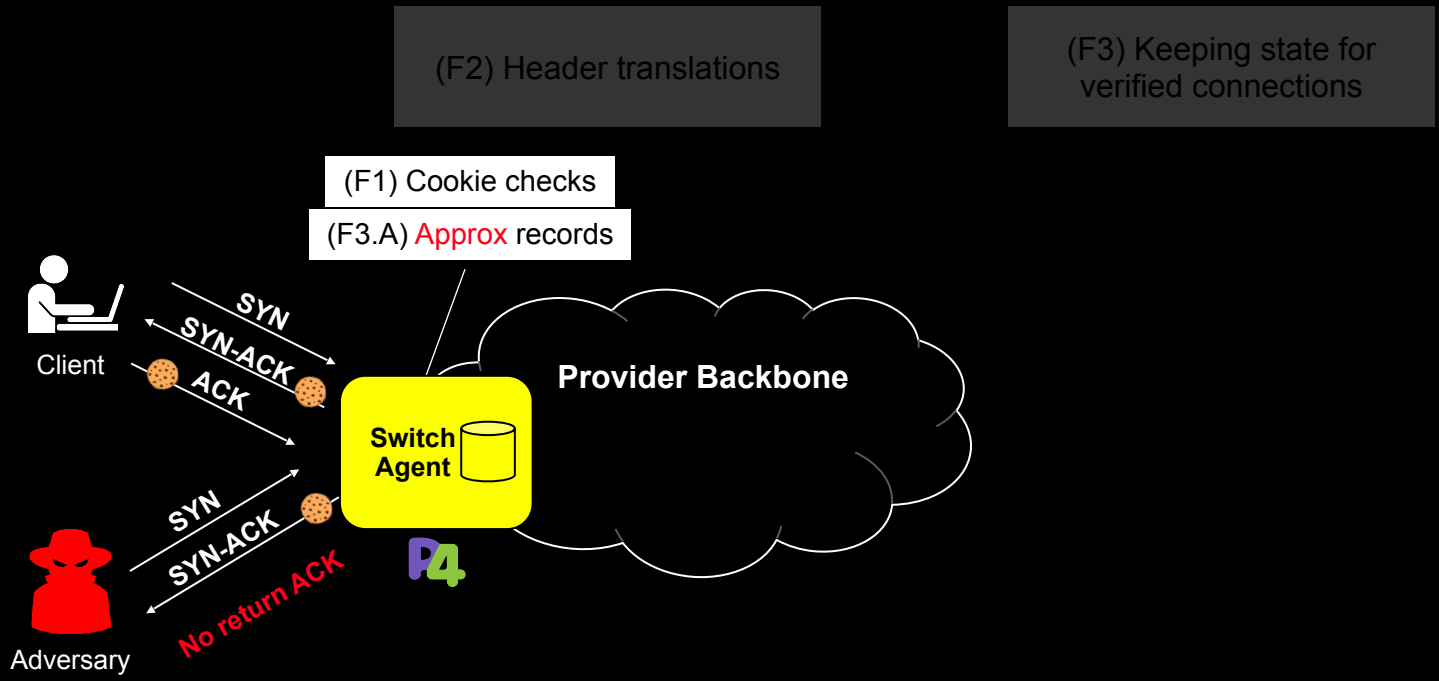
(F3) Keeping state for verified connections

(F1) Cookie checks



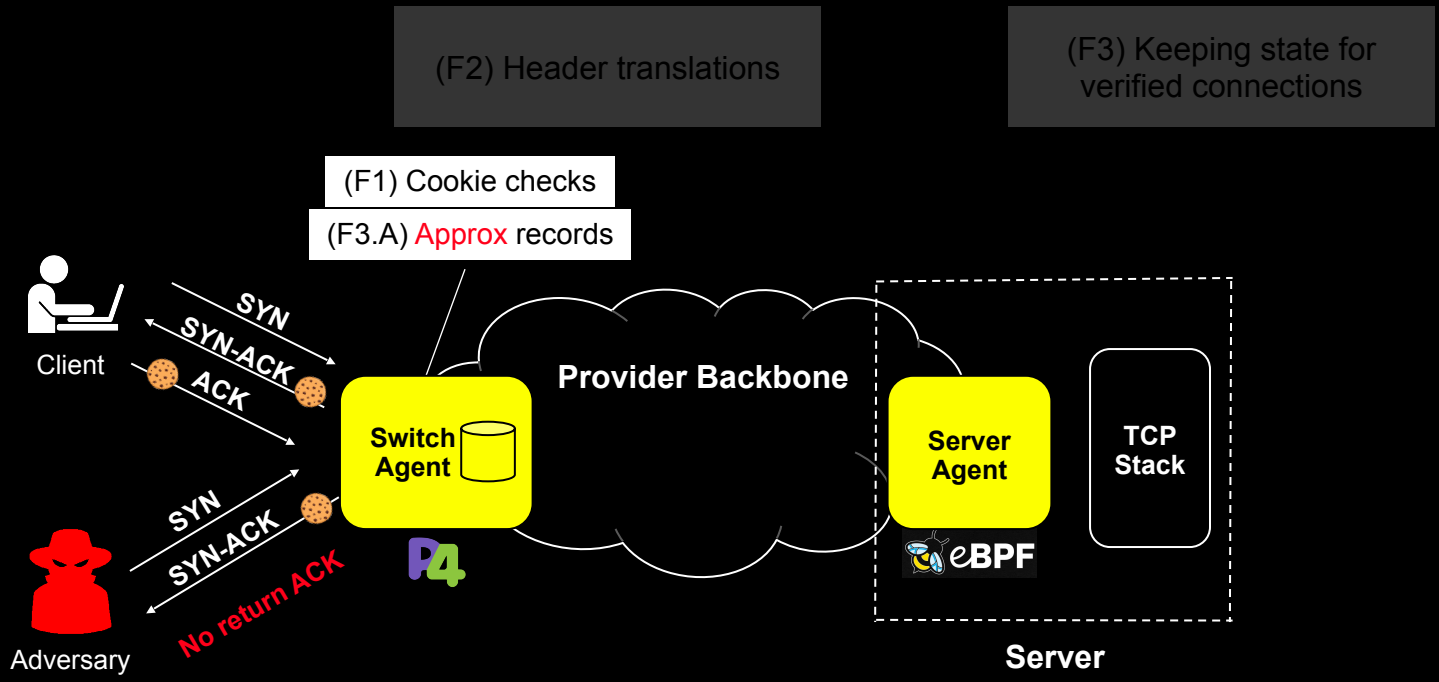
SmartCookie's Split-Proxy Architecture

Switch agent *approximately* tracks verified connections (F3.A)



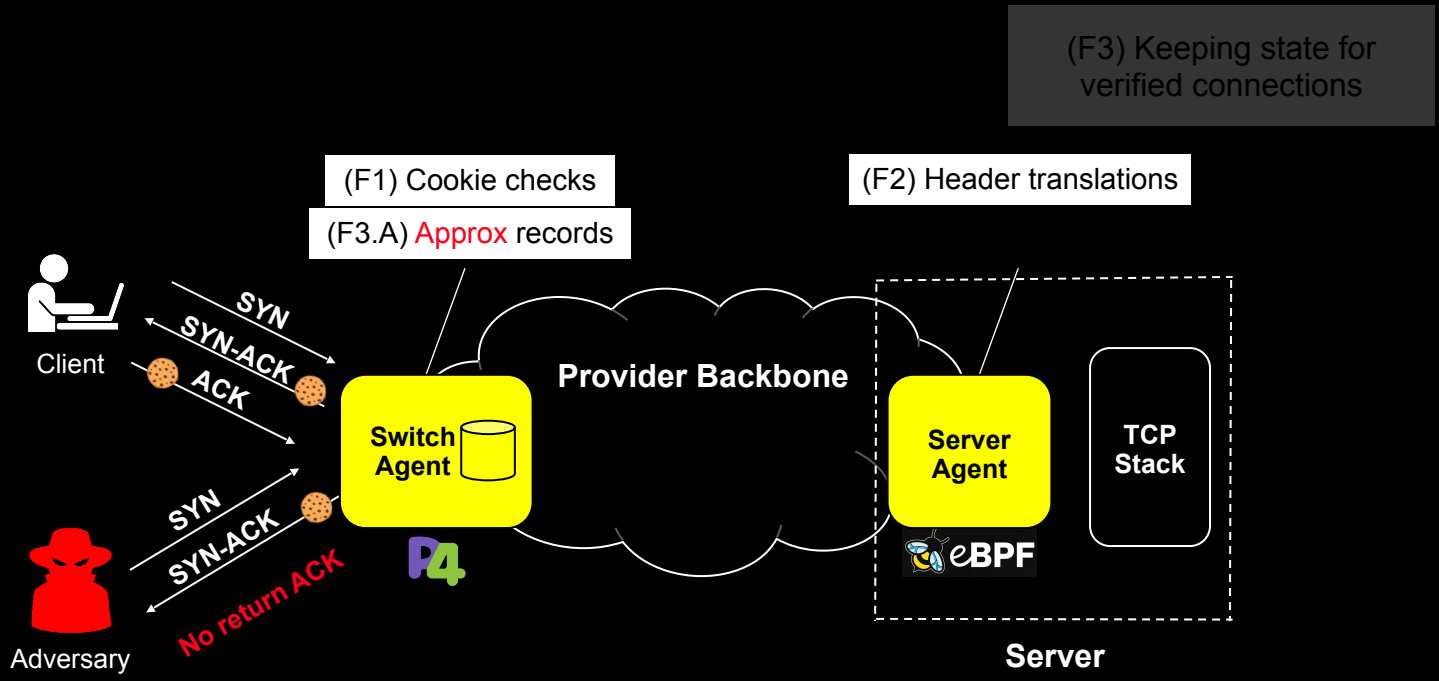
SmartCookie's Split-Proxy Architecture

Server agent handles header translations on behalf of switch agent (F2)



SmartCookie's Split-Proxy Architecture

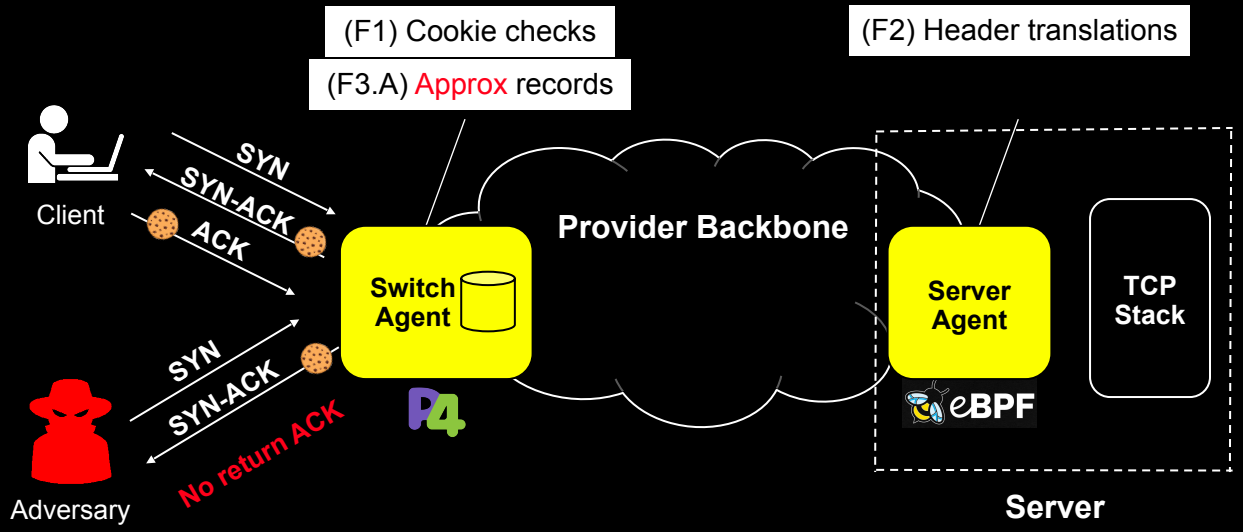
Server agent handles header translations on behalf of switch agent (F2)



SmartCookie's Split-Proxy Architecture

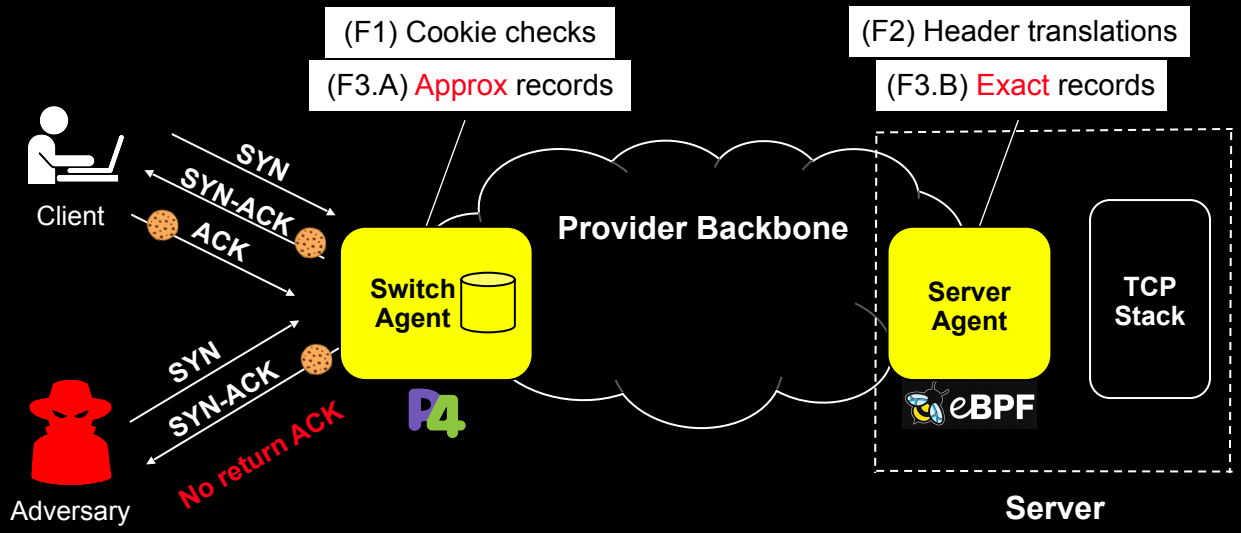
Server agent *exactly* tracks verified connections (F3.B)

(F3) Keeping state for verified connections



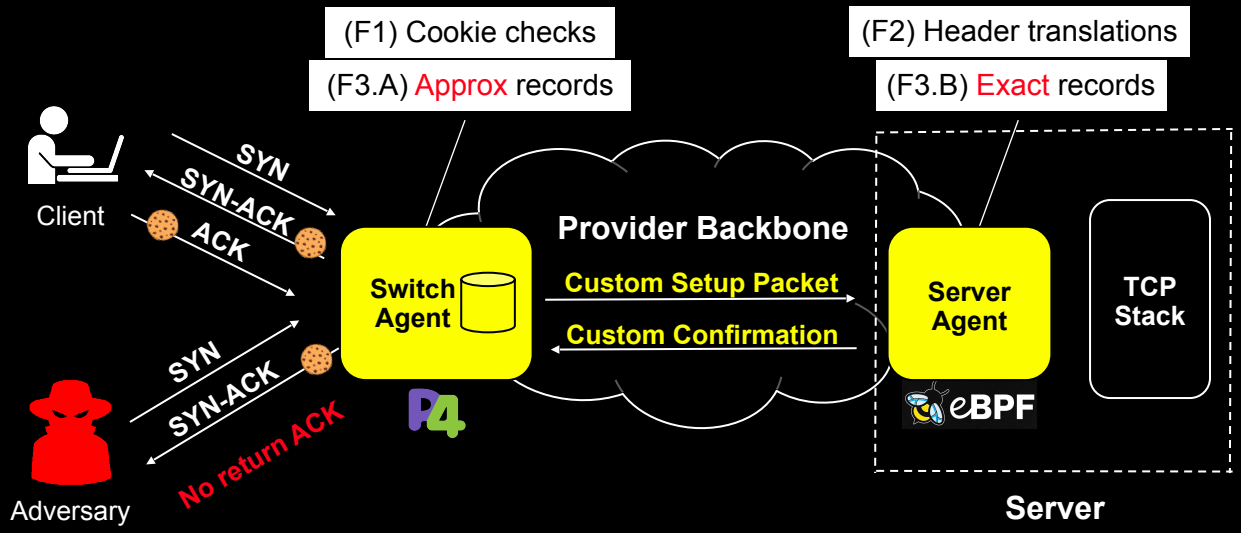
SmartCookie's Split-Proxy Architecture

Server agent *exactly* tracks verified connections (F3.B)



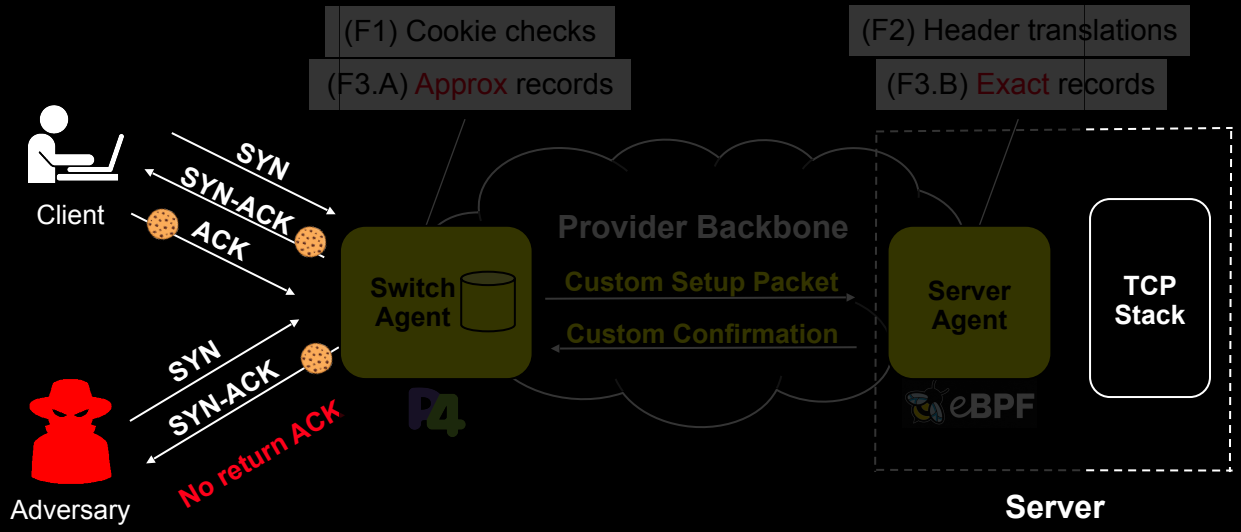
SmartCookie's Split-Proxy Architecture

Custom collaborative protocol between SmartCookie components



SmartCookie's Split-Proxy Architecture

…does not require any modifications to the client or server's network stack



SmartCookie delivers security, scalability, and performance

Robust hashing for **secure cookies** on switch (F1), §5

Approximate data structures for switch memory scalability (F3.A), §6.3 - §6.4

Efficient server-side eBPF support with **exact** memory (F2) + (F3.B), §6.1 - §6.2

Please read our paper for more details!

Outline

Motivation

SmartCookie

Results

Conclusion

Evaluation

Can SmartCookie deliver *security* at high attack rates?

Can SmartCookie protect CPU capacity for *scalability*?

Can SmartCookie maintain client *performance* under attack?

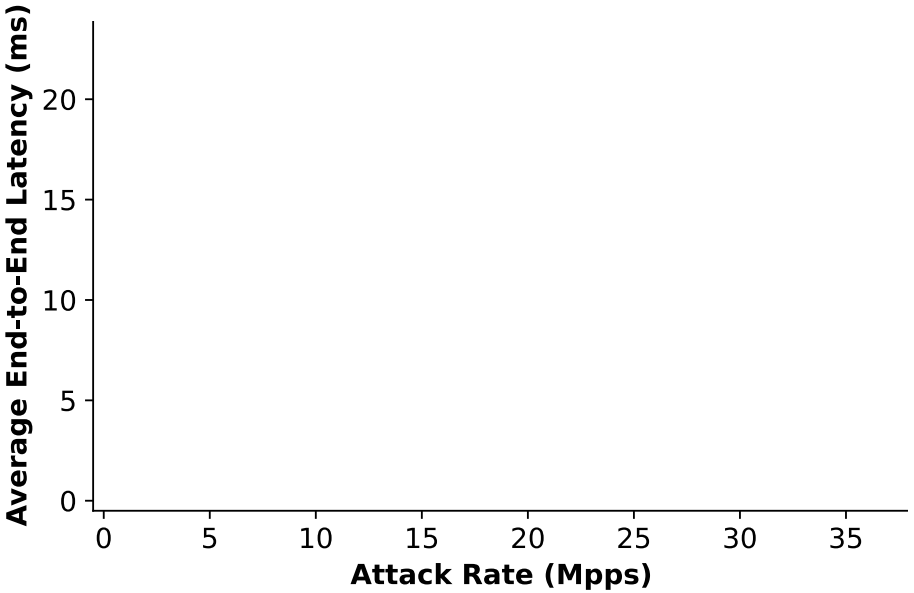
Evaluation

Can SmartCookie deliver *security* at high attack rates?

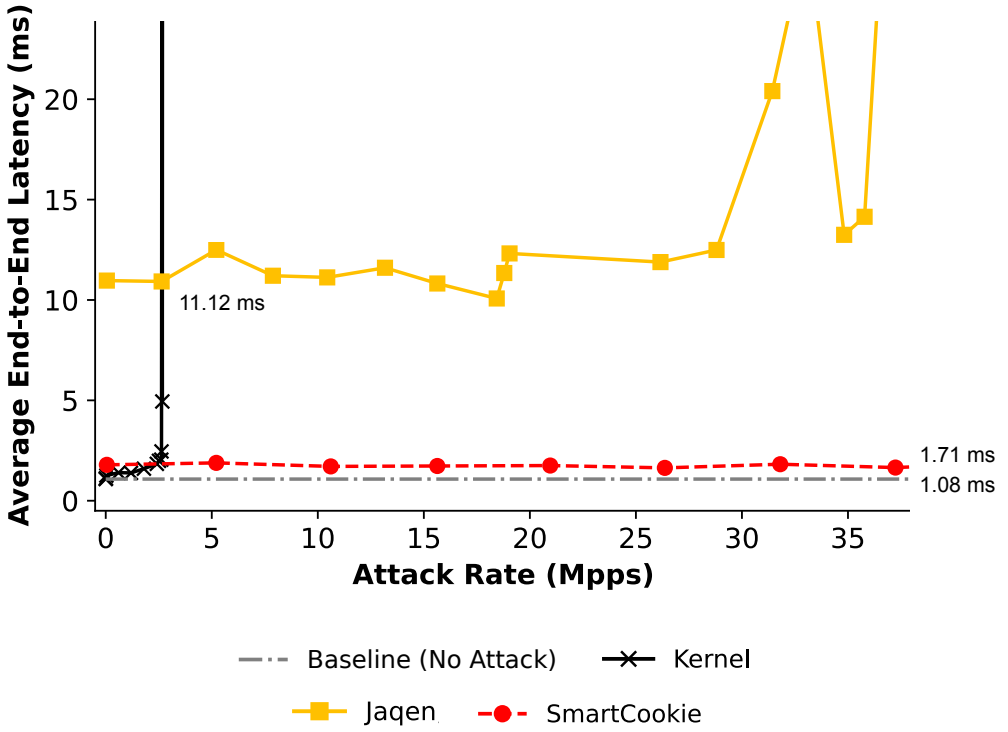
Can SmartCookie protect CPU capacity for *scalability*?

Can SmartCookie maintain client *performance* under attack? this talk

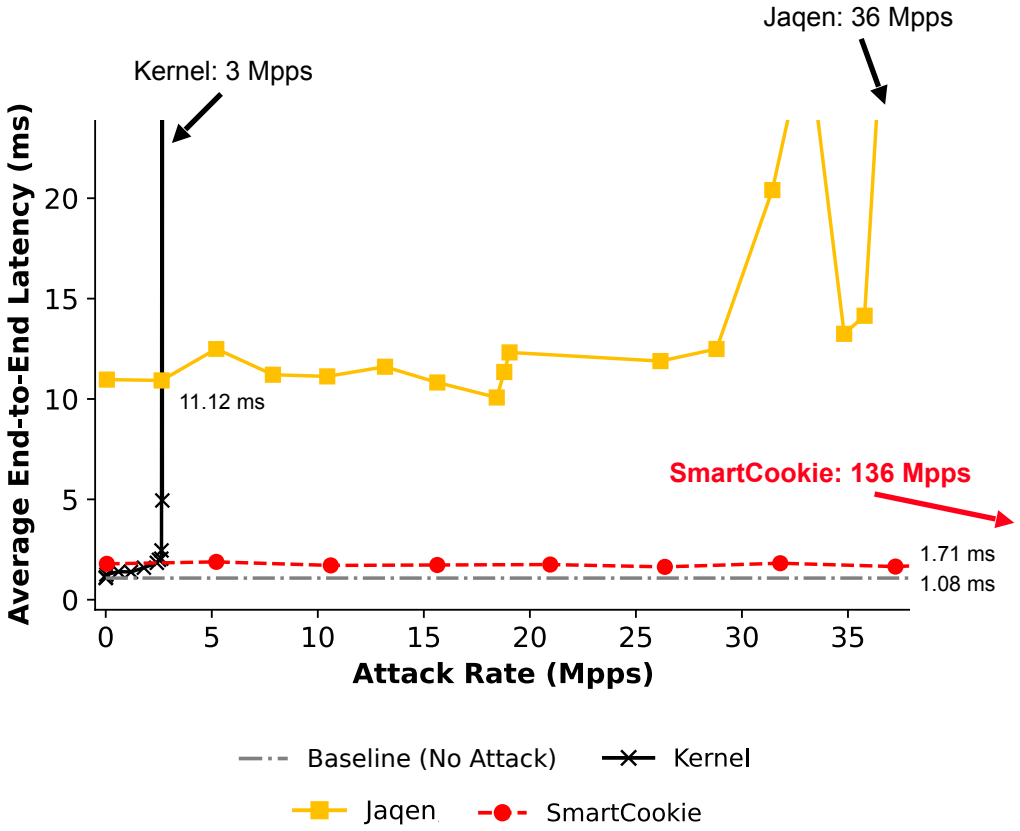
Protecting performance



Protecting performance



Protecting performance



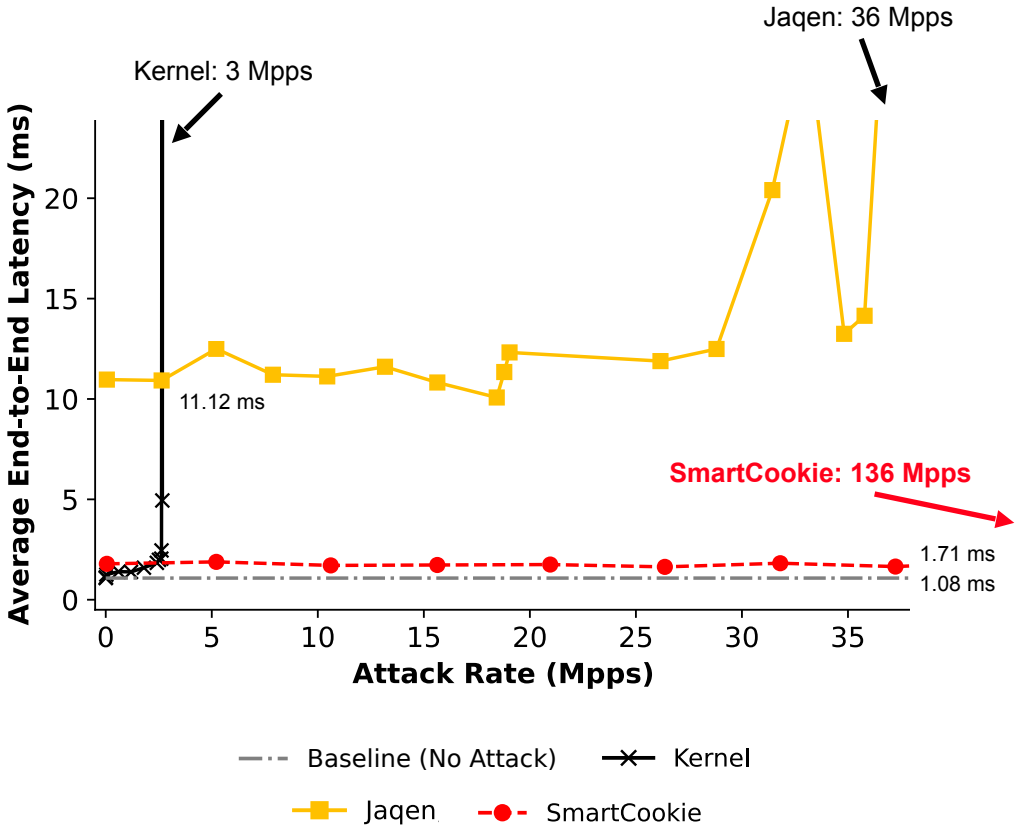
Protecting performance

Latency reduced 48-84% vs. Jaqen

Close to baseline latency without attack

In fact, zero packet loss until 136 Mpps

Throughput outperforms
- Jaqen by one order of magnitude
- Kernel by two orders of magnitude



Outline

Motivation

SmartCookie

Results

Conclusion

SmartCookie: Layered Hardware+Software Codesign

PRINCETON UNIVERSITY

sophiayoo@princeton.edu

https://github.com/Princeton-Cabernet/p4-projects/tree/master/SmartCookie

Live demo video on YouTube

usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION Artifact Evaluation

SmartCookie: Layered Hardware+Software Codesign

A split-proxy defense approach that

(i) exploits division of labor across targets;

(ii) with approximation in early layers but exact overall results;

(iii) to provide security, scalability, *and* performance

PRINCETON UNIVERSITY

sophiayoo@princeton.edu

https://github.com/Princeton-Cabernet/p4-projects/tree/master/SmartCookie

Live demo video on YouTube

usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION Artifact Evaluation

SmartCookie: Layered Hardware+Software Codesign

SmartCookie

A split-proxy defense approach that

(i) exploits division of labor across targets;

(ii) with approximation in early layers but exact overall results;

(iii) to provide security, scalability, *and* performance

Vision for future work

Split-layer design for other protocols, volumetric attacks, and IDS/IPS systems

…reducing processing and traffic inspection loads by moving parts earlier on path

PRINCETON UNIVERSITY

sophiayoo@princeton.edu

https://github.com/Princeton-Cabernet/p4-projects/tree/master/SmartCookie

Live demo video on YouTube

usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION Artifact Evaluation