The Garden of Forking Paths

Alex Gaynor
Developer, Maintainer

Cryptographic libraries in the Python ecosystem

Bob Beck
Developer,

OpenSSH, OpenSSL, BoringSSL and LibreSSL

James Fuller
Developer, Maintainer

curl/libcurl
curl-security team

Pedro Monreal
Developer, Maintainer

OpenSSL and other cryptographic libraries

Dmitry Belyavskiy
Developer, Maintainer

OpenSSL and OpenSSH

What this panel is about

We're exploring the balance between *upstream and downstream*, *diversity and fragmentation*, and *responsibility and freedom* in open-source cryptography.

Projects like OpenSSL, LibreSSL, BoringSSL, and others coexist — sometimes in harmony, sometimes in tension.

This discussion brings together maintainers and engineers from across ecosystems to ask:
How do we collaborate without collapsing, and evolve without fragmenting?

Isn’t maintaining out-of-tree patches

just creating forks while pretending not to?

When a distro ships a “patched upstream project,” are they misleading users by still calling it same name?

Do forks do anything good for users?

How to avoid more forks?

In ten years, will we have more or fewer major SSL/TLS forks?

At what point does “healthy diversity”

become “dangerous fragmentation”?

Do forks exist because OpenSSL

is too big / too complex?

What did forks get right/wrong?

Is it the fault of upstream when downstream doesn’t push code upstream?

Who owns the “blast radius” when a distro patch breaks things?

Should we adopt a norm:

no downstream-only engineering?

How can two upstreams with different views coexist?

Bounty program — good or bad?

Closing words