



نحو استفاده از چارچوب پیشنهادی

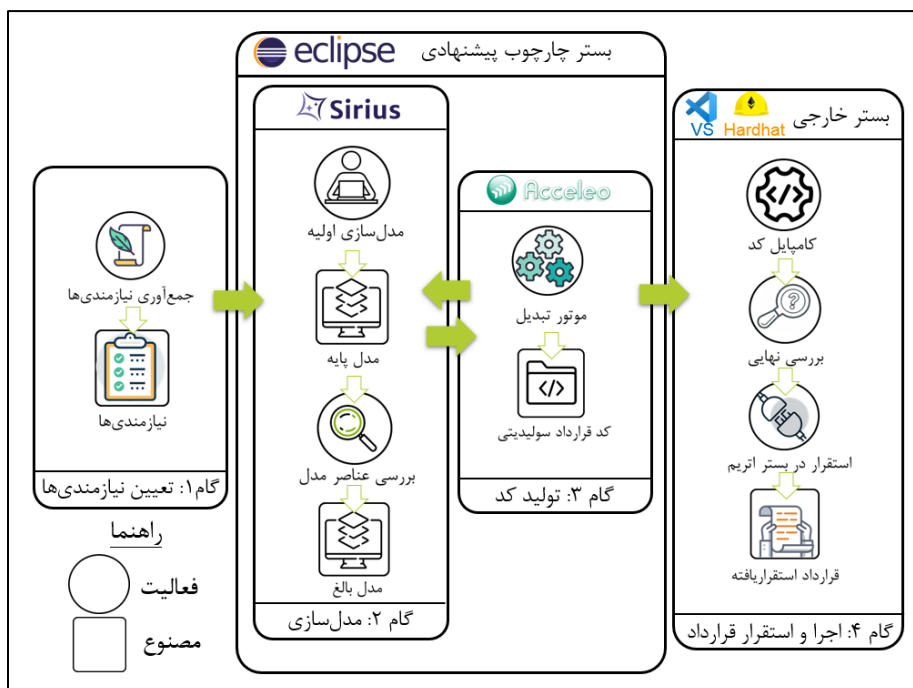
برای استفاده از چارچوب پیشنهادی گام‌های زیر را دنبال نمایید.

گام ۱: آشنایی با فرآیند استفاده از چارچوب پیشنهادی

پیش‌نیاز: قبل از شروع به کار لازم است که مستند «راهنمایی نصب» را با دقت مطالعه و در نرم‌افزار اکلیپس دنبال کنید تا

چارچوب پیشنهادی بر روی بستر اکلیپس استقرار یابد.

برای کار با چارچوب پیشنهادی شکل ۱ ارائه شده است. در گام نخست به جمع‌آوری نیازمندی‌های قرارداد هوشمند خود بپردازید و فهرستی از نیازمندی‌های لازم را جمع‌آوری کنید. در گام دوم، در بخش مدل‌سازی ابتدا با توجه به نیازمندی‌های خود و مفاهیم پایه‌ی به مدل‌سازی سرویس‌های قرارداد هوشمند بپردازید تا یک مدل بالغ از قرارداد هوشمند بدست آید. در این راستا، برای بالغ‌سازی این مدل گام‌های مربوطه‌ای نظیر اعتبارسنجی^۱ (ساختاری یا معنایی) را طی نمایید. در گام سوم، با راه‌اندازی موتور تبدیل می‌توانید یک قدم به کد اجرایی نزدیک‌تر شوید. برای این منظور، ابتدا در نوار ابزار اکلیپس، بخش کنسول را بر روی بُوم مدل‌سازی فعال کنید تا قابل رؤیت باشد. سپس بر روی دکمه‌ی اجرای موتور تبدیل (دکمه Run، ماژول build) کلیک نمایید، موتور تبدیل براساس مولفه‌های داخلی نظیر مولفه‌ی اعتبارسنجی و یا مولفه‌ی مشاور امنیتی شما را به گام ۲ و یا گام ۴ از طریق کنسول هدایت خواهد نمود. اگر کنسول به شما اظهارهای ساخت مدل را نمایش داد به مدل مربوطه مراجعه کنید و خطاهای مربوطه را رفع نمایید تا مدل قابل پذیرشی برای موتور تبدیل ایجاد شود. در غیر این صورت به بستر خارجی بروید و کد تولید شده را دریافت نمایید. در بستر هاردت براساس صلاحیت‌های می‌توانید رویه‌ی تست یک (فاز بررسی نهایی) و یا استقرار بنا به کلیدهای رابط برنامه‌نویسی را دنبال نمایید (بخش تنظیمات شبکه هاردت^۲).



شکل ۱- فرآیند استفاده از چارچوب پیشنهادی جهت تولید کد امن قراردادهای هوشمند

^۱ برای اعتبار سنجی روی مدل توصیفی، از طریق موس راست کلیک کنید و گزینه‌ی validate diagram را کلیک نمایید.


























^۲ Network Setting (<https://hardhat.org/tutorial/deploying-to-a-live-network#deploying-to-remote-networks>)

برای کار با بستر خارجی استفاده از دو نرم افزار ویژوال کد^۳ (به همراه نصب افزونه^۴ solidity) و کنسول هایپر^۵ را پیشنهاد می کنیم. پس از دریافت کد اجرایی به بستر درایور هارد هت مراجعه کنید و با راست کلیک نمودن در محیط درایور، کنسول هایپر را اجرایی کنید و در محیط کنسول هایپر دستور «npx hardhat compile» را وارد نمایید تا بایت کد قرارداد تولید شود. برای اجرایی کردن تست یک در کنسول هایپر نیز می توانید از دستور «npx hardhat test» استفاده نمایید. برای آموزش و نوشتن تست یک می توانید به بخش آموزش های هارد هت^۶، موکا^۷ و وفل^۸ مراجعه کنید.

گام ۲: مفاهیم پایه مدل سازی

برای مدل سازی در محیط ویرایشگر گرافیکی می بایست از نمادگان جدول ۱ استفاده نماید. برای این منظور به تشریح تکمیلی این نمادگان خواهیم پرداخت. شایان ذکر است که با کشیدن و رها کردن هر کدام از عناصر موجود به بوم مدل سازی، تکه ای به مدل در حال ساخت افزوده می شود. علاوه بر آن، از بخش جعبه ویژگی ها می توانید به این عناصر شکل دهید و یا مشخصه ای اجرایی برای آن ها در نظر بگیرید. چنین مشخصه های بنا به رویه ی ساخت توسط موتور تبدیل در کد اجرایی بازتاب خواهند یافت.

جدول ۱- نمادگان گرافیکی به کار رفته در محیط ویرایشگر گرافیکی سیریوس

Entity	Short Description	Entity	Short Description	Entity	Short Description	Entity	Short Description	Entity	Short Description
	create smart contract		add token to ERC1155		import external library		create transaction		create authorization
	create service		create contract instance		create condition		create requirement		binding a service
	Incoming objects		create participant		create iteration		create event		
	outgoing objects		create ledger		create comment		create instruction		
	create token		create structure		log to hardhat		create selfdestruct		
	create pulloverpush service		create data				create commit reveal		
Service Orientation		Structural			Behavioural				

در ابتدا صحبت به بخش ۴-۴-۲ (نحو انتزاعی و طراحی فرامدل یکپارچه) از پایان نامه مراجعه کنید و کلیت مفاهیم را مطالعه نمایید، موارد بیان شده در بخش مربوطه با مدل سازی همسو هستند. از این رو، در بخش فعلی سعی بر آن داریم تا چکیده ای از مفاهیم مورد نیاز را بنا به رویه ی پرسش و پاسخ بازگو کنیم.

- چگونه کنترل دسترسی مبتنی بر نقش را در بوم مدل سازی ایجاد نمایم؟

برای ساخت کنترل دسترسی مبتنی بر نقش، از ترکیب گره های دفترکل (Ledger) و ثبت ویژگی ACL در آن، در هر نقطه ای مجاز از فضای قرارداد هوشمند و احراز مجوز (Authorization) استفاده کنید. نقش توصیف شده (نقش مدنظر) را به نام دفترکل انتصاب دهید و نام دفترکل را به گره احراز مجوز در سرویس مربوطه معرفی کنید. موتور تبدیل در زمان تولید کد، تمامی نقش های مربوطه و گره های دسترسی مجاز بر روی سرویس های هدف را به ترتیب اولویت بنا به الگوی ساخت کنترل دسترسی مبتنی بر نقش تولید خواهد نمود. توجه داشته باشید که دفترکل دارنده نقش مربوطه می بایست از جنس آدرس و مقادیر بولی باشد.

³ VisualCode (<https://code.visualstudio.com>)

⁴ Solidity Extension (<https://marketplace.visualstudio.com/items?itemName=Eleven01.Solidity>)

⁵ Hyper (<https://hyper.is>)

⁶ Unit-Test By Hardhat (<https://hardhat.org/tutorial/testing-contracts#5.-testing-contracts>)

⁷ MochaJs (<https://mochajs.org>)

⁸ Waffle (<https://ethereum-waffle.readthedocs.io/en/latest>)



- چگونه کنترل دسترسی پایه را در بوم مدل سازی ایجاد نمایم؟
برای ساخت کنترل دسترسی پایه از گره های مشارکت کننده و ثبت ویژگی ACL در آن، در هر نقطه ای مجاز از فضای قرارداد هوشمند و احراز مجوز (Authorization) استفاده کنید. نام مشارکت کننده را به گره احراز مجوز از سرویس مربوطه معرفی کنید. موتور تبدیل در زمان تولید کد، گره دسترسی را بر روی سرویس هدف اعمال خواهد نمود.
- مشخصه های `msg.sender`، `msg.value` و `msg.data` در مدل سازی به چه معنایی هستند و چه کاربردی دارند؟
برای همسوسازی مفاهیم از این سه مشخصه برای معنادار کردن بخش مدل سازی استفاده شده است. برای آن که مطلب برای خوانندگان گرامی روشن شود به شرح یک مثال اکتفا می کنیم. فرض کنید، آلیس یک کاربر اتریم است که دارای دو زوج کلید عمومی و خصوصی است. آلیس به سرویس پرداختی از یک قرارداد هوشمند، مقداری رمزارز اتر به همراه یک داده خاص ارسال می نماید. حال سرویس مذکور از طریق مشخصه `msg.sender` به کلید عمومی آلیس (یا مبدا تراکنش) پی می برد و از طریق مشخصه `msg.value` به میزان رمزارز ارسالی پی خواهد برد و از طریق مشخصه `msg.data` داده ای ارسالی خاص آلیس را دریافت خواهد نمود. به زبان ساده این سه مشخصه آدرس فرستنده تراکنش، مقدار رمزارز ارسالی و داده ای ارسالی به سرویس ها هستند. کاربر نهایی در چارچوب پیشنهادی می تواند از این سه مشخصه استفاده نماید تا بحث سرویس گرایی قرارداد را تکمیل نماید. شایان ذکر است که این سه مشخصه در محیط اتریم نیز نمود معنایی دارند و برای توسعه دهندگان اتریم نیز گویا هستند.
- در سرویس قرارداد هوشمند چگونه منطق سرویس را پیاده سازی نمایم؟
برای انجام چنین کاری گره دستورالعمل از جعبه ابزار را بر روی سرویس مربوطه کشیده و رها کنید و سپس در جعبه ویژگی دستورالعمل های مد نظر خود را وارد نمایید. دستورالعمل های یک سرویس می تواند از جنس کاهش و یا افزایش حساب، فرامین منطقی و یا سایر گزاره های انتصابی باشند. توجه داشته باشید که محدودیتی برای توصیف گزاره های اجرایی در نوع و یا تعدد آنها نیست.
- آیا امکان غیرفعال سازی پوشش یک نوع آسیب پذیری در بخش مدل سازی وجود دارد؟
پاسخ روشن است، کاربر نهایی می تواند از طریق اعمال نکردن برخی از مشخصه های اجرایی پوشش یک آسیب پذیری را به صورت صلاح دیدی لغو نماید.
- برای ساخت داده های پویا و یا درهم چه کنیم؟ آیا چارچوب حاضر بر خلاف سایرین از این ویژگی ها حمایت می کند؟
پاسخ مثبت است، در چارچوب حاضر پوشش داده های پویا و پیچیده نیز محقق شده است. برای این منظور می توانید گره داده را در هر فضای مجاز از قرارداد هوشمند اعمال نماید و نوع داده را از جنس پویا قرار دهید. علاوه بر آن، امکان ساخت سازه های با چندین داده نیز محقق شده است و محدودیتی در ساخت و یا تعدد آنها نیست. مدیریت و بروزرسانی چنین داده های از طریق گره دستورالعمل در فضای سرویس ها قابلیت انجام دارد.
- برای آدرس دهی درایور هاردت به کدام بخش از موتور تبدیل مراجعه کنیم؟
به ماژول `copytodriver.java` از بخش سرویس مراجعه کنید و در بخش ابتدای از این ماژول، آدرس درایور هاردت را تنظیم نمایید.